

9.3

*IBM MQ Administration Reference*

**IBM**

**Note**

Before using this information and the product it supports, read the information in [“Notices” on page 2815.](#)

This edition applies to version 9 release 3 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 2007, 2024.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents



<b>Administration reference.....</b>	<b>5</b>
Commands reference.....	5
Command sets comparison.....	5
How to read syntax diagrams.....	19
IBM MQ control commands reference.....	21
MQSC commands reference.....	251
Programmable command formats (PCFs) reference.....	983
CL commands for IBM i reference.....	1551
MFT commands reference.....	1977
MQIPT commands reference.....	2153
Administrative REST API reference.....	2159
REST API resources.....	2159
REST API and PCF equivalents.....	2379
IBM MQ Administration Interface reference.....	2402
MQAI calls.....	2402
MQAI selectors.....	2483
Managed File Transfer administration reference.....	2485
How MFT agents allocate source transfer slots to new requests.....	2485
MFT agent status values.....	2485
MFT process controller overview.....	2486
MFT agent process controller status values.....	2488
MFT logger status values.....	2489
MFT logger process controller status values.....	2489
MFT process controller exit codes.....	2490
Guidelines for transferring files.....	2491
Regular expressions used by MFT.....	2523
Substitution variables for use with user-defined Connect:Direct processes.....	2523
Example: A Connect:Direct process file that calls MFT commands.....	2527
Restrictions of the Connect:Direct bridge agent.....	2528
FTPS server support by the protocol bridge.....	2528
SFTP server support by the protocol bridge.....	2529
FIPS support in MFT.....	2530
MFT database logger tables.....	2531
Authorities for the MFT logger.....	2546
File permissions for destination files.....	2547
MQ message properties set by MFT on messages written to destination queues.....	2548
IBM MQ message properties read by MFT from messages on source queues.....	2550
Guidance for setting MQ attributes and MFT properties associated with message size.....	2550
Guidance for specifying a wait time on a message-to-file transfer.....	2553
Available code pages for MFT.....	2554
How MFT agents use Java heap and native heap memory.....	2612
XML message formats used by MFT.....	2613
IBM MQ utilities on z/OS reference.....	2735
IBM MQ utilities on z/OS by category.....	2735
IBM MQ utility program (CSQUTIL) on z/OS.....	2739
The change log inventory utility (CSQJU003) on z/OS.....	2773
The print log map utility (CSQJU004) on z/OS.....	2781
The log print utility (CSQ1LOGP) on z/OS.....	2782
The queue sharing group utility (CSQ5PQSG) on z/OS.....	2793
The active log preformat utility (CSQJUFMT) on z/OS.....	2797
The dead-letter queue handler utility (CSQUDLQH) on z/OS.....	2798
The BSDS conversion utility (CSQJUCNV) on z/OS.....	2808

The message security policy utility (CSQ0UTIL).....	2809
Display queue manager information utility (CSQUDSPM).....	2810
<b>Notices.....</b>	<b>2815</b>
Programming interface information.....	2816
Trademarks.....	2816

## Administration reference

---

Use the links to reference information in this section to help you operate and administer IBM MQ.

- [“Commands reference” on page 5](#)
- [“Administrative REST API reference” on page 2159](#)
-  [“IBM MQ Administration Interface reference” on page 2402](#)
- [“Managed File Transfer administration reference” on page 2485](#)
-  [“IBM MQ utilities on z/OS reference” on page 2735](#)

### Related reference

[Queue names](#)

[System and default objects](#)

## Commands reference

---

Use commands to manage queue manager objects (control commands, MQSC commands, PCF commands), Managed File Transfer (MFT) objects, and IBM MQ Internet Pass-Thru.

[“Command sets comparison” on page 5](#)

[“How to read syntax diagrams” on page 19](#)

[“IBM MQ control commands reference” on page 21](#)

[“MQSC commands reference” on page 251](#)

[“Programmable command formats \(PCFs\) reference” on page 983](#)

[“CL commands for IBM i reference” on page 1551](#)

[“MFT commands reference” on page 1977](#)


[“MQIPT commands reference” on page 2153](#)


### Related concepts

[Ways of administering IBM MQ queue managers and associated resources](#)

### Command sets comparison

The tables in this section compare the facilities available for AIX, Linux, and Windows from the different administration command sets, and also show whether you can perform each function by using the IBM MQ Explorer or REST API.

**Note:**  These comparison tables do not apply to IBM MQ for z/OS®. For information on how to use MQSC commands and PCF commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

 These comparison tables do not apply to IBM MQ for IBM i. For information on how to use MQSC commands and PCF commands on IBM i, see [Alternative ways of administering IBM MQ for IBM i](#).

### Related concepts

[Ways of administering IBM MQ queue managers and associated resources](#)

### Related tasks

[Administering IBM MQ](#)

## Queue manager commands

A table of queue manager commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

*Table 1. Queue manager commands*

Description	PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
Change Queue Manager	<a href="#">Change Queue Manager</a>	<a href="#">ALTER QMGR</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
Create Queue Manager	No equivalent	No equivalent	<a href="#">crtmqm</a>	No equivalent	Yes
Delete Queue Manager	No equivalent	No equivalent	<a href="#">dlmqm</a>	No equivalent	Yes
Inquire Queue Manager	<a href="#">Inquire Queue Manager</a>	<a href="#">DISPLAY QMGR</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
Inquire Queue Manager Status	<a href="#">Inquire Queue Manager Status</a>	<a href="#">DISPLAY QMSTATUS</a>	<a href="#">dspmq</a>	GET /admin/installation GET /admin/qmgr	Yes
Ping Queue Manager	<a href="#">Ping Queue Manager</a>	<a href="#">PING QMGR</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	No
Refresh Queue Manager	<a href="#">Refresh Queue Manager</a>	<a href="#">REFRESH QMGR</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
Reset Queue Manager	<a href="#">Reset Queue Manager</a>	<a href="#">RESET QMGR</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	No
Start Queue Manager	No equivalent	No equivalent	<a href="#">strmqm</a>	No equivalent	Yes

Table 1. Queue manager commands (continued)

Description	PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
Stop Queue Manager	No equivalent	No equivalent	<a href="#">endmqm</a>	No equivalent	Yes

**Related concepts**

[Ways of administering IBM MQ queue managers and associated resources](#)

**Related tasks**

[Creating and managing queue managers on Multiplatforms](#)

**Related reference**

“REST API and PCF equivalents for queue managers” on page 2379

For most REST API optional query parameters and attributes for queue managers, an equivalent PCF parameter or attribute exists. Use the tables that are provided to understand these equivalents.

**ALW Command server commands**

A table of command server commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

Table 2. Commands for command server administration

Description	PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
Display command server	<a href="#">Inquire Queue Manager Status</a>	<a href="#">DISPLAY QMSTATUS</a>	<a href="#">dspmqcsv</a>	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
Start command server	<a href="#">Change Queue Manager</a>	<a href="#">ALTER QMGR</a>	<a href="#">strmqcsv</a>	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
Stop command server	No equivalent	No equivalent	<a href="#">endmqcsv</a>	No equivalent	Yes

**Related concepts**

[Ways of administering IBM MQ queue managers and associated resources](#)

**Related tasks**

[Administering IBM MQ](#)

A table of authority commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

*Table 3. Commands for authority administration*

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Delete authority record</a>	<a href="#">DELETE AUTHREC</a>	<a href="#">setmqaut</a>	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on <a href="#">page 2159</a> to run MQSC command	Yes
<a href="#">Inquire authority records</a>	<a href="#">DISPLAY AUTHREC</a>	<a href="#">dmpmqaut</a>	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on <a href="#">page 2159</a> to run MQSC command	Yes
<a href="#">Inquire entity authority</a>	<a href="#">DISPLAY ENTAUTH</a>	<a href="#">dspmqaut</a>	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on <a href="#">page 2159</a> to run MQSC command	Yes
<a href="#">Refresh Security</a>	<a href="#">REFRESH SECURITY</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on <a href="#">page 2159</a> to run MQSC command	Yes
<a href="#">Set authority record</a>	<a href="#">SET AUTHREC</a>	<a href="#">setmqaut</a>	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on <a href="#">page 2159</a> to run MQSC command	Yes

**Related concepts**

[Ways of administering IBM MQ queue managers and associated resources](#)

**Related tasks**

[Administering IBM MQ](#)



A table of cluster commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

*Table 4. Cluster commands*

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Inquire Cluster Queue Manager</a>	<a href="#">DISPLAY CLUSQMGR</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Refresh Cluster</a>	<a href="#">REFRESH CLUSTER</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Reset Cluster</a>	<a href="#">RESET CLUSTER</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	No
<a href="#">Resume Queue Manager Cluster</a>	<a href="#">RESUME QMGR</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Suspend Queue Manager Cluster</a>	<a href="#">SUSPEND QMGR</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes

### Related concepts

[Ways of administering IBM MQ queue managers and associated resources](#)

### Related tasks

[Administering IBM MQ](#)

## Authentication information commands

This table of authentication information commands shows equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

*Table 5. Authentication information commands*

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Change Authentication Information Object</a>	<a href="#">ALTER AUTHINFO</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Copy Authentication Information Object</a>	<a href="#">DEFINE AUTHINFO(x) LIKE(y)</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Create Authentication Information Object</a>	<a href="#">DEFINE AUTHINFO</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Delete Authentication Information Object</a>	<a href="#">DELETE AUTHINFO</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Authentication Information Object</a>	<a href="#">DISPLAY AUTHINFO</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes

### Related concepts

[Ways of administering IBM MQ queue managers and associated resources](#)

### Related tasks

[Administering IBM MQ](#)

A table of channel commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

*Table 6. Channel commands*

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Change Channel</a>	<a href="#">ALTER CHANNEL</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Copy Channel</a>	<a href="#">DEFINE CHANNEL(x) LIKE(y)</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Create Channel</a>	<a href="#">DEFINE CHANNEL</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Delete Channel</a>	<a href="#">DELETE CHANNEL</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Channel</a>	<a href="#">DISPLAY CHANNEL</a>	No equivalent	GET /admin/qmgr/{qmgrName}/channel	Yes
<a href="#">Inquire Channel Names</a>	<a href="#">DISPLAY CHANNEL</a>	No equivalent	GET /admin/qmgr/{qmgrName}/channel	Yes
<a href="#">Inquire Channel Status</a>	<a href="#">DISPLAY CHSTATUS</a>	No equivalent	GET /admin/qmgr/{qmgrName}/channel	Yes
<a href="#">Ping Channel</a>	<a href="#">PING CHANNEL</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Purge Channel</a>	<a href="#">PURGE CHANNEL</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Reset Channel</a>	<a href="#">RESET CHANNEL</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Resolve Channel</a>	<a href="#">RESOLVE CHANNEL</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Start Channel</a>	<a href="#">START CHANNEL</a>	<a href="#">runmqchl</a>	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Start Channel Initiator</a>	<a href="#">START CHINIT</a>	<a href="#">runmqchi</a>	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	No
<a href="#">Stop Channel</a>	<a href="#">STOP CHANNEL</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes

### Related concepts

[Ways of administering IBM MQ queue managers and associated resources](#)

### Related tasks

[Administering IBM MQ](#)

## Listener commands

A table of listener commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Change Listener</a>	<a href="#">ALTER LISTENER</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes

Table 7. Listener commands (continued)

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Copy Listener</a>	<a href="#">DEFINE LISTENER(x) LIKE(y)</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Create Listener</a>	<a href="#">DEFINE LISTENER</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Delete Listener</a>	<a href="#">DELETE LISTENER</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Listener</a>	<a href="#">DISPLAY LISTENER</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Listener Status</a>	<a href="#">DISPLAY LSSTATUS</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Start Channel Listener</a>	<a href="#">START LISTENER “1”</a> <a href="#">on page 13</a>	<a href="#">runmqlsr</a>	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Stop Listener</a>	<a href="#">STOP LISTENER</a>	<a href="#">endmqlsr “2”</a> on page 13	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>Used with listener objects only</li> <li>Stops all active listeners</li> </ol>				

**Related concepts**

[Ways of administering IBM MQ queue managers and associated resources](#)

**Related tasks**

[Administering IBM MQ](#)

A table of namelist commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

*Table 8. Namelist commands*

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Change Namelist</a>	<a href="#">ALTER NAMELIST</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Copy Namelist</a>	<a href="#">DEFINE NAMELIST(x)</a> <a href="#">LIKE(y)</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Create Namelist</a>	<a href="#">DEFINE NAMELIST</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Delete Namelist</a>	<a href="#">DELETE NAMELIST</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Namelist</a>	<a href="#">DISPLAY NAMELIST</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Namelist Names</a>	<a href="#">DISPLAY NAMELIST</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes

### Related concepts

[Ways of administering IBM MQ queue managers and associated resources](#)

### Related tasks

[Administering IBM MQ](#)

A table of process commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

*Table 9. Process commands*

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Change Process</a>	<a href="#">ALTER PROCESS</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Copy Process</a>	<a href="#">DEFINE PROCESS(x) LIKE(y)</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Create Process</a>	<a href="#">DEFINE PROCESS</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Delete Process</a>	<a href="#">DELETE PROCESS</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Process</a>	<a href="#">DISPLAY PROCESS</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Process Names</a>	<a href="#">DISPLAY PROCESS</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes

### Related concepts

[Ways of administering IBM MQ queue managers and associated resources](#)

### Related tasks

[Administering IBM MQ](#)

A table of queue commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

*Table 10. Queue commands*

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Change Queue</a>	<a href="#">ALTER QLOCAL</a> <a href="#">ALTER QALIAS</a> <a href="#">ALTER QMODEL</a> <a href="#">ALTER QREMOTE</a>	No equivalent	<a href="#">PATCH /admin/qmgr/{qmgrName}/queue</a>	Yes
<a href="#">Clear Queue</a>	<a href="#">CLEAR QLOCAL</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Copy Queue</a>	<a href="#">DEFINE QLOCAL(x) LIKE(y)</a> <a href="#">DEFINE QALIAS(x) LIKE(y)</a> <a href="#">DEFINE QMODEL(x) LIKE(y)</a> <a href="#">DEFINE QREMOTE(x) LIKE(y)</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Create Queue</a>	<a href="#">DEFINE QLOCAL</a> <a href="#">DEFINE QALIAS</a> <a href="#">DEFINE QMODEL</a> <a href="#">DEFINE QREMOTE</a>	No equivalent	<a href="#">POST /admin/qmgr/{qmgrName}/queue</a>	Yes
<a href="#">Delete Queue</a>	<a href="#">DELETE QLOCAL</a> <a href="#">DELETE QALIAS</a> <a href="#">DELETE QMODEL</a> <a href="#">DELETE QREMOTE</a>	No equivalent	<a href="#">DELETE /admin/qmgr/{qmgrName}/queue</a>	Yes
<a href="#">Inquire Queue</a>	<a href="#">DISPLAY QUEUE</a>	No equivalent	<a href="#">GET /admin/qmgr/{qmgrName}/queue</a>	Yes
<a href="#">Inquire Queue Names</a>	<a href="#">DISPLAY QUEUE</a>	No equivalent	Use “ <a href="#">/admin/action/qmgr/{qmgrName}/mqsc</a> ” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Queue Status</a>	<a href="#">DISPLAY QSTATUS</a>	No equivalent	<a href="#">GET /admin/qmgr/{qmgrName}/queue</a>	Yes



Table 10. Queue commands (continued)

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Reset Queue Statistics</a>	No equivalent	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	No

**Related concepts**

Ways of administering IBM MQ queue managers and associated resources

**Related tasks**

[Administering IBM MQ](#)

**ALW Service commands**

A table of service commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

Table 11. Service commands

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Change Service</a>	<a href="#">ALTER SERVICE</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Copy Service</a>	<a href="#">DEFINE SERVICE(x) LIKE(y)</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Create Service</a>	<a href="#">DEFINE SERVICE</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Delete Service</a>	<a href="#">DELETE SERVICE</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Inquire Service</a>	<a href="#">DISPLAY SERVICE</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes

PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
<a href="#">Inquire Service Status</a>	<a href="#">DISPLAY SVSTATUS</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Start Service</a>	<a href="#">START SERVICE</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes
<a href="#">Stop Service</a>	<a href="#">STOP SERVICE</a>	No equivalent	Use “/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 to run MQSC command	Yes

### Related concepts

[Ways of administering IBM MQ queue managers and associated resources](#)

### Related tasks

[Administering IBM MQ](#)

## Other commands

A table of other commands, showing the command description, and the equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

Description	PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
Create conversion exit	No equivalent	No equivalent	<a href="#">crtmqcvx</a>	No equivalent	No
Display files used by objects	No equivalent	No equivalent	<a href="#">dspmqfls</a>	No equivalent	No
Display formatted trace	No equivalent	No equivalent	<a href="#">dspmqtrc</a> <sup>“1” on page 19</sup>	No equivalent	No
Display version information	No equivalent	No equivalent	<a href="#">dspmqver</a>	No equivalent	No
Display transactions	No equivalent	No equivalent	<a href="#">dspmqtrn</a>	No equivalent	No
Dump log	No equivalent	No equivalent	<a href="#">dmpmqlog</a>	No equivalent	No
Dump MQ Configuration	No equivalent	No equivalent	<a href="#">dmpmqcfg</a>	No equivalent	No
End trace	No equivalent	No equivalent	<a href="#">endmqtrc</a>	No equivalent	Yes

Table 12. Other commands (continued)

Description	PCF command	MQSC command	Control command	REST API resource and HTTP method	IBM MQ Explorer equivalent?
Escape	Escape	No equivalent	No equivalent	POST /admin/action/qmgr/{qmgrName}/mqsc	No
Record media image	No equivalent	No equivalent	<a href="#">rcdmqimg</a>	No equivalent	No
Re-create media object	No equivalent	No equivalent	<a href="#">rcrmqobj</a>	No equivalent	No
Resolve transactions	No equivalent	No equivalent	<a href="#">rsvmqtrn</a>	No equivalent	No
Run client trigger monitor	No equivalent	No equivalent	<a href="#">runmqtmc</a>	No equivalent	No
Run dead-letter queue handler	No equivalent	No equivalent	<a href="#">runmqdlq</a>	No equivalent	No
Run MQSC commands	No equivalent	No equivalent	<a href="#">runmqsc</a>	No equivalent	No
Run trigger monitor	No equivalent	No equivalent	<a href="#">runmqtrm</a>	No equivalent	No
Set service connection points	No equivalent	No equivalent	<a href="#">setmqscp</a> <sup>"2" on page 19</sup>	No equivalent	No
Start IBM MQ trace	No equivalent	No equivalent	<a href="#">strmqtrc</a>	No equivalent	Yes
IBM MQ Services control	No equivalent	No equivalent	<a href="#">amqmdain</a> <sup>"2" on page 19</sup>	No equivalent	No
<b>Notes:</b>					
1. Not supported on IBM MQ for Windows.					
2. Supported by IBM MQ for Windows only.					

**Related concepts**

[Ways of administering IBM MQ queue managers and associated resources](#)

**Related tasks**

[Administering IBM MQ](#)

**How to read syntax diagrams**

The syntax for a command and its options is presented in the form of a syntax diagram, also known as a railroad diagram. A syntax diagram is a visual format suitable for sighted users. It tells you what options you can supply with the command, and how to enter them. It indicates relationships between different options, and sometimes different values for an option.

Each syntax diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a syntax diagram from left to right and from top to bottom, following the direction of the arrows.

Other conventions used in syntax diagrams are shown in [Table 13 on page 20](#).

<i>Table 13. How to read syntax diagrams</i>	
<b>Convention</b>	<b>Meaning</b>
	You must specify values A, B, and C. Required values are shown on the main line of a syntax diagram.
	You may specify value A. Optional values are shown below the main line of a syntax diagram.
	Values A, B, and C are alternatives, one of which you must specify.
	Values A, B, and C are alternatives, one of which you might specify.
	This shows that a value (for example, A, or B, or C) must be selected, and if another is to be selected then a comma must be used between the values.
	You might specify value A multiple times. The separator in this example is optional.
	Values A, B, and C are alternatives, one of which you might specify. If you specify none of the values shown, the default A (the value shown above the main line) is used.
	The syntax fragment Name is shown separately from the main syntax diagram.
Punctuation and uppercase values	Specify exactly as shown.

## IBM MQ control commands reference

Reference information about the IBM MQ control commands.

For information about running these commands on Multiplatforms, see [Administering IBM MQ for Multiplatforms using control commands](#).

### **Multi** **addmqinf (add configuration information)**

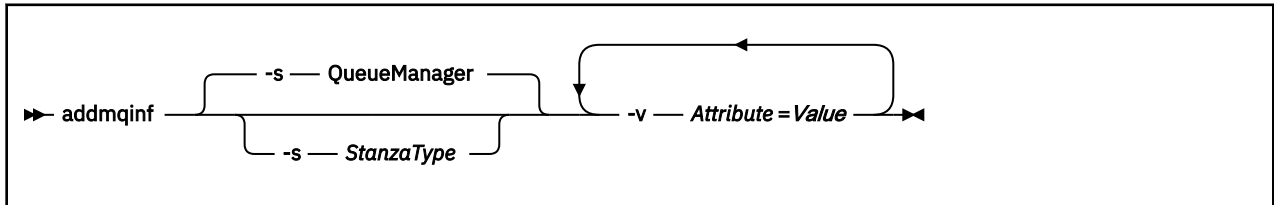
Add IBM MQ configuration information on AIX, Linux, and Windows only.

#### Purpose

Use the **addmqinf** command to add information to the IBM MQ configuration data.

For example, use **dspmqinf** to display and **addmqinf** to copy configuration data from the system where a queue manager was created, to other systems where the same multi-instance queue manager is also to be started.

#### Syntax



#### Required parameters

##### **-v Attribute = Value**

The name and value of the stanza attributes to be placed in the stanza specified in the command.

Table 14 on page 21 lists the QueueManager stanza attribute values. The queue manager stanza is the only stanza that is currently supported.

Attribute	Value	Required or optional
<b>Name</b>	The name of the queue manager. You must provide a different name from any other queue manager stanza on the system.	Required
<b>Prefix</b>	The directory path under which this queue manager data directory is stored by default. You can use <b>Prefix</b> to modify the location of the queue manager data directories. The value of <b>Directory</b> is automatically appended to this path.	Required
<b>Directory</b>	The name of the queue manager data directory. Sometimes the name must be provided (as in “Example” on page 22 ), because it is different from the queue manager name. Copy the directory name from the value returned by <b>dspmqinf</b> . The rules for transforming queue manager names into directory names are described in <a href="#">Understanding IBM MQ file names</a> .	Required

Table 14. QueueManager stanza attributes (continued)		
Attribute	Value	Required or optional
<b>DataPath</b>	<p>The directory path where the queue manager data files are placed. The value of <b>Directory</b> is not automatically appended to this path and you must provide the transformed queue manager name as part of <b>DataPath</b>.</p> <p>Linux AIX If the <b>DataPath</b> attribute is omitted on AIX and Linux, the queue manager data directory path is defined as <b>Prefix / Directory</b>.</p>	<p>Linux AIX</p> <p>On AIX and Linux: Optional</p> <p>Windows</p> <p>On Windows: Required</p>
<b>EphemeralPrefix</b>	<p>Specifies the path to the directory, within which the queue manager ephemeral data is kept, such as IPC sockets.</p> <p>If the <b>EphemeralPrefix</b> attribute is omitted, the queue manager ephemeral prefix is defined as <b>Prefix</b>.</p>	Optional

## Optional parameters

### -s *StanzaType*

A stanza of the type *StanzaType* is added to the IBM MQ configuration.

The default value of *StanzaType* is `QueueManager`.

The only supported value of *StanzaType* is `QueueManager`.

## Return codes

Table 15. Return code identifiers and descriptions

Return code	Description
0	Successful operation
1	Queue manager location is invalid (either <b>Prefix</b> or <b>DataPath</b> )
39	Bad command-line parameters
45	Stanza already exists
46	Required configuration attribute is missing
58	Inconsistent use of installations detected
69	Storage is not available
71	Unexpected error
72	Queue manager name error
100	Log location is invalid

### Example

```
addmqinf -v DataPath=/MQHA/qmgrs/QM!NAME +
-v Prefix=/var/mqm +
-v Directory=QM!NAME +
-v Name=QM.NAME
```

Creates the following stanza in `mqs.ini`:

```
QueueManager:  
Name=QM.NAME  
Prefix=/var/mqm  
Directory=QM!NAME  
DataPath=/MQHA/qmgrs/QM!NAME
```

## Usage notes

Use **dspmqinf** with **addmqinf** to create an instance of a multi-instance queue manager on a different server.

To use this command you must be an IBM MQ administrator and a member of the mqm group.

## Related commands

Table 16. Related commands and their descriptions

Command	Description
<a href="#">“dspmqinf (display configuration information)”</a> on page 86	Display IBM MQ configuration information
<a href="#">“rmvmqinf (remove configuration information)”</a> on page 148	Remove IBM MQ configuration information

## Windows amqmdain (services control)

**amqmdain** is used to configure or control some Windows specific administrative tasks.

### Purpose

The **amqmdain** command applies to IBM MQ for Windows only.

You can use **amqmdain** to perform some Windows specific administrative tasks.

Starting a queue manager with **amqmdain** is equivalent to using the **strmqm** command with the option **-ss**. **amqmdain** makes the queue manager run in a non-interactive session under a different user account. However, to ensure that all queue manager startup feedback is returned to the command line, use the **strmqm -ss** command rather than **amqmdain**.

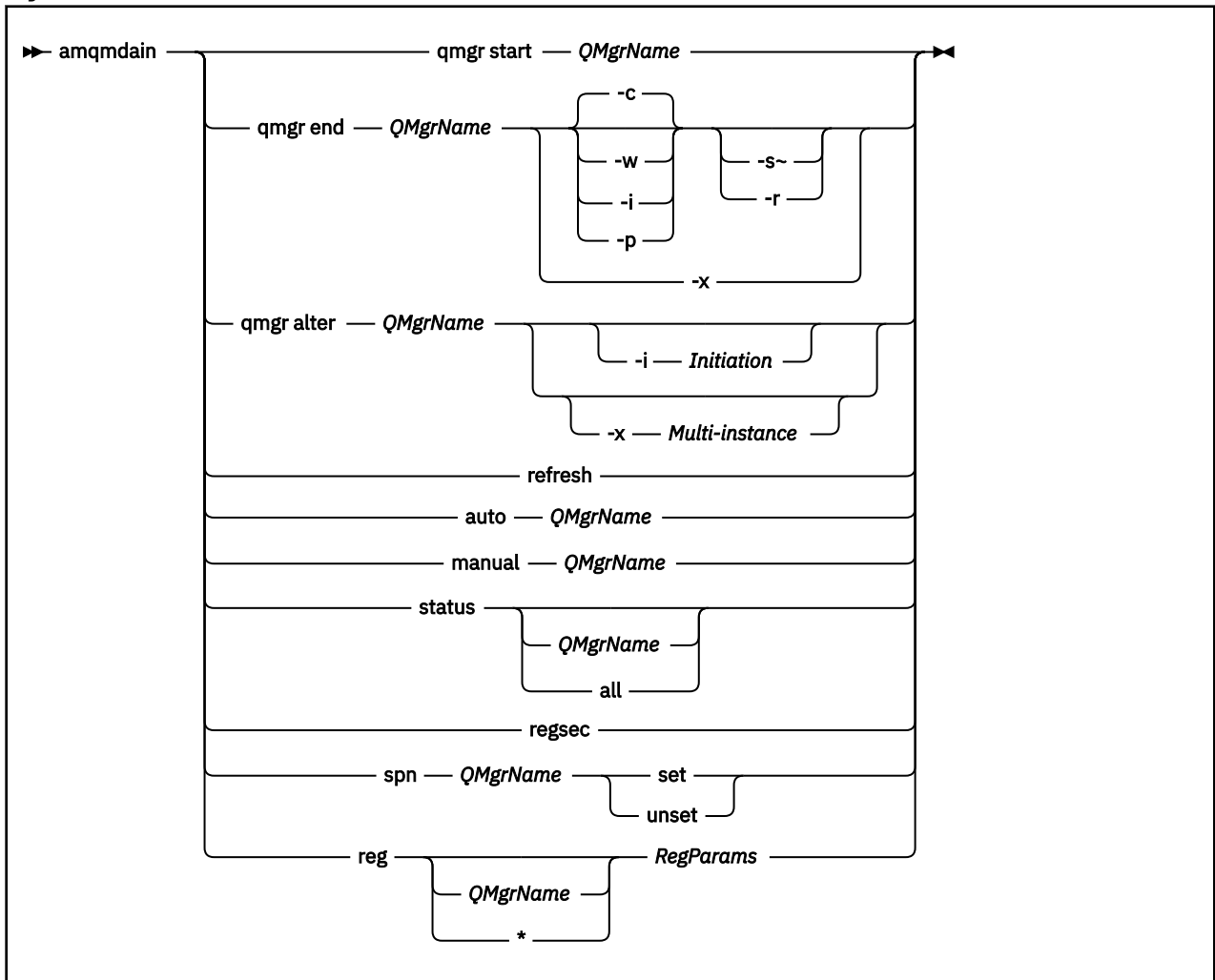
You must use the **amqmdain** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the **dspmq** command as follows:

```
dspmq -o installation
```

To administer and define IBM MQ service and listener objects, use MQSC commands, PCF commands, or the IBM MQ Explorer.

The **amqmdain** command has been updated to modify either the `.ini` files or the registry as appropriate.

## Syntax



### Keywords and parameters

All parameters are required unless the description states they are optional.

In every case, *QMgrName* is the name of the queue manager to which the command applies.

#### **qmgr start *QMgrName***

Starts a queue manager.

This parameter can also be written in the form *start QMgrName*.

If you start your queue manager as a service and need the queue manager to continue to run after logoff, use `strmqm -ss qmgr` instead of `amqmdain start qmgr`.

#### **qmgr end *QMgrName***

Ends a queue manager.

This parameter can also be written in the form **end *QMgrName***.

For consistency across platforms, use `endmqm qmgr` instead of `amqmdain end qmgr`.

For fuller descriptions of the options, see [“endmqm \(end queue manager\)”](#) on page 121.

#### **-c**

Controlled (or quiesced) shutdown.

#### **-w**

Wait shutdown.



- i**  
Immediate shut down.
- p**  
Pre-emptive shut down.
- r**  
Reconnect clients.
- s**  
Switch over to a standby queue manager instance.
- x**  
End the standby instance of the queue manager without ending the active instance.

**qmgr alter QMgrName**

Alters a queue manager.

**-i Initiation**

Specifies the initiation type. Possible values are:

<i>Table 17. Initiation command parameters.</i>	
<b>Value</b>	<b>Description</b>
auto	Sets the queue manager to automatic startup (when the machine starts, or more precisely when the IBM MQ service starts). The syntax is:  <pre>amqmdain qmgr alter QmgrName -i auto</pre>
interactive	Sets the queue manager to manual startup that then runs under the logged on (interactive) user. The syntax is:  <pre>amqmdain qmgr alter QmgrName -i interactive</pre>
service	Sets the queue manager to manual startup that then runs as a service. The syntax is:  <pre>amqmdain qmgr alter QmgrName -i service</pre>

**-x Multi-instance**

Specifies if auto queue manager start by the IBM MQ service permits multiple instances. Equivalent to the `-sax` option on the **crtmqm** command. Also specifies if the **amqmdain start qmgr** command permits standby instances. Possible values are:

<i>Table 18. Multi-instance command parameters.</i>	
<b>Value</b>	<b>Description</b>
set	Sets automatic queue manager startup to permit multiple instances. Issues <b>strmqm -x</b> . The set option is ignored for queue managers that are initiated interactively or as a manual service startup. The syntax of the command is:  <pre>amqmdain qmgr alter QmgrName -x set</pre>

Table 18. Multi-instance command parameters. (continued)

Value	Description
unset	Sets automatic queue manager startup to single instance. Issues <b>strmqm</b> . The unset option is ignored for queue managers that are initiated interactively or as a manual service startup. The syntax of the command is:  <pre>amqmdain qmgr alter QmgrName -x unset</pre>

**refresh**

Refreshes or checks the status of a queue manager. You will not see anything returned on the screen after executing this command.

**auto QMgrName**

Sets a queue manager to automatic startup.

**manual QMgrName**

Sets a queue manager to manual startup.

**status QMgrName| all**

These parameters are optional.

Table 19. Status command parameters.

Header	Header
If no parameter is supplied:	Displays the status of the IBM MQ services.
If a QMgrName is supplied:	Displays the status of the named queue manager.
If the parameter <i>all</i> is supplied:	Displays the status of the IBM MQ services and all queue managers.

**regsec**

Ensures that the security permissions assigned to the Registry keys containing installation information are correct.

**spn QMgrName set | unset**

You can set or unset the service principal name for a queue manager.

**reg QMgrName| \* RegParams**

Parameters QMgrName, and \* are optional.

Table 20. Reg command parameters.

Value	Description
If RegParams is specified alone:	Modifies queue manager configuration information related to the default queue manager.
If QMgrName and RegParams are specified:	Modifies queue manager configuration information related to the queue manager specified by QMgrName.
If * and RegParams are specified:	Modifies IBM MQ configuration information.

The parameter, *RegParams*, specifies the stanzas to change, and the changes that are to be made. *RegParams* takes one of the following forms:

- -c add -s stanza -v attribute= value
- -c remove -s stanza -v [attribute|\*]
- -c display -s stanza -v [attribute|\*]

If you are specifying queue manager configuration information, the valid values for *stanza* are:

```
XAResourceManager\name
ApiExitLocal\name
Channels
ExitPath
InstanceData
Log
QueueManagerStartup
TCP
LU62
SPX
NetBios
Connection
QMErrorLog
Broker

ExitPropertiesLocal
SSL
```

If you are modifying IBM MQ configuration information, the valid values for *stanza* are:

```
ApiExitCommon\name
ApiExitTemplate\name
ACPI
AllQueueManagers
Channels
DefaultQueueManager
LogDefaults
ExitProperties
```

Note the following usage considerations:

- **amqmdain** does not validate the values you specify for *name*, *attribute*, or *value*.
- When you specify add, and an attribute exists, it is modified.
- If a stanza does not exist, **amqmdain** creates it.
- When you specify remove, you can use the value \* to remove all attributes.
- When you specify display, you can use the value \* to display all attributes which have been defined. This value only displays the attributes which have been defined and not the complete list of valid attributes.
- If you use remove to delete the only attribute in a stanza, the stanza itself is deleted.
- Any modification you make to the Registry re-secures all IBM MQ Registry entries.

## Examples

The following example adds an XAResourceManager to queue manager TEST. The commands issued are:

```
amqmdain reg TEST -c add -s XAResourceManager\Sample -v SwitchFile=sf1
amqmdain reg TEST -c add -s XAResourceManager\Sample -v ThreadOfControl=THREAD
amqmdain reg TEST -c add -s XAResourceManager\Sample -v XAOpenString=openit
amqmdain reg TEST -c add -s XAResourceManager\Sample -v XACloseString=ccloseit
```

To display the values set by the commands, use:

```
amqmdain reg TEST -c display -s XAResourceManager\Sample -v *
```

The display would look something like the following:

```
0784726, 5639-B43 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Displaying registry value for Queue Manager 'TEST'
Attribute = Name, Value = Sample
Attribute = SwitchFile, Value = sf1
Attribute = ThreadOfControl, Value = THREAD
```

```
Attribute = XAOpenString, Value = openit
Attribute = XACloseString, Value = closeit
```

To remove the XAResourceManager from queue manager TEST, use:

```
amqmdain reg TEST -c remove -s XAResourceManager\Sample -v *
```

## Return codes

Table 21. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
-2	Syntax error
-3	Failed to initialize MFC
-6	Feature no longer supported
-7	Configuration failed
-9	Unexpected Registry error
-16	Failed to configure service principal name
-29	Inconsistent use of installations detected
62	The queue manager is associated with a different installation
71	Unexpected error
<b>Windows</b> 119	Permission denied (Windows only)

### Note:

1. If the **qmgr start QMgrName** command is issued, all return codes that can be returned with **strmqm**, can be returned here also. For a list of these return codes, see [“strmqm \(start queue manager\)”](#) on page 237.
2. If the **qmgr end QMgrName** command is issued, all return codes that can be returned with **endmqm**, can be returned here also. For a list of these return codes, see [“endmqm \(end queue manager\)”](#) on page 121.

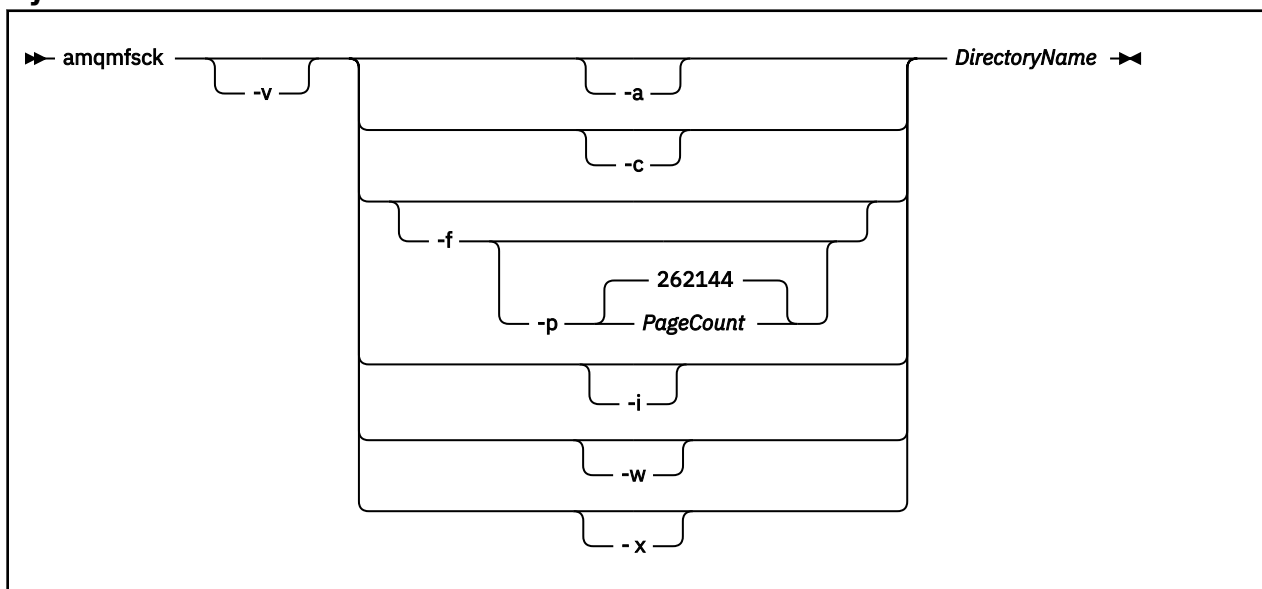
## IBM i Linux AIX **amqmfsc** (file system check)

**amqmfsc** checks whether a shared file system on IBM i, AIX and Linux systems meets the requirements for storing the queue manager data of a multi-instance queue manager.

### Purpose

The **amqmfsc** command applies only to IBM i, AIX and Linux systems. You do not need to check the network drive on Windows. **amqmfsc** tests that a file system correctly handles concurrent writes to a file and the waiting for and releasing of locks.

## Syntax



### Required parameters

#### DirectoryName

The name of the directory to check.

### Optional parameters

#### -a

Perform the second phase of the data integrity test.

Run this on two machines at the same time. You must have formatted the test file using the `-f` option previously

#### -c

Test writing to a file in the directory concurrently.

#### -f

Perform the first phase of the data integrity test.

Formats a file in the directory in preparation for data integrity testing.

#### -i

Perform the third phase of the data integrity test.

Checks the integrity of the file after the failure to discover whether the test worked.

#### -p

Specifies the size of the test file used in the data integrity test in pages. .

The size is rounded up to the nearest multiple of 16 pages. The file is formatted with *PageCount* pages of 4 KB.

The optimum size of the file depends on the speed of the filesystem and the nature of the test you perform. If this parameter is omitted, the test file is 262144 pages, or 1 GB.

The size is automatically reduced so that the formatting completes in about 60 seconds even on a very slow filesystem.

#### -v

Verbose output.

#### -w

Test waiting for and releasing locks.

**-x**

Deletes any files created by **amqmfscck** during the testing of the directory.

Do not use this option until you have completed the testing, or if you need to change the number of pages used in the integrity test.

## Usage

You must be an IBM MQ Administrator to run the command. You must have read/write access to the directory being checked.

**IBM i** On IBM i, use QSH to run the program. There is no CL command.

The command returns an exit code of zero if the tests complete successfully.

The task, [Verifying shared file system behavior](#), describes how to use **amqmfscck** to check the whether of a file system is suitable for multi-instance queue managers.

## Interpreting your results

If the check fails, the file system is not capable of being used by IBM MQ queue managers. If the tests fail, choose verbose mode to help you to interpret the errors. The output from the verbose option helps you understand why the command failed, and if the problem can be solved by reconfiguring the file system.

Sometimes the failure might be an access control problem that can be fixed by changing directory ownership or permissions. Sometimes the failure can be fixed by reconfiguring the file system to behave in a different way. For example, some file systems have performance options that might need to be changed. It is also possible that the file system protocol does not support concurrency sufficiently robustly, and you must use a different file system. For example, you must use NFSv4 rather than NFSv3.

If the check succeeds, the command reports `The tests on the directory completed successfully`. If your environment is not listed as supported in the [Testing statement for IBM MQ multi-instance queue manager file systems](#), this result does not necessarily mean that you can run IBM MQ multi-instance queue managers successfully.

You must plan and run a variety of tests to satisfy yourself that you have covered all foreseeable circumstances. Some failures are intermittent, and there is a better chance of discovering them if you run the tests more than once.

### Related tasks

[Verifying shared file system behavior](#)

## **Multi** **crtmqcvx (create data conversion code)**

Create data conversion code from data type structures.

## Purpose

Use the **crtmqcvx** command to create a fragment of code that performs data conversion on data type structures. The command generates a C function that can be used in an exit to convert C structures.

The command reads an input file containing structures to be converted, and writes an output file containing code fragments to convert those structures.

For information about using this command, see [Utility for creating conversion-exit code](#).

## Syntax

```
►► crtmqcvx — SourceFile — TargetFile ◄◄
```

## Required parameters

### SourceFile

The input file containing the C structures to convert.

### TargetFile

The output file containing the code fragments generated to convert the structures.

## Return codes

Table 22. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

## Examples

The following example shows the results of using the data conversion command against a source C structure. The command issued is:

```
crtmqcvx source.tmp target.c
```

The input file, `source.tmp`, looks like this:

```
/* This is a test C structure which can be converted by the */  
/* crtmqcvx utility                                         */  
  
struct my_structure  
{  
    int    code;  
    MQLONG value;  
};
```

The output file, `target.c`, produced by the command, looks like this:

```

MQLONG Convertmy_structure(
    PMQDXP pExitParms,
    PMQBYTE *in_cursor,
    PMQBYTE *out_cursor,
    PMQBYTE in_lastbyte,
    PMQBYTE out_lastbyte,
    MQHCONN hConn,
    MQLONG opts,
    MQLONG MsgEncoding,
    MQLONG ReqEncoding,
    MQLONG MsgCCSID,
    MQLONG ReqCCSID,
    MQLONG CompCode,
    MQLONG Reason)
{
    MQLONG ReturnCode = MQRC_NONE;

    ConvertLong(1); /* code */

    AlignLong();
    ConvertLong(1); /* value */

Fail:
    return(ReturnCode);
}

```

You can use these code fragments in your applications to convert data structures. However, if you do so, the fragment uses macros supplied in the header file `amqsvmha.h`.

## Multi **crtmqdir (create IBM MQ directories)**

Create, check, and correct IBM MQ directories and files.

### Purpose

Use the **crtmqdir** command to check that the necessary directories and files used by IBM MQ exist and have the appropriate ownership and permissions. The command can optionally create any missing directories or files, and correct any inconsistent ownership or permissions.



**Attention:** The scope of this command is `MQ_DATA_PATH` which, for example, is `/var/mqm` on Linux. This command does not affect `MQ_INSTALLATION_PATH`, which is `/opt/mqm` on Linux.

The system-wide directories and files are created as part of the IBM MQ installation procedure. The tool can subsequently be run to check or ensure that the necessary IBM MQ directories and files continue to have appropriate ownership and permissions.

### Important:

1. You must have sufficient permission to determine whether the configuration is correct and, optionally, correct that configuration.
2. When you use the **-a** parameter, no queue managers can be running.
3. When you use the **-m** parameter, the queue manager you specified must be stopped.
4. You must not create, delete, or start any queue managers while **crtmqdir** is running.

Linux

AIX

On AIX and Linux, this typically means that you are the `mqm` user. This is necessary when using the **-a** or **-m** parameters, together with the **-f** parameter.

Depending on the configuration, the **crtmqdir** command might require you to be an operating system administrator, or superuser.

Note:

Linux

AIX

The security of `data path/log/qm`, on AIX and Linux, is set to 2770.

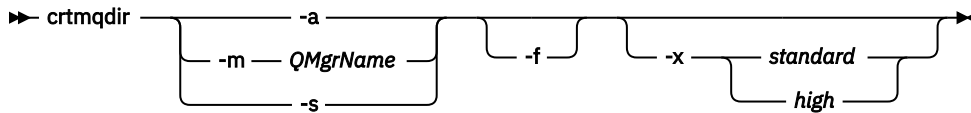
Windows

On Windows, this typically means that you are a member of the IBM MQ administration group. This is necessary when using the **-a** or **-m** parameters.



**IBM i** On IBM i, you must run the command as a member of the IBM MQ administrative group. This is necessary when using the **-a** or **-m** parameters, together with the **-f** parameter.

## Syntax



## Required parameters

Specify one of the following parameters only:

**-a**

Check all directories; that is, system-wide directories and all the queue managers.



**Attention:** The queue manager must be associated with the current installation.

**-m**

Check directories for the specified queue manager name.



**Attention:** The queue manager must be associated with the current installation.

**-s**

Check the system-wide directories; that is, directories that are not queue manager specific.

## Optional parameters

**-f**

This option causes directories or files to be created if they are missing, and on AIX and Linux only, ownership or permissions to be corrected if they are inappropriately set.

If **-a** or **-m** is specified on AIX and Linux, as a minimum, the program attempts to correct ownership or permissions on files that were created at the time of queue manager creation.

### **-x level of permissions**

Specify one of the following values only:

#### **standard**

By default, directories and files get a standard set of permissions, but a high level of permissions can be requested.

#### **high**

This option applies to the following platforms:

- **AIX** AIX
- **Linux** Linux

It ensures that files in the following directories can be deleted only by the owner:

- errors
- trace
- webui

## Return codes

Table 23. Return code identifiers and descriptions

Return code	Description
0	Successful completion
10	A warning occurred
20	An error occurred

## Examples

- The following command checks and fixes the system-wide directories:

```
crtmqdir -s -f
```

- The following command checks (but does not fix) queue manager QM1:

```
crtmqdir -m Qm1
```

### Multi

## crtmqenv (create IBM MQ environment)

Create a list of environment variables for an installation of IBM MQ, on AIX, Linux, and Windows.

## Purpose

You can use the **crtmqenv** command to create a list of environment variables with the appropriate values for an installation of IBM MQ. The list of environment variables is displayed on the command line, and any variables that exist on the system have the IBM MQ values added to them. This command does not set the environment variables for you, but gives you the appropriate strings to set the variables yourself, for example, within your own scripts.

If you want the environment variables set for you in a shell environment, you can use the [setmqenv](#) command instead of using the **crtmqenv** command.

You can specify which installation the environment is created for by specifying a queue manager name, an installation name, or an installation path. You can also create the environment for the installation that issues the **crtmqenv** command by issuing the command with the **-s** parameter.

This command lists the following environment variables, and their values, appropriate to your system:

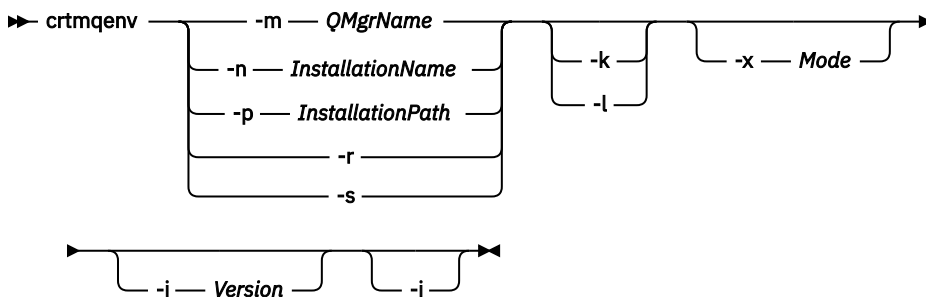
- CLASSPATH
- INCLUDE
- LIB
- MANPATH
- MQ\_DATA\_PATH
- MQ\_ENV\_MODE
- MQ\_FILE\_PATH
- MQ\_INSTALLATION\_NAME
- MQ\_INSTALLATION\_PATH
- MQ\_JAVA\_INSTALL\_PATH
- MQ\_JAVA\_DATA\_PATH
- MQ\_JAVA\_LIB\_PATH
- MQ\_JAVA\_JVM\_FLAG
- MQ\_JRE\_PATH

- PATH

## Usage notes

The **crtmqenv** command removes all directories for all IBM MQ installations from the environment variables before adding new references to the installation for which you are setting up the environment. Therefore, if you want to set any additional environment variables that reference IBM MQ, set the variables after issuing the **crtmqenv** command. For example, if you want to add `MQ_INSTALLATION_PATH/java/lib` to `LD_LIBRARY_PATH`, you must do so after running **crtmqenv**.

## Syntax



## Required Parameters

### -m *QMgrName*

Create the environment for the installation associated with the queue manager *QMgrName*.

### -n *InstallationName*

Create the environment for the installation named *InstallationName*.

### -p *InstallationPath*

Create the environment for the installation in the path *InstallationPath*.

### -r

Remove all installations from the environment.

### -s

Create the environment for the installation that issued the command.

## Optional Parameters

### Linux AIX -k

Applies to AIX and Linux only. If the **-k** flag is specified:

- **AIX** On AIX, the `LIBPATH` environment variable is set.
- **Linux** On Linux, the `LD_LIBRARY_PATH` environment variable is set.

Include the `LD_LIBRARY_PATH`, or `LIBPATH`, environment variable in the environment, adding the path to the IBM MQ libraries at the start of the current `LD_LIBRARY_PATH`, or `LIBPATH`, variable.

### Linux AIX -l

Applies to AIX and Linux only. If the **-l** flag is specified:

- **AIX** On AIX, the `LIBPATH` environment variable is set.
- **Linux** On Linux, the `LD_LIBRARY_PATH` environment variable is set.

Include the `LD_LIBRARY_PATH`, or `LIBPATH`, environment variable in the environment, adding the path to the IBM MQ libraries at the end of the current `LD_LIBRARY_PATH`, or `LIBPATH`, variable.

### **-x Mode**

Mode can take the value 32, or 64.

Create a 32-bit or 64-bit environment:

- If you specify `-x 32`, the PATH environment variable is changed to add a prefix to the binary path for 32 bit executables.
- If you specify `-x 64`, the PATH environment variable is changed to add a prefix to the binary path for 64 bit executables.

If this parameter is not specified, the environment matches that of the queue manager or installation specified in the command.

Any attempt to display a 64-bit environment with a 32-bit installation fails.

### **-j Version**

Version can take the value 2.0, or 3.0.

- **JMS 2.0** If you specify `-j 2.0` the CLASSPATH environment variable is changed to include the JAR files necessary to run JMS 2.0 applications. This is the default if `-j` is not specified.
- **JM 3.0** If you specify `-j 3.0` the CLASSPATH environment variable is changed to include the JAR files necessary to run Jakarta Messaging 3.0 applications.

IBM MQ 9.3.0 introduces support for [Jakarta Messaging 3.0](#). JMS 2.0 is still fully supported.

### **-i**

List only the additions to the environment.

When this parameter is specified, the environment variables set for previous installations remain in the environment variable path and must be manually removed.

## **Return codes**

Table 24. Return code identifiers and descriptions

Return code	Description
0	Command completed normally.
10	Command completed with unexpected results.
20	An error occurred during processing.

### **Examples**

The following examples assume that a copy of IBM MQ is installed in `/opt/mqm` on a Linux or AIX system.

1. This command creates a list of environment variables for an installation installed in `/opt/mqm`:

```
/opt/mqm/bin/crtmqenv -s
```

2. This command creates a list of environment variables for an installation installed in `/opt/mqm2`, and includes the path to the installation at the end of the current value of the `LD_LIBRARY_PATH` variable:

```
/opt/mqm/bin/crtmqenv -p /opt/mqm2 -l
```

3. This command creates a list of environment variables for the queue manager QM1, in a 32-bit environment:

```
/opt/mqm/bin/crtmqenv -m QM1 -x 32
```

The following example assumes that a copy of IBM MQ is installed in `C:\Program Files\IBM\MQ` on a Windows system.

1. This command creates a list of environment variables for an installation called `installation1`:

```
"C:\Program Files\IBM\MQ\crtmqenv" -n installation1
```

### Related concepts

[Multiple installations](#)

### Related tasks

[Choosing a primary installation](#)

### Related reference

[“setmqenv \(set IBM MQ environment\)” on page 206](#)

Use the **setmqenv** command to set up the IBM MQ environment on AIX, Linux, and Windows.

Linux

AIX

## crtmqinst (create IBM MQ installation)

Create installation entries in `mqinst.ini` on AIX and Linux systems.

### Purpose

File `mqinst.ini` contains information about all IBM MQ installations on a system. For more information about `mqinst.ini`, see [Installation configuration file, mqinst.ini](#).

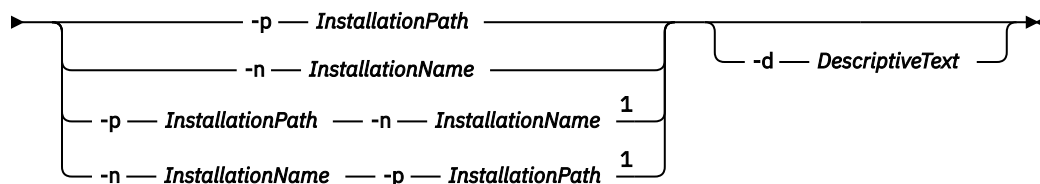


**Attention:** Only the user `root` can run this command.

The first IBM MQ installation is automatically given an installation name of `Installation1` because the **crtmqinst** command is not available until an installation of IBM MQ is on the system. Subsequent installations can have an installation name set before installation occurs, by using the **crtmqinst** command. The installation name cannot be changed after installation. For more information about installation names, see [Choosing an installation name](#).

### Syntax

►► **crtmqinst** ►►



Notes:

<sup>1</sup> When specified together, the installation name and installation path must refer to the same installation.

### Parameters

#### -d

Text that describes the installation.

The text can be up to 64 single-byte characters, or 32 double-byte characters. The default value is all blanks. You must use quotation marks around the text if it contains spaces.

#### -n *InstallationName*

The name of the installation.

The name can contain up to 16 single-byte characters and must be a combination of alphabetic and numeric characters in the ranges a-z, A-Z, and 0-9. The installation name must be unique,

regardless of whether uppercase or lowercase characters are used. For example, the names `INSTALLATIONNAME` and `InstallationName` are not unique.

If you do not supply the installation name, the next available name in the series `Installation1`, `Installation2...` is used.

#### **-p *InstallationPath***

The installation path. If you do not supply the installation path, `/opt/mqm` is used on AIX and Linux systems, and `/usr/mqm` is used on AIX.

## Return codes

Table 25. Return code identifiers and descriptions

Return code	Description
0	Entry created without error
10	Invalid installation level
36	Invalid arguments supplied
37	Descriptive text was in error
45	Entry already exists
59	Invalid installation specified
71	Unexpected error
89	.ini file error
96	Could not lock .ini file
98	Insufficient authority to access .ini file
131	Resource problem

### Example

1. This command creates an entry with an installation name of `myInstallation`, an installation path of `/opt/myInstallation`, and a description "My IBM MQ installation":

```
crtmqinst -n MyInstallation -p /opt/myInstallation -d "My IBM MQ installation"
```

Quotation marks are needed because the descriptive text contains spaces.

**Note:** On AIX and Linux, the **crtmqinst** command must be run by the root user because full access permissions are required to write to the `mqinst.ini` configuration file.



## **crtmqm (create queue manager)**

Create a queue manager.

### Purpose

Use the **crtmqm** command to create a queue manager and define the default and system objects. The objects created by the **crtmqm** command are listed in [System and default objects](#). When you have created a queue manager, use the **strmqm** command to start it.

The queue manager is automatically associated with the installation from which the **crtmqm** command was issued. To change the associated installation, use the **setmqm** command.

**Windows** Note that the Windows installer does not automatically add the user that performs the installation to the mqm group. For more details, see [Authority to administer IBM MQ on AIX, Linux, and Windows systems](#).

## Usage notes

**Linux**

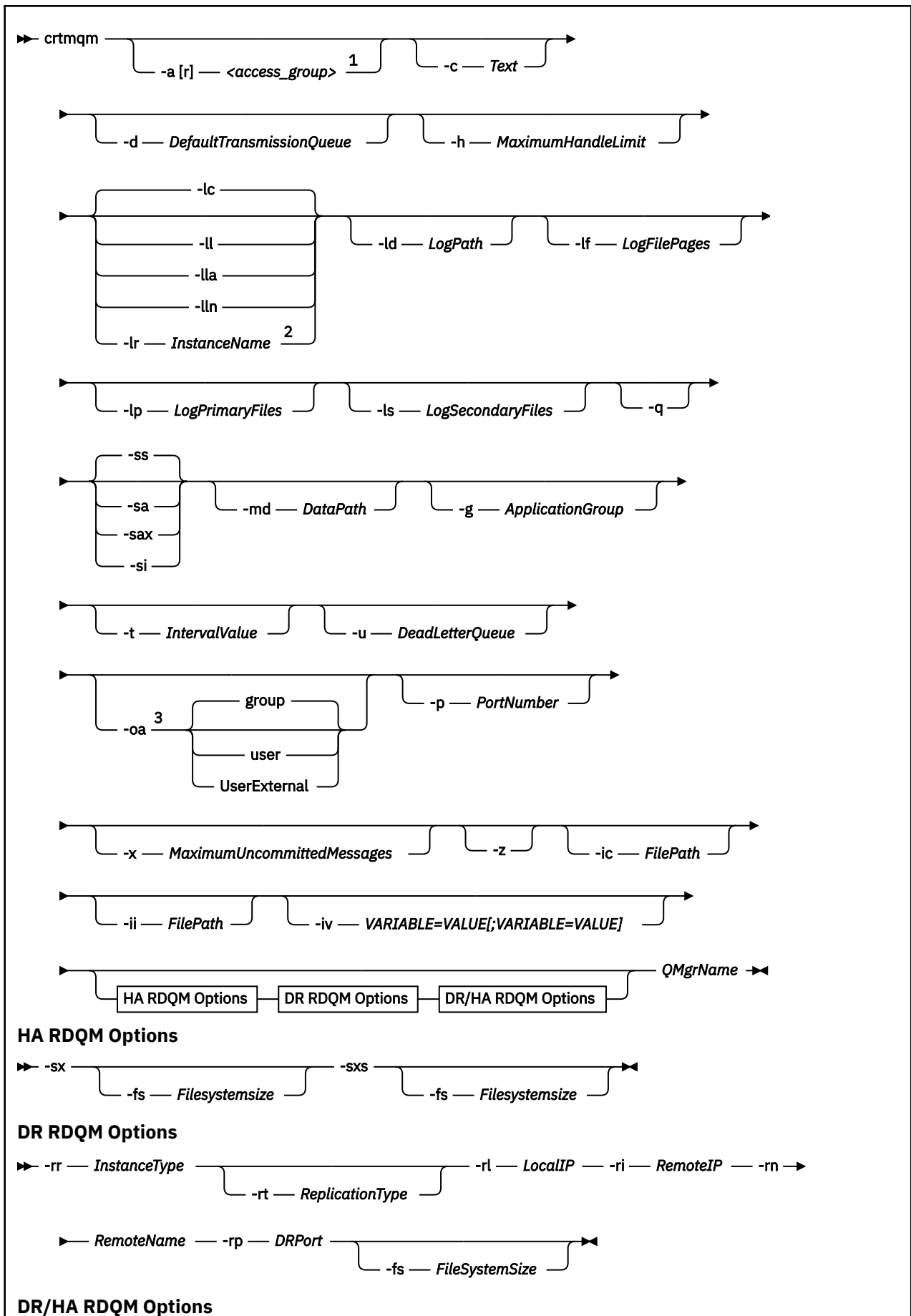
**Linux** From IBM MQ 9.2.0, you can use the environment variable MQLICENSE to accept or view the license.

The options you can set the MQLICENSE environment variable to are 'accept' or 'view'. Other values are treated as if the environment variable is not set.

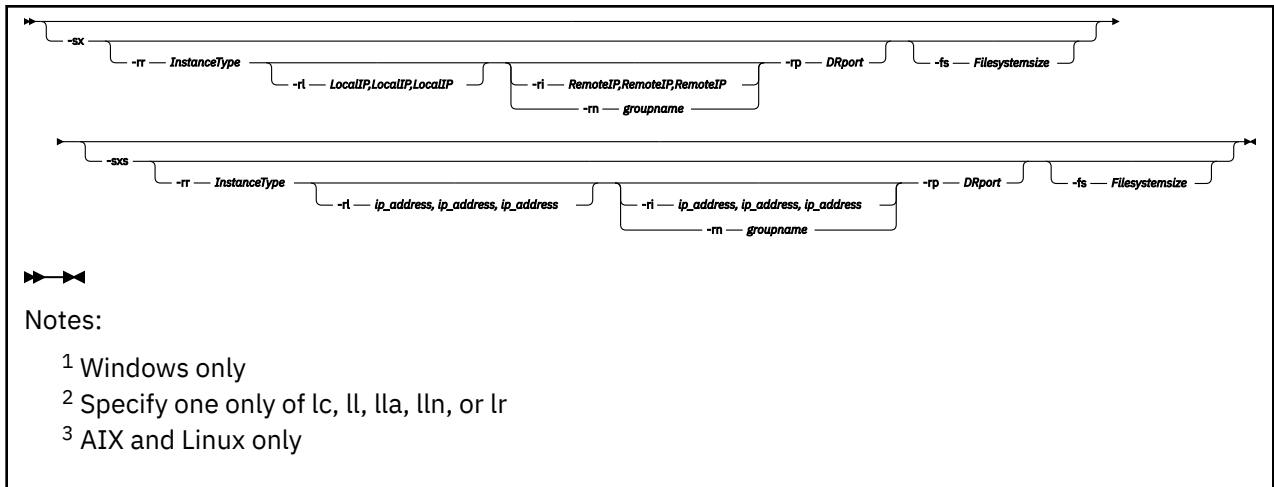
You can also set the MQLICENSE environment variable on the **addmqinf** and **dspmqinf** commands. However, you cannot set this environment variable directly on the **setmqprd** command.

**Note:** You can use this environment variable only when the licence has not already been accepted in the normal way (that is, running **mqlicense exe**). If the license has already been accepted, this environment variable is ignored regardless of the value.

## Syntax







## Required parameters

### QMgrName

The name of the queue manager that you want to create. The name can contain up to 48 characters. This parameter must be the last item in the command.

**Note:** The *QMgrName* is used by IBM MQ applications, other IBM MQ queue managers, and IBM MQ control commands to identify this queue manager.

No other queue manager with the same name can exist on this machine. Where this queue manager is going to connect to other queue managers you must ensure that queue manager names are unique within that group of queue managers.

The *QMgrName* is also used to name the directories created on disk for the queue manager. Due to filesystem limitations the name of the directories created might not be identical to the *QMgrName* supplied on the `crtmqm` command.

In these cases the directories created will be based upon the supplied *QMgrName*, but might be modified, or have a suffix such as `.000` or `.001`, and so on, added to the queue manager name.

## Optional parameters

### Windows `-a AccessGroup` or `-ar AccessGroup`

Use the access group parameter to specify a Windows security group, members of which will be granted full access to all queue manager data files. The group can either be a local or global group, depending on the syntax used.

Valid syntax for the group name is as follows:

*LocalGroup*  
*Domain name\GlobalGroup name*  
*GlobalGroup name @ Domain name*

You must define the additional access group before running the `crtmqm` command with the `-a` or the `-ar` option.

If you specify the group using `-ar` instead of `-a`, the local mqm group is not granted access to the queue manager data files. Use this option if the file system hosting the queue manager data files does not support access control entries for locally defined groups.

The group is typically a global security group, which is used to provide multi-instance queue managers with access to a shared queue manager data and logs folder. Use the additional security access group to set read and write permissions on the folder or to share containing queue manager data and log files.

The additional security access group is an alternative to using the local group named `mqm` to set permissions on the folder containing queue manager data and logs. Unlike the local group `mqm`, you can make the additional security access group a local or a global group. It must be a global group to set permissions on the shared folders that contain the data and log files used by multi-instance queue managers.

The Windows operating system checks the access permissions to read and write queue manager data and log files. It checks the permissions of the user ID that is running queue manager processes. The user ID that is checked depends on whether you started the queue manager as a service or you started it interactively. If you started the queue manager as a service, the user ID checked by the Windows system is the user ID you configured with the **Prepare** IBM MQ wizard. If you started the queue manager interactively, the user ID checked by the Windows system is the user ID that ran the **strmqm** command.

The user ID must be a member of the local `mqm` group to start the queue manager. If the user ID is a member of the additional security access group, the queue manager can read and write files that are given permissions by using the group.

**Restriction:** You can specify an additional security access group only on Windows operating system. If you specify an additional security access group on other operating systems, the **crtmqm** command returns an error.

#### **-c Text**

Descriptive text for this queue manager. You can use up to 64 characters; the default is all blanks.

If you include special characters, enclose the description in single quotation marks. The maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

#### **-d DefaultTransmissionQueue**

The name of the local transmission queue where remote messages are put if a transmission queue is not explicitly defined for their destination. There is no default.

#### **MQ Appliance -fe**

Specifies that the file system for the queue manager is encrypted. You can optionally specify the encryption passphrase by using the **-fp** parameter, otherwise you are prompted for the passphrase when you run the command.

#### **MQ Appliance -fp Passphrase**

Used in conjunction with the **-fe** parameter, optionally specifies the encryption passphrase. If you specify **-fe** but omit **-fp** you are prompted for the passphrase when you run the command. The passphrase can be between 1 and 512 characters. You should store a copy of the passphrase somewhere safe.

**Linux**

**AIX**

#### **-g ApplicationGroup**

On AIX and Linux, the name of the group that contains members that are allowed to perform the following actions:

- Run MQI applications
- Update all IPCC resources
- Change the contents of some queue manager directories

The default value is **-g all**, which allows unrestricted access.

The **-g ApplicationGroup** value is recorded in the queue manager configuration file, `qm.ini`.

The `mqm` user ID and the user running the command must belong to the specified Application Group. For further details of the operation of restricted mode, see [Restricted mode](#).

#### **-h MaximumHandleLimit**

The maximum number of handles that an application can open at the same time.

Specify a value in the range 1 - 999999999. The default value is 256.

### Multi **-ic FilePath**

Automatic configuration of MQSC attributes.

Specify the location containing MQSC commands to be automatically applied to the queue manager on every queue manager restart. This can be a filename, or a directory where each file \*.mqsc is automatically processed. See [automatic configuration from an MQSC script at startup](#) for more details.

### Multi **-ii FilePath**

Automatic configuration of qm.ini file attributes.

Specify a configuration to be automatically applied to the qm.ini file on every queue manager restart. This can be a filename containing INI format information, or a directory where \*.ini is automatically processed. See [automatic configuration from an INI script at startup](#) for more details.

### Multi **-iv VARIABLE=VALUE[;VARIABLE=VALUE]**

Configuration variable for use with automatic uniform clusters.

Specify a name and associated value for use as an insert during MQSC definitions. This parameter is only used for CONNAME fields in defining cluster receivers for automatic uniform clusters. For example:

```
-iv CONNAME=QMA.host.name(1414)
```

The next set of parameter descriptions relate to logging, which is described in [Using the log for recovery](#).

**Note:** Choose the logging arrangements with care, because some cannot be changed after they are committed. The defaults for the logging options to **crtmqm** can be overridden by attributes in the mqs.ini file.

If you specify the logging attributes in the mqs.ini file, those attributes override the default values of the logging command line parameters to **crtmqm**.

### IBM i **-lb BufferSize**

The journal buffer size. This is a number in the range 32000 - 15761440. The default is 32000.

### **-lc**

Use circular logging. This method is the default logging method.

### **-ld LogPath**

The directory used to store log files. The default directory to store log paths is defined when you install IBM MQ.

If the volume containing the log file directory supports file security, the log file directory must have access permissions. The permissions allow the user IDs, under whose authority the queue manager runs, read and write access to the directory and its subdirectories. When you install IBM MQ, you grant permissions to the user IDs and to the mqm group on the default log directory. If you set the *LogPath* parameter to write the log file to a different directory, you must grant the user IDs permission to read and write to the directory. The user ID and permissions for AIX and Linux are different from those for the Windows system:

### Linux AIX and Linux

The directory and its subdirectories must be owned by the user mqm in the group mqm.

If the log file is shared between different instances of the queue manager, the security identifiers (sid) that are used must be the same for the different instances. You must have set the user mqm to the same sid on the different servers running instances of the queue manager. Likewise for the group mqm.

## Windows Windows

If the directory is accessed by only one instance of the queue manager, you must give read and write access permission to the directory for the following groups and users:

- The local group mqm
- The local group Administrators
- The SYSTEM user ID

To give different instances of a queue manager access to the shared log directory, the queue manager must access the log directory using a global user. Give the global group, which contains the global user, read and write access permission to the log directory. The global group is the additional security access group specified in the **-a** parameter.

## Windows

In IBM MQ for Windows systems, the default directory is `C:\ProgramData\IBM\MQ\log` (assuming that C: is your data drive). If the volume supports file security, the SYSTEM ID, Administrators, and mqm group must be granted read/write access to the directory.

## Linux

## AIX

In IBM MQ for AIX or Linux systems, the default directory is `/var/mqm/log`. User ID mqm and group mqm must have full authorities to the log files.

If you change the locations of these files, you must give these authorities yourself. If these authorities are set automatically, then the log files are in their default locations.

## IBM i

On IBM i the *logpath* directory is the queue manager library.

### -lf *LogFilePages*

The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

## Linux

## AIX

In IBM MQ for AIX or Linux systems, the default number of log file pages is 4096, giving a log file size of 16 MB. The minimum number of log file pages is 64 and the maximum is 65535.

## Windows

In IBM MQ for Windows systems, the default number of log file pages is 4096, giving a log file size of 16 MB. The minimum number of log file pages is 32 and the maximum is 65535.

**Note:** The size of the log files for a queue manager specified during creation of that queue manager cannot be changed.

## IBM i

On IBM i this parameter is used to specify the journal receiver threshold.

## Multi

### -ll *LinearLogging*

Use linear logging.

On Multiplatforms, if you create a queue manager using the existing **-ll** option, you need to carry out manual management of log extents as previously (**LogManagement=Manual**).

## Multi

### -lla

Use linear logging with automatic management of log extents (**LogManagement=Automatic**).

## Multi -lln

Use linear logging with archive management of log extents (**LogManagement=Archive**).

## CP4I MQ Adv. -lr *InstanceName*

Use log replication. Specify this option when configuring a Native HA group. The *InstanceName* provided is used by Native HA to identify this copy of log data and must be unique. The *InstanceName* can contain up to 48 characters. Valid characters in an *InstanceName* are:

- Uppercase or lowercase alphabetic characters (A-Z, a-z)
- Numeric characters (0-9)
- Dash (-), the leading character is not permitted to be a dash
- Period (.)
- Underscore (\_)

Leading or embedded blanks are not permitted.

### -lp *LogPrimaryFiles*

The log files allocated when the queue manager is created.

#### Windows

On a Windows system:

- The minimum number of primary log files that you can have is 2 and the maximum is 254.
- The total number of primary and secondary log files must not exceed 255 and must not be less than 3.

#### Linux

#### AIX

On AIX and Linux systems:

- The minimum number of primary log files you can have is 2 and the maximum is 510. The default is 3.
- The total number of primary and secondary log files must not exceed 511 and must not be less than 3.

Operating system limits can reduce the maximum log size.

The value is examined when the queue manager is created or started. You can change it after the queue manager has been created. However, a change in the value is not effective until the queue manager is restarted, and the effect might not be immediate.

For more information about primary log files, see [What logs look like](#).

To calculate the size of the primary log files, see [Calculating the size of the log](#).

### -ls *LogSecondaryFiles*

The log files allocated when the primary files are exhausted.

#### Windows

On a Windows system:

- The minimum number of secondary log files that you can have is 1 and the maximum is 253.
- The total number of primary and secondary log files must not exceed 255 and must not be less than 3.

#### Linux

#### AIX

On AIX and Linux systems:

- The minimum number of secondary log files that you can have is 2 and the maximum is 509. The default is 2.
- The total number of primary and secondary log files must not exceed 511 and must not be less than 3.

Operating system limits can reduce the maximum log size.

The value is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

For more information about the use of secondary log files, see [What logs look like](#).

To calculate the size of the secondary log files, see [Calculating the size of the log](#).

### IBM i **-lz ASPInfo**

Specify an auxiliary storage pool number (1-32, default 1) or an auxiliary storage pool device name for the IBM i journal.

### **-md DataPath**

#### Linux

The directory used to hold the data files for a queue manager.

#### Windows

In IBM MQ for Windows systems, the default is `C:\ProgramData\IBM\MQ\mqm\qmgrs` (assuming that C: is your data drive). If the volume supports file security, the SYSTEM ID, Administrators, and mqm group must be granted read/write access to the directory.

#### Linux

#### AIX

In IBM MQ for AIX or Linux systems, the default is `/var/mqm/qmgrs`. User ID mqm and group must have full authorities to the log files.

mqm

#### Linux

For RDQM on Linux systems, the default is `/var/mqm/vols/qmgrname/qmgr/`.

The **DataPath** parameter is provided to assist in the configuration of multi-instance queue managers. For example, on AIX and Linux systems: if the `/var/mqm` directory is located on a local file system, use the **DataPath** parameter and the **LogPath** parameter to point to the shared file systems accessible to multiple queue managers.

**Note:** A queue manager created using **DataPath** parameter runs on versions of the product earlier than IBM WebSphere® MQ 7.0.1, but the queue manager must be reconfigured to remove the **DataPath** parameter. You have two options to restore the queue manager to a pre-IBM WebSphere MQ 7.0.1 configuration and run without the **DataPath** parameter: If you are confident about editing queue manager configurations, you can manually configure the queue manager using the **Prefix** queue manager configuration parameter. Alternatively, complete the following steps to edit the queue manager:

1. Stop the queue manager.
2. Save the queue manager data and log directories.
3. Delete the queue manager.
4. Backout IBM WebSphere MQ to the pre-IBM WebSphere MQ 7.0.1 fix level.
5. Create the queue manager with the same name.
6. Replace the new queue manager data and log directories with the ones you saved.

## **-oa group|user|UserExternal**

Linux

AIX

On AIX and Linux systems, you can specify whether group or user authorization is to be used. If you do not set this parameter, group authorization is used. You can change the authorization model later by setting the **SecurityPolicy** parameter in the Service stanza of the `qm.ini` file (see [Service stanza of the qm.ini file](#)).

V 9.3.0

From IBM MQ 9.3.0, you can use the additional option of `UserExternal` when creating new queue managers. If you select this option, you can create a non-operating system user name, with a maximum of 12 characters, that:

- Must conform to the [Rules for naming IBM MQ objects](#)
- Is not known to the system
- Can be used both for checking and setting authorizations

V 9.3.0

If you create a non-operating system user name, that user is considered to belong to no groups, except the nobody group. See [Principals and groups on AIX, Linux, and Windows](#) for more information.

For further information, see [Object authority manager \(OAM\)](#).

## **-p PortNumber**

Create a managed TCP listener on the specified port.

Specify a valid port value in the range 1-65535, to create a TCP listener object that uses the specified port. The new listener is called `SYSTEM.LISTENER.TCP.1`. This listener is under queue manager control, and is started and stopped along with the queue manager.

## **-q**

Makes this queue manager the default queue manager. The new queue manager replaces any existing default queue manager.

If you accidentally use this flag and you want to revert to an existing queue manager as the default queue manager, change the default queue manager as described in [Making an existing queue manager the default](#).

Linux

## **-rr InstanceType**

Create a disaster recovery replicated data queue manager (DR RDQM). Specify **-rr p** to create the primary instance of the queue manager or specify **-rr s** to create the secondary instance. You must be `root` or a user in the `mqm` group with `sudo` privileges to use this command.

Use **-rr** with the **-sx** or the **-sxs** parameter to create a DR/HA RDQM.

Linux

## **-rt ReplicationType**

Optionally specify whether your DR RDQM configuration uses synchronous or asynchronous replication. Specify **-rt s** for synchronous and **-rt a** for asynchronous. Asynchronous is the default.

Linux

## **-rl LocalIP**

Specify the local IP address used for replication of data between primary and secondary instances of a DR RDQM.

Use **-rl LocalIP,LocalIP,LocalIP** with the **-sx** or the **-sxs** parameter to create a DR/HA RDQM and specify the three IP addresses used for DR replication on the local HA group.

### Linux **-ri RemoteIP**

Specify the remote IP address used for replication of data between primary and secondary instances of a DR RDQM.

Use **-ri RemoteIP,RemoteIP,RemoteIP** with the **-sx** or the **-sxs** parameter to create a DR/HA RDQM and specify the three IP addresses used for DR replication on the remote HA group. You must specify either the **-ri** or the **-rn** parameter when creating a DR/HA RDQM.

### Linux **-rn RemoteName**

Specifies the name of the system that is hosting the other instance of the queue manager. The name is the `+` value that is returned if you run `uname -n` on that server.

Use **-rn GroupName** with the **-sx** or the **-sxs** parameter to create a DR/HA RDQM and specify name of the remote HA group. The *GroupName* refers to the group defined in the DRGroup stanza in the `rdqm.ini` file. You must specify either the **-rn** or the **-ri** parameter when creating a DR/HA RDQM.

### Linux **-rp DRPortx**

Specifies the port to use for DR replication.

### MQ Appliance **-sa**

Automatic queue manager startup, for the appliance. The queue manager is configured to start automatically when the appliance restarts. This argument is mutually exclusive with **-sx**.

### Windows **-sa**

Automatic queue manager startup. For Windows systems.

The queue manager is configured to start automatically when the IBM MQ Service starts.

This is the default option if you create a queue manager from IBM MQ Explorer.

### Windows **-sax**

Automatic queue manager startup, permitting multiple instances. For Windows systems only.

The queue manager is configured to start automatically when the IBM MQ Service starts.

If an instance of the queue manager is not already running the queue manager starts, the instance becomes active, and standby instances are permitted elsewhere. If a queue manager instance that permits standbys is already active on a different server, the new instance becomes a standby instance.

Only one instance of a queue manager can run on a server.

### **-si**

Interactive (manual) queue manager startup.

The queue manager is configured to start only when you manually request startup by using the **strmqm** command. The queue manager runs under the (interactive) user when that user is logged-on. Queue managers configured with interactive startup end when the user who started them logs off.

### **-ss**

Service (manual) queue manager startup.

A queue manager configured to start only when manually requested by using the **strmqm** command. The queue manager then runs as a child process of the service when the IBM MQ Service starts. Queue managers configured with service startup continue to run even after the interactive user has logged off.

This is the default option if you create a queue manager from the command line.



## Linux **-sx** [*DR parameters*][*-fs FilesystemSize*]

Create a high availability replicated data queue manager (HA RDQM) on the primary node for that queue manager (do not specify DR parameters). RDQM is a high availability solution that is available on Linux only. See [Creating an HA RDQM](#) for more details about creating an RDQM. You must be root or a user in the mqm group with sudo privileges to use this command. The default size for file system size is 3 GB. You can specify a different file system size using the `-fs` option. The default unit is GB (so `-fs 8` creates an 8 GB file system size). You can specify a different unit, for example, specify `-fs 1024M` to create a 1024 MB file system size. The queue manager is started automatically.

Specify DR parameters to create a DR/HA RDQM on the primary node for that queue manager. See [Creating DR/HA RDQMs](#) for details. The DR parameters are **-rr**, **-ri**, **-rl**, **-rn**, **-rp**.

## Linux **-sxs** [*DR parameters*][*-fs FilesystemSize*]

Create a replicated data queue manager (RDQM) on a secondary node (do not specify DR parameters). RDQM is a high availability solution that is available on Linux only. See [Creating an HA RDQM](#) for more details about creating an RDQM. You must be the root user to use this command. The default size for file system size is 3 GB. The default size for file system size is 3 GB. You can specify a different file system size using the `-fs` option. The default unit is GB (so `-fs 8` creates an 8 GB file system size). You can specify a different unit, for example, specify `-fs 1024M` to create a 1024 MB file system size.

Specify DR parameters to create a DR/HA RDQM on a secondary node. See [Creating DR/HA RDQMs](#) for details. The DR parameters are **-rr**, **-ri**, **-rl**, **-rn**, **-rp**.

### **-t IntervalValue**

The trigger time interval in milliseconds for all queues controlled by this queue manager. This value specifies the length of time triggering is suspended, after the queue manager receives a trigger-generating message. That is, if the arrival of a message on a queue causes a trigger message to be put on the initiation queue, any message arriving on the same queue within the specified interval does not generate another trigger message.

You can use the trigger time interval to ensure that your application is allowed sufficient time to deal with a trigger condition before it is alerted to deal with another trigger condition on the same queue. You might choose to see all trigger events that happen; if so, set a low or zero value in this field.

Specify a value in the range 0 - 999999999. The default is 999999999 milliseconds; a time of more than 11 days. Allowing the default to be used effectively means that triggering is disabled after the first trigger message. However, an application can enable triggering again by servicing the queue using a command to alter the queue to reset the trigger attribute.

### **-u DeadLetterQueue**

The name of the local queue that is to be used as the dead-letter (undelivered-message) queue. Messages are put on this queue if they cannot be routed to their correct destination.

The default is no dead-letter queue.

### **-x MaximumUncommittedMessages**

The maximum number of uncommitted messages under any one sync point. The uncommitted messages are the sum of:

- The number of messages that can be retrieved from queues
- The number of messages that can be put on queues
- Any trigger messages generated within this unit of work

This limit does not apply to messages that are retrieved or put outside a sync point.

Specify a value in the range 1 - 999999999. The default value is 10000 uncommitted messages.

### **-z**

Suppresses error messages.

This flag is used within IBM MQ to suppress unwanted error messages. Do not use this flag when using a command line. Using this flag can result in a loss of information.

## Return codes

Table 26. Return code identifiers and descriptions




Return code	Description
0	Queue manager created
8	Queue manager exists
18	Invalid trigger interval
19	Invalid dead letter queue
20	Invalid default transmit queue
21	Invalid max handles value
22	Invalid max uncommitted messages value
25	Error creating the queue manager directory structure
37	Invalid queue manager description
38	The access group specified cannot be found
39	Invalid parameter specified
49	Queue manager stopping
58	Inconsistent use of installations detected
63	Invalid Native HA instance name
69	Storage unavailable
70	Queue space unavailable
71	Unexpected error
72	Queue manager name error
74	The IBM MQ service is not started
 CP4I	Log replication is unavailable on this platform
	
93	
95	Log replication is incompatible with RDQM
100	Log location invalid
105	The queue manager was created but could not be set as the default queue manager
111	Queue manager created. However, there was a problem processing the default queue manager definition in the product configuration file. The default queue manager specification might be incorrect
115	Invalid log size
119	 Permission denied ( Windows only)
155	The group ID specified is not valid
156	The owning group ID can only be changed on AIX and Linux systems

Table 26. Return code identifiers and descriptions (continued)

Return code	Description
157	The group ID chosen is invalid

### Examples

- The following command creates a default queue manager called `Paint.queue.manager`, with a description of `Paint shop`, and creates the system and default objects. It also specifies that linear logging is to be used:

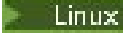
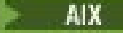
```
crtmqm -c "Paint shop" -ll -q Paint.queue.manager
```

- The following command creates a default queue manager called `Paint.queue.manager`, creates the system and default objects, and requests two primary and three secondary log files:

```
crtmqm -c "Paint shop" -ll -lp 2 -ls 3 -q Paint.queue.manager
```

- The following command creates a queue manager called `travel`, creates the system and default objects, sets the trigger interval to 5000 milliseconds (5 seconds), and specifies `SYSTEM.DEAD.LETTER.QUEUE` as its dead-letter queue.

```
crtmqm -t 5000 -u SYSTEM.DEAD.LETTER.QUEUE travel
```

-   The following command creates a queue manager called `QM1` on AIX and Linux systems, which has log and queue manager data folders in a common parent directory. The parent directory is to be shared on highly available networked storage to create a multi-instance queue manager. Before issuing the command, create other parameters `/MQHA`, `/MQHA/logs` and `/MQHA/qmgrs` owned by the user and group `mqm`, and with permissions `rxwxrxrx-x`.

```
crtmqm -ld /MQHA/logs -md /MQHA/qmgrs QM1
```

### Related concepts

[Working with dead-letter queues](#)

### Related reference

[strmqm \(start queue manager\)](#)

Start a queue manager or ready it for standby operation.

[endmqm \(end queue manager\)](#)

Stop a queue manager or switch to a standby queue manager or to a replica queue manager.

[dlmqm \(delete queue manager\)](#)

Delete a queue manager.

[setmqm \(set the associated installation of a queue manager\)](#)

Set the associated installation of a queue manager.

### **dlmqinst (delete MQ installation)**

Delete installation entries from `mqinst.ini` on AIX and Linux systems.

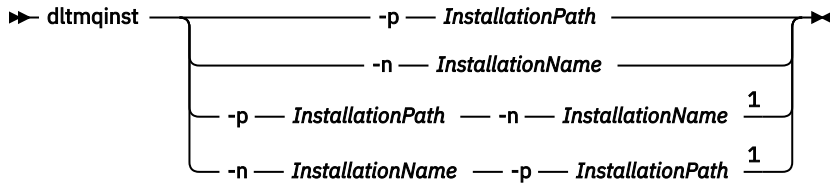
### Purpose

File `mqinst.ini` contains information about all IBM MQ installations on a system. For more information about `mqinst.ini`, see [Installation configuration file, mqinst.ini](#).



**Attention:** Only the user root can run this command.

## Syntax



Notes:

<sup>1</sup> When specified together, the installation name and installation path must refer to the same installation.

## Parameters

### **-n InstallationName**

The name of the installation.

### **-p InstallationPath**

The installation path is the location where IBM MQ is installed.

## Return codes

Table 27. Return code identifiers and descriptions

Return code	Description
0	Entry deleted without error
5	Entry still active
36	Invalid arguments supplied
44	Entry does not exist
59	Invalid installation specified
71	Unexpected error
89	ini file error
96	Could not lock ini file
98	Insufficient authority to access ini file
131	Resource problem

## Example

1. This command deletes an entry with an installation name of myInstallation, and an installation path of /opt/myInstallation:

```
dltmqinst -n MyInstallation -p /opt/myInstallation
```

**Note:** You can only use the **dltmqinst** command on another installation from the one it runs from. If you only have one IBM MQ installation, the command will not work.

**dltmqm (delete queue manager)**

Delete a queue manager.

**Purpose**

Use the **dltmqm** command to delete a specified queue manager and all objects associated with it. Before you can delete a queue manager, you must end it using the **endmqm** command.

You must use the **dltmqm** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmqr -o installation` command.

**Windows**

On Windows, it is an error to delete a queue manager when queue manager files are open. If you get this error, close the files and reissue the command.

**Syntax**

```
▶▶ dltmqm -z QMgrName ▶▶
```

**Required parameters****QMGrName**

The name of the queue manager to delete.

**Optional parameters****-z**


Suppresses error messages.

**Return codes**

Table 28. Return code identifiers and descriptions

Return code	Description
0	Queue manager deleted
3	Queue manager being created
5	Queue manager running
16	Queue manager does not exist
24	A process that was using the previous instance of the queue manager has not yet disconnected.
25	An error occurred while creating or checking the directory structure for the queue manager.
26	Queue manager running as a standby instance.
27	Queue manager could not obtain data lock.
29	Queue manager deleted, however there was a problem removing it from Active Directory.
33	An error occurred while deleting the directory structure for the queue manager.
39	Invalid parameter specified
49	Queue manager stopping
58	Inconsistent use of installations detected

Table 28. Return code identifiers and descriptions (continued)

Return code	Description
62	The queue manager is associated with a different installation
69	Storage not available
71	Unexpected error
72	Queue manager name error
74	The IBM MQ service is not started.
100	Log location invalid.
112	Queue manager deleted. However, there was a problem processing the default queue manager definition in the product configuration file. The default queue manager specification might be incorrect.
119	 Permission denied ( Windows only).

## Examples


1. The following command deletes the queue manager `saturn.queue.manager`.

```
dltmqm saturn.queue.manager
```

2. The following command deletes the queue manager `travel` and also suppresses any messages caused by the command.

```
dltmqm -z travel
```

## Usage notes

 On Windows, it is an error to delete a queue manager when queue manager files are open. If you get this error, close the files and reissue the command.

Deleting a cluster queue manager does not remove it from the cluster. To check whether the queue manager you want to delete is part of a cluster, issue the command **DIS CLUSQMGR(\*)**. Then check whether this queue manager is listed in the output. If it is listed as a cluster queue manager you must remove the queue manager from the cluster before deleting it. See the related link for instructions.

If you do delete a cluster queue manager without first removing it from the cluster, the cluster continues to regard the deleted queue manager as a member of the cluster for at least 30 days. You can remove it from the cluster using the command **RESET CLUSTER** on a full repository queue manager. Re-creating a queue manager with an identical name and then trying to remove that queue manager from the cluster does not result in the cluster queue manager being removed from the cluster. This is because the newly created queue manager, although having the same name, does not have the same queue manager ID (QMID). Therefore it is treated as a different queue manager by the cluster.

### Related reference

[crtmqm \(create queue manager\)](#)

Create a queue manager.

[strmqm \(start queue manager\)](#)

Start a queue manager or ready it for standby operation.

[endmqm \(end queue manager\)](#)

Stop a queue manager or switch to a standby queue manager or to a replica queue manager.

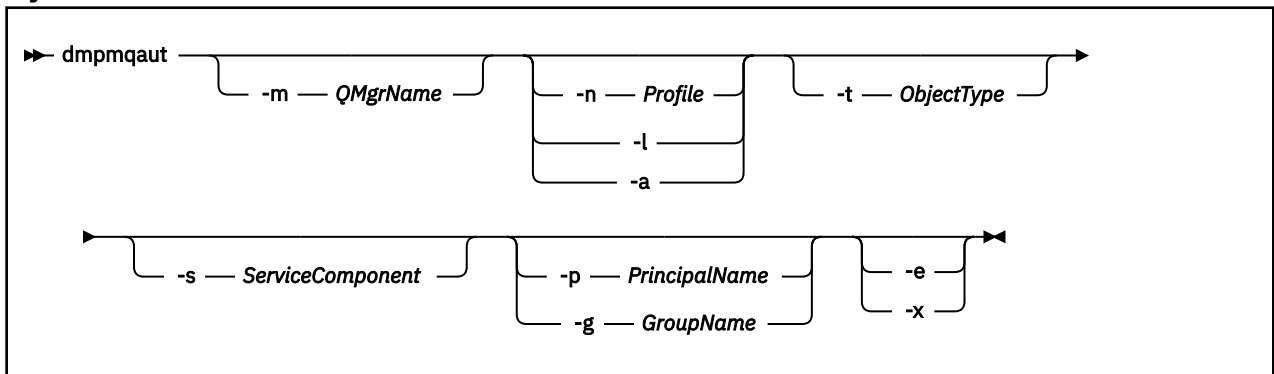
## Multi **dmpmqaut (dump MQ authorizations)**

Dump a list of current authorizations for a range of IBM MQ object types and profiles.

### Purpose

Use the **dmpmqaut** command to dump the current authorizations to a specified object.

### Syntax



### Optional parameters

#### **-m** *QMGrName*

Dump authority records only for the queue manager specified. If you omit this parameter, only authority records for the default queue manager are dumped.

#### **-n** *Profile*

The name of the profile for which to dump authorizations. The profile name can be generic, using wildcard characters to specify a range of names as explained in [Using OAM generic profiles on AIX, Linux, and Windows systems](#).

#### **-l**

Dump only the profile name and type. Use this option to generate a *terse* list of all defined profile names and types.

#### **-a**

Generate set authority commands.

#### **-t** *ObjectType*

The type of object for which to dump authorizations. Possible values are:

A table showing possible values, and descriptions, for the -t flag.

Value	Description
<b>authinfo</b>	An authentication information object, for use with TLS channel security
<b>channel</b> or <b>chl</b>	A channel
<b>clntconn</b> or <b>clcn</b>	A client connection channel
<b>listener</b> or <b>lstr</b>	A listener
<b>namelist</b> or <b>nl</b>	A namelist
<b>process</b> or <b>prcs</b>	A process
<b>queue</b> or <b>q</b>	A queue or queues matching the object name parameter
<b>qmgr</b>	A queue manager
<b>rqmname</b> or <b>rqmn</b>	A remote queue manager name
<b>service</b> or <b>srvc</b>	A service
<b>topic</b> or <b>top</b>	A topic

### **-s ServiceComponent**

If installable authorization services are supported, specifies the name of the authorization service for which to dump authorizations. This parameter is optional; if you omit it, the authorization inquiry is made to the first installable component for the service.

#### **Windows** -p **PrincipalName**

This parameter applies to Windows only; AIX and Linux systems keep only group authority records.

The name of a user for whom to dump authorizations to the specified object. The name of the principal can optionally include a domain name, specified in the following format:

```
userid@domain
```

For more information about including domain names on the name of a principal, see [Principals and groups](#).

### **-g GroupName**

The name of the user group for which to dump authorizations. You can specify only one name, which must be the name of an existing user group.

**Windows** For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following formats:

```
GroupName@domain  
domain\GroupName
```

### **-e**

Display all profiles used to calculate the cumulative authority that the entity has to the object specified in -n *Profile*. The variable *Profile* must not contain any wildcard characters.

The following parameters must also be specified:

- -m *QMgrName*
- -n *Profile*
- -t *ObjectType*

and either -p *PrincipalName*, or -g *GroupName*.



**-x**

Display all profiles with the same name as specified in **-n Profile**. This option does not apply to the QMGR object, so a dump request of the form `dmpmqaut -m QM -t QMGR ... -x` is not valid.

## Examples

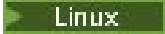

The following examples show the use of **dmpmqaut** to dump authority records for generic profiles:

1. This example dumps all authority records with a profile that matches queue a.b.c for principal user1.

```
dmpmqaut -m qm1 -n a.b.c -t q -p user1
```

The resulting dump would look something like this:

```
profile:    a.b.*
object type: queue
entity:     user1
type:       principal
authority:  get, browse, put, inq
```

**Note:**   On AIX and Linux, you cannot use the `-p` option. You must use `-g groupname` instead.

2. This example dumps all authority records with a profile that matches queue a.b.c.

```
dmpmqaut -m qmgr1 -n a.b.c -t q
```

The resulting dump would look something like this:

```
profile:    a.b.c
object type: queue
entity:     Administrator
type:       principal
authority:  all
-----
profile:    a.b.*
object type: queue
entity:     user1
type:       principal
authority:  get, browse, put, inq
-----
profile:    a.**
object type: queue
entity:     group1
type:       group
authority:  get
```

3. This example dumps all authority records for profile a.b.\*, of type queue.

```
dmpmqaut -m qmgr1 -n a.b.* -t q
```

The resulting dump would look something like this:

```
profile:    a.b.*
object type: queue
entity:     user1
type:       principal
authority:  get, browse, put, inq
```

4. This example dumps all authority records for queue manager qmX.

```
dmpmqaut -m qmX
```

The resulting dump would look something like this:

```
profile:    q1
object type: queue
entity:    Administrator
type:      principal
authority:  all
-----
profile:    q*
object type: queue
entity:    user1
type:      principal
authority:  get, browse
-----
profile:    name.*
object type: namelist
entity:    user2
type:      principal
authority:  get
-----
profile:    pr1
object type: process
entity:    group1
type:      group
authority:  get
```

5. This example dumps all profile names and object types for queue manager qmX.

```
dmpmqaut -m qmX -l
```

The resulting dump would look something like this:

```
profile: q1, type: queue
profile: q*, type: queue
profile: name.*, type: namelist
profile: pr1, type: process
```

#### Note:

1.  For Windows only, all principals displayed include domain information, for example:

```
profile:    a.b.*
object type: queue
entity:    user1@domain1
type:      principal
authority:  get, browse, put, inq
```


2. Each class of object has authority records for each group or principal. These records have the profile name @CLASS and track the crt (create) authority common to all objects of that class. If the crt authority for any object of that class is changed then this record is updated. For example:

```
profile:    @class
object type: queue
entity:    test
entity type: principal
authority:  crt
```

This shows that members of the group test have crt authority to the class queue.



**Attention:** You cannot delete the @CLASS entries (the system is working as designed)

3.  For Windows only, members of the "Administrators" group are by default given full authority. This authority, however, is given automatically by the OAM, and is not defined by the authority records. The **dmpmqaut** command displays authority defined only by the authority records. Unless an authority record has been explicitly defined, therefore, running the **dmpmqaut** command against the "Administrators" group displays no authority record for that group.

## Related reference

[“setmqaut \(grant or revoke authority\)” on page 193](#)

Change the authorizations to a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

[“DISPLAY AUTHREC \(display authority records\) on Multiplatforms” on page 646](#)

Use the MQSC command DISPLAY AUTHREC to display the authority records associated with a profile name.

[“SET AUTHREC \(set authority records\) on Multiplatforms” on page 920](#)

Use the MQSC command SET AUTHREC to set authority records associated with a profile name.

## **dmpmqcfcfg (dump queue manager configuration)**

Use the **dmpmqcfcfg** command to dump the configuration of an IBM MQ queue manager.

### Purpose

Use the **dmpmqcfcfg** command to dump the configuration of IBM MQ queue managers. If any default object has been edited, the **-a** option must be used if the dumped configuration will be used to restore the configuration.



**CAUTION:** When moving a queue manager from one operating system to another, you use **dmpmqcfcfg** to save the configuration information of the queue manager that you want to move, and then copy the object definitions across to the new queue manager that you create on the new operating system. You must take great care with copying the object definitions, because some manual modification of the definitions might be needed. For more information, see [Moving a queue manager to a different operating system](#).

The **dmpmqcfcfg** utility dumps only subscriptions of type MQSUBTYPE\_ADMIN, that is, only subscriptions that are created using the MQSC command **DEFINE SUB** or its PCF equivalent. The output from **dmpmqcfcfg** is a **runmqsc** command to enable the administration subscription to be re-created. Subscriptions that are created by applications using the MQSUB MQI call of type MQSUBTYPE\_API are not part of the queue manager configuration, even if durable, and so are not dumped by **dmpmqcfcfg**. MQTT channels will only be returned for types **-t all** and **-t mqttchl** if the telemetry (MQXR) service is running. For instructions on how to start the telemetry service, see [Administering MQ Telemetry](#).

From IBM MQ 8.0, the output of the **dmpmqcfcfg** is changed to ensure that password fields are commented out in the generated commands. This change brings the **dmpmqcfcfg** command in line with the DISPLAY commands, that show password fields as PASSWORD(\*\*\*\*\*).

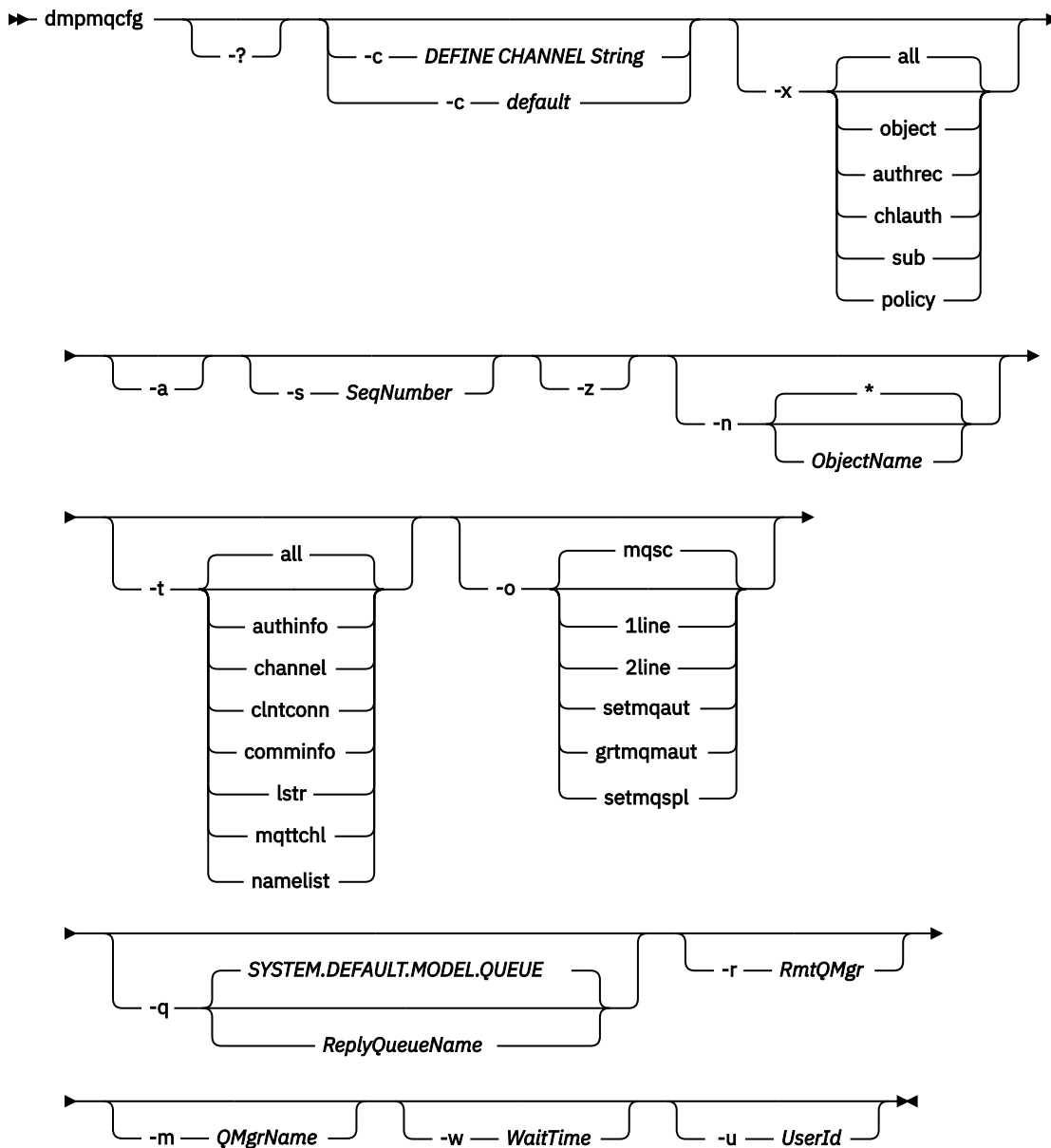
**Note:** The **dmpmqcfcfg** command does not make a backup of the Advanced Message Security policies. If you want to export the Advanced Message Security policies, ensure that you run **dspmqspl** with the **-export** flag. This command exports the policies for Advanced Message Security into a text file, which can be used for restoration purposes. For more information see [“dspmqspl \(display security policy\)” on page 99](#).



**Attention:** The inquiries used by **dmpmqcfcfg** inquire only QSGDISP(QMGR) definitions by default. You can inquire additional definitions by using the environment variable **AMQ\_DMPMQCFEG\_QSGDISP\_DEFAULT**. For more information about the values that you can set with this environment variable, see [AMQ\\_DMPMQCFEG\\_QSGDISP\\_DEFAULT](#).



For example, you could use **AMQ\_DMPMQCFEG\_QSGDISP\_DEFAULT** to query a z/OS queue manager in a queue sharing group from an IBM MQ for Multiplatforms installation. Using the environment variable allows you to include shared objects that would otherwise not be included in the results.



## Optional parameters

**-?**

Inquire the usage message for `dmpmqcfg`.

**-c**

Force a client mode connection. If the **-c** parameter is qualified with the option `default`, the default client connection process is used. If **-c** is omitted, the default is to attempt to connect to the queue manager first by using server bindings and then if this fails by using client bindings.

If the option is qualified with an MQSC `DEFINE CHANNEL CHLTYPE(CLNTCONN)` string then this is parsed and if successful, used to create a temporary connection to the queue manager.

**-x [all|object|authrec|chlauth|sub|policy]**

Filter the definition procedure to show object definitions, authority records, channel authentication records, durable subscriptions or policy. The default value `all` is that all types are returned.

Note that when you specify an export type of `policy`, the security policies for the queue manager are reported in the configuration information dumped.

**-a**

Return object definitions to show all attributes. The default is to return only attributes which differ from the defaults for the object type.

**-s SeqNumber**

Reset channel sequence number for sender, server and cluster sender channel types to the numeric value specified. The value SeqNumber must be in the range 1 - 999999999.

**-z**

Activate silent mode in which warnings, such as those which appear when inquiring attributes from a queue manager of a higher command level are suppressed.

**-n [\*|ObjectName]**

Filter the definitions produced by object or profile name, the object/profile name can contain a single asterisk. The \* option can be placed only at the end of the entered filter string.

@class authority records are included in **dmpmqcfig** output regardless of the object or profile filter specified.



**Attention:** You cannot delete the @CLASS entries (the system is working as designed)

**-t**

Choose a single type of object to export. The following table shows the possible values:

<i>Table 29. Possible values for -t parameter</i>	
<b>Value</b>	<b>Description</b>
all	All object types
authinfo	An authentication information object
channel or chl	A channel
comminfo	A communications information object
lstr or listener	A listener
mqttchl	An MQTT channel
namelist or nl	A namelist
process or prcs	A process
queue or q	A queue
qmgr	A queue manager
svrc or service	A service
topic or top	A topic

**-o [mqsc|1line|2line|setmqaut|grtmqaut|setmqsp1]**

The following table shows the possible values:


<i>Table 30. Possible values for -o parameter options</i>	
<b>Value</b>	<b>Description</b>
mqsc	Multi-line MQSC that can be used as direct input to <b>runmqsc</b>
1line	MQSC with all attributes on a single line for line diffing
2line	MQSC with output on two lines. The first line is an MQSC command string and the second is a commented version with immutable values.
 setmqaut	setmqaut statements for AIX, Linux, and Windows queue managers; valid only when -x authrec is specified.

Table 30. Possible values for <b>-o</b> parameter options (continued)	
Value	Description
Linux grtmqmaut	Linux only; generates iSeries syntax for granting access to the objects.
setmqsp1	<p>The security policies for the queue manager are reported in the format of <b>setmqsp1</b> command lines. This format can be used to generate scripts to restore policy configuration to a queue manager.</p> <p>Note that the <b>setmqsp1</b> command lines produced by this format includes parameters (-m) that specify the queue manager from which the definition was backed up. This implies that the definitions need to be replayed against the same queue manager.</p> <p>If you need to back up policy definitions from one queue manager, and restore them to a different queue manager, consider using the default MQSC format where the queue manager name is not explicitly specified.</p>

**-q**

The name of the reply-to queue used when getting configuration information.

**-r**

The name of the remote queue manager/transmit queue when using queued mode. If this parameter is omitted the configuration for the directly connected queue manager (specified with the **-m** parameter) is dumped.

**-m**

The name of the queue manager to connect to. If omitted the default queue manager name is used.

**-w WaitTime**

The time, in seconds, that **dmpmqcfig** waits for replies to its commands.

Any replies received after a timeout are discarded, but the MQSC commands still run.

The check for timeout is performed once for each command reply.

Specify a time in the range 1 through 999999; the default value is 60 seconds.

Timed-out failure is indicated by:

- Nonzero return code to the calling shell or environment.
- Error message to stdout or stderr.

**-u UserId**

The ID of the user authorized to dump the configuration of queue managers.

### Authorizations

You must have MQZAO\_OUTPUT (+put) authority to access the command input queue (SYSTEM.ADMIN.COMMAND.QUEUE) and MQZAO\_DISPLAY (+dsp) authority to access the default model queue (SYSTEM.DEFAULT.MODEL.QUEUE), to be able to create a temporary dynamic queue if using the default reply queue.

You must also have MQZAO\_CONNECT (+connect) and MQZAO\_INQUIRE (+inq) authority for the queue manager, and MQZAO\_DISPLAY (+dsp) authority for every object that is requested.

No authority is required on the object type (RQMNAME) to limit, or restrict the use of, the **dmpmqcfig** command to display details about any OBJTYPE(RQMNAME).

## Return code

If a failure occurs **dmpmqc fg** returns an error code. Otherwise, the command outputs a footer, an example of which follows:

```
*****
* Script ended on 2016-01-05   at 05.10.09
* Number of Inquiry commands issued: 14
* Number of Inquiry commands completed: 14
* Number of Inquiry responses processed: 273
* QueueManager count: 1
* Queue count: 55
* Namelist count: 3
* Process count: 1
* Channel count: 10
* AuthInfo count: 4
* Listener count: 1
* Service count: 1
* CommInfo count: 1
* Topic count: 5
* Subscription count: 1
* ChlAuthRec count: 3
* Policy count: 1
* AuthRec count: 186
* Number of objects/records: 273
*****
```

## Examples

To make these examples work you need to ensure that your system is set up for remote MQSC operation. See [Configuring queue managers for remote administration](#).

```
dmpmqc fg -m MYQMGR -c "DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(CLNTCONN)
CONNAME('myhost.mycorp.com(1414)')"
```

dumps all the configuration information from remote queue manager *MYQMGR* in MQSC format and creates an ad-hoc client connection to the queue manager using a client channel called *SYSTEM.ADMIN.SVRCONN*.


**Note:** You need to ensure that a server-connection channel with the same name exists.

```
dmpmqc fg -m LOCALQM -x MYQMGR
```

dumps all configuration information from remote queue manager *MYQMGR*, in MQSC format, connects initially to local queue manager *LOCALQM*, and sends inquiry messages through this local queue manager.

**Note:** You need to ensure that the local queue manager has a transmission queue named *MYQMGR*, with channel pairings defined in both directions, to send and receive replies between queue managers.

### Related tasks

 [Backing up queue manager configuration](#)

 [Restoring queue manager configuration](#)

### Related reference

[“runmqsc \(run MQSC commands\)” on page 177](#)

Reference information about the **runmqsc** command prompt, which you can use to issue MQSC commands to a queue manager.

## Multi **dmpmqlog (dump MQ formatted log)**

Display and format a portion of the IBM MQ system log.

### Purpose

Use the **dmpmqlog** command to dump a formatted version of the IBM MQ system log to standard out.

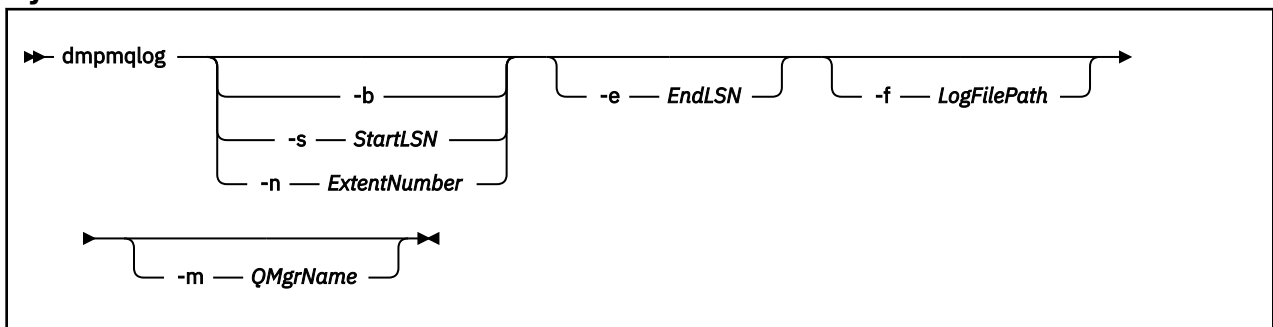
The log to be dumped must have been created on the same type of operating system as that being used to issue the command.

From IBM MQ 9.1.0, the **dmpmqlog** command outputs a timestamp with each log record as shown in the following example:

```
LOG RECORD - LSN <0:0:4615:42406>
*****

HLG Header: lreclsize 212, version 1, rmid 0, eyecatcher HLRH
Creation Time: 2017-01-30 13:50:31.146 GMT Standard Time (UTC +0)
```

### Syntax



### Optional parameters

#### Dump start point

Use one of the following parameters to specify the log sequence number (LSN) at which the dump should start. If you omit this, dumping starts by default from the LSN of the first record in the active portion of the log.

##### **-b**

Start dumping from the base LSN. The base LSN identifies the start of the log extent that contains the start of the active portion of the log.

##### **-s StartLSN**

Start dumping from the specified LSN. The LSN is specified in the format `nnnn:nnnn:nnnn:nnnn`.

If you are using a circular log, the LSN value must be equal to or greater than the base LSN value of the log.

##### **-n ExtentNumber**

Start dumping from the specified extent number. The extent number must be in the range 0 - 9999999.

This parameter is valid only for queue managers using linear logging.

##### **-e EndLSN**

End dumping at the specified LSN. The LSN is specified in the format `nnnn:nnnn:nnnn:nnnn`.



### **-f LogFilePath**

The absolute (rather than relative) directory path name to the log files. The specified directory must contain the log header file (amqh1ct1.lfh) and a subdirectory called active. The active subdirectory must contain the log files. By default, log files are assumed to be in the directories specified in the IBM MQ configuration information. If you use this option, queue names associated with queue identifiers are shown in the dump only if you use the -m option to name a queue manager name that has the object catalog file in its directory path.

On a system that supports long file names this file is called qmqmobjcat and, to map the queue identifiers to queue names, it must be the file used when the log files were created. For example, for a queue manager named qm1, the object catalog file is located in the directory ..\qmgrs\qm1\qmanager\. To achieve this mapping, you might need to create a temporary queue manager, for example named tmpq, replace its object catalog with the one associated with the specific log files, and then start **dmpmqlog**, specifying **-m tmpq** and **-f** with the absolute directory path name to the log files.

### **-m QMgrName**

The name of the queue manager. If you omit this parameter, the name of the default queue manager is used.

**Note:** Do not dump the log while the queue manager is running, and do not start the queue manager while **dmpmqlog** is running.

## **dmpmqmsg (queue load and unload)**

Use the **dmpmqmsg** utility to copy or move the contents of a queue, or its messages, to a file. Formerly the IBM MQ **qload** utility.

### **Purpose**

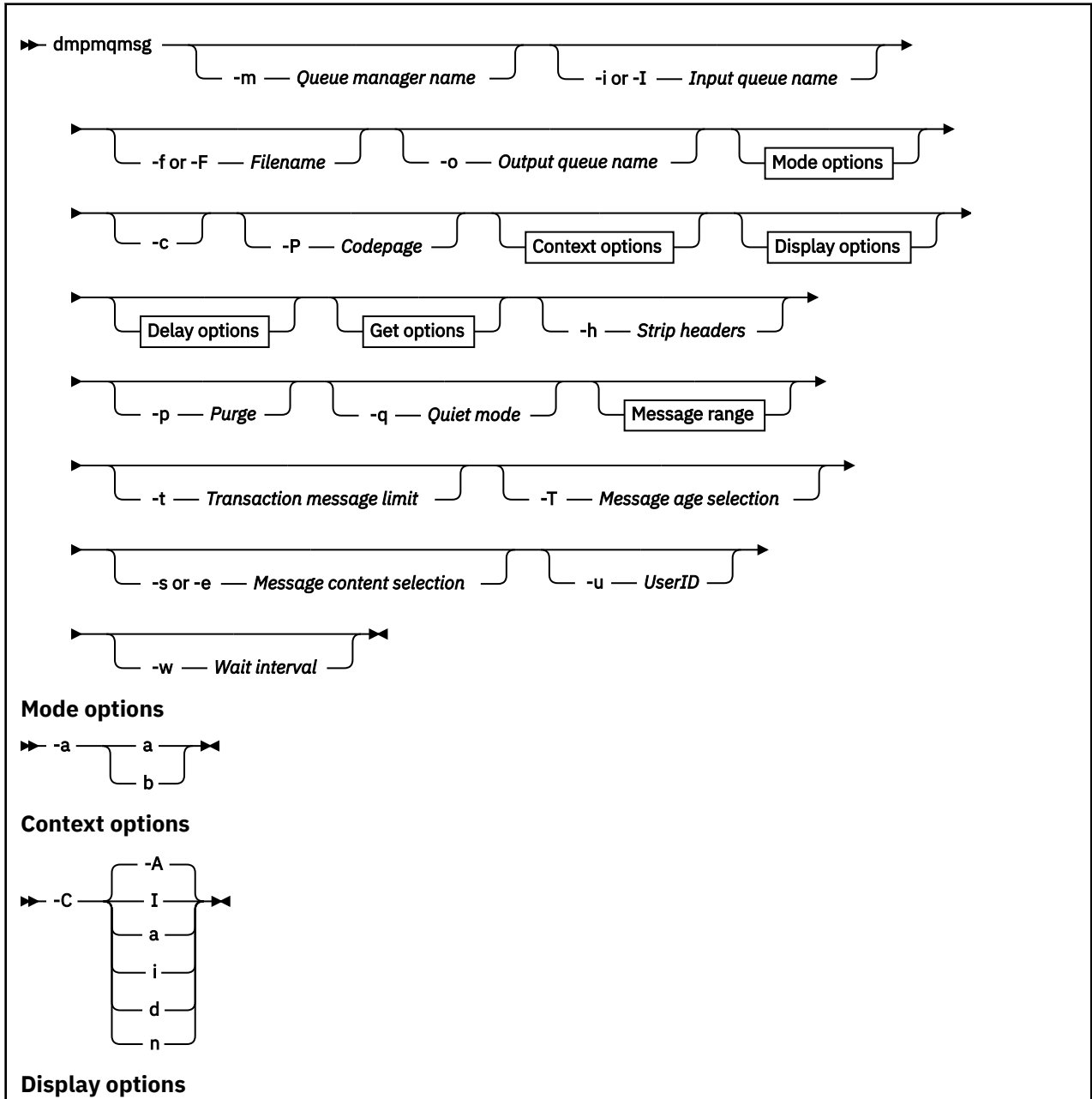
**ALW** From IBM MQ 8.0, the **qload** utility, that previously shipped in IBM MQ Supportpac MO03, has been integrated into IBM MQ as the **dmpmqmsg** utility.

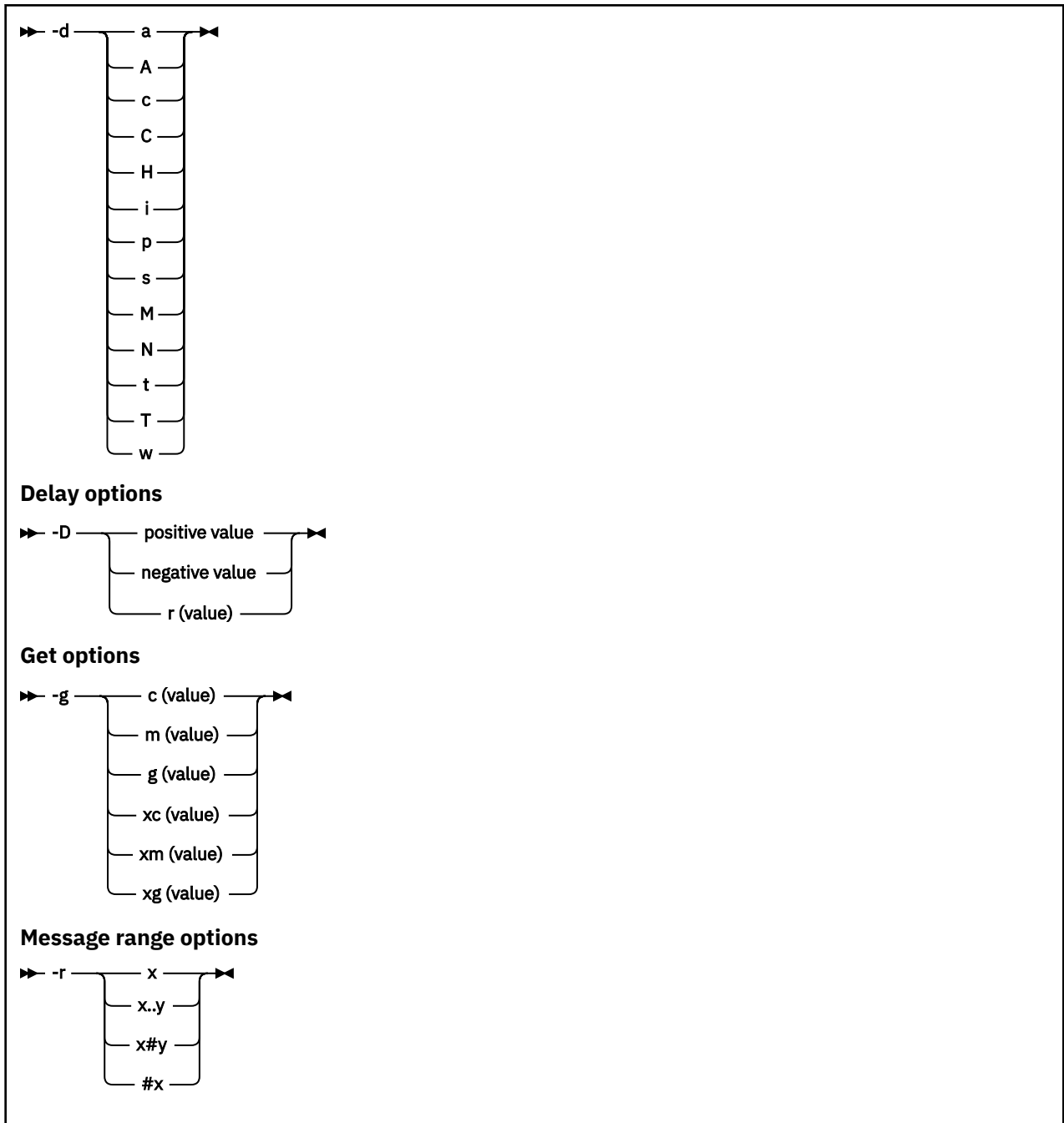
- On AIX and Linux platforms the utility is available in <installdir>./bin.
- On Windows platforms the utility is available in <installdir>./bin64 as part of the server fileset.

For more information, see [Using the dmpmqmsg utility](#).

**z/OS** On z/OS the utility is available as an executable module CSQUDMSG in the SCSQLOAD library, with an alias of QLOAD for compatibility. Sample JCL is provided as member CSQ4QLOD in SCSQPROC. You can also reload messages as described in [“Restoring messages from a data set to a queue \(LOAD\) on z/OS” on page 2767](#) and [“Restoring messages from a data set to a queue \(SLOAD\) on z/OS” on page 2769](#).

## Syntax





## Optional parameters

### **-m** *QueueManagerName*

The name of the queue manager on which the queue, or queues, exist.

### **-i** or **-I** *Input queue name*

The name of the input queue.


**Note:** Using *-i* browses the queue (non-destructive get), whereas using *-I* deletes messages from the queue (destructive get).

### **-f** or **-F** *Filename*

Specifies either the source or target file name.

**Note:**

- Using `-F` on a target file forces output to a file if it already exists. The program does not ask you if the file should be overwritten.
- Take care to ensure that appropriate access controls are set on the output file, as users who are not permitted to access messages on the queue might have access to read the output file.

 On AIX and Linux, permissions for new files are set according to the current `umask` when the utility is run.

 On Windows, permissions for new files are inherited from the parent directory ACL.

**-o Output queue name**

Specifies the name of the output queue.

**-a**

Controls whether the file is opened in append or binary mode, by adding one of the following values to the keyword:

**a**

Append mode


**b**

Binary mode

**-c**

Connect in client mode.

If you do not select this flag, the utility runs in local mode, which is the default.

 This option is not available on z/OS.

**-P**

Controls whether messages taken from a queue are converted.

Use the command

```
-P CCSID [ : X 'Encoding' ]
```

For example `-P850:111`

**-C**

Controls the context option, by adding one of the following values to the keyword:

**A**

Set all context. This is the default value.

**I**

Set identity context.

**a**

Pass all context.

**p**

Pass identity context.

Use of the *pass* options is not applicable if the source messages are browsed on a queue.

**d**

Default context.

**n**

No context.

**-d**

Controls the display option or options, by adding one or more of the following values to the keyword.

For example `-dsCM:`

**a**

Add ASCII columns to the hexadecimal output in the file to aid readability.

**A**

Write ASCII lines of data wherever possible.



On EBCDIC platforms, data is instead written in EBCDIC.

**c**

Output *ApplicationOriginData* and *ApplicationIdentityData* as characters

**C**

Display the *Correlation Identifier* in the queue summary.

**H**

Do not write the file header.

Files created with this option are not loadable by the program as the program does not recognize the file format. However, if necessary you can use an editor to add an appropriate header manually, to make the file loadable.

**i**

Include the message index in the output.

**p**

Printable character output format.

This format is not code page safe. Loading a file written in this format, while running in a new code page does not guarantee to produce the same message.

**s**

Write a simple summary of the messages found on input.

**M**

Display the *Message Identifier* in the queue summary.

**N**

Do not write out the message descriptor content, only the message payload.

**t**

Text line output format.

This format is not code page safe. Loading a file written in this format, while running in a new code page does not guarantee to produce the same message.

**T**

Display the time the message has been on the queue.

**w Length**

Set the data width for the output.

**-D**

Add a delay, expressed in milliseconds, before writing a message to the output destination, by adding one of the following values to the keyword. For example:

**-Dpositive\_value**

Add a fixed delay before putting a message. For example, `-D500` puts each message half a second apart.

**-Dnegative\_value**

Add a random delay, up to the specified value before putting a message. For example, `-D-10000` adds a random delay of up to 10 seconds before putting a message.

**rvalue**

Replays the messages at a percentage of their original put speed. For example:

**r**

Replays messages at their original speed.

**r50**

Replays messages at half their original speed.

**r200**

Replays messages at twice their original speed.

**-g**

Filter by Message identifier, Correlation identifier, or Group identifier, by adding one of the following values to the keyword.

**cvalue**

Get by character Correlation identifier.

**mvalue**

Get by character Message identifier.

**gvalue**

Get by character Group identifier.

**xcvalue**

Get by hexadecimal Correlation identifier.

**xmvalue**

Get by hexadecimal Message identifier.

**xgvalue**

Get by hexadecimal Group identifier.

**-h**

Strip headers.

Any Dead Letter Queue header (MQDLH) or Transmission Queue header (MQXQH) is removed from the message before the message is written.

**-o**

Output queue name.

**-p**

Causes the source queue to be purged of messages as they are copied to the target destination.

**-q**

Sets quiet mode. When set, the program does not output its usual summary of activity.

**-r**

**Note:** If the **dmpmqmsg** command runs with the **-r** option set to 0, the command copies all messages to the destination, whether that destination is a file or queue.

Sets the applicable message range by adding one of the following values to the keyword.

**x**

Only message x, for example, -r10. If r is 0, copies all messages to the destination.

**x..y**

From message x to message y. For example, -r 10..20. -r0..9 copies one to nine messages to the destination.

**x#y**

Output y messages starting at message x. For example, -r 100#10. , -r0#4 copies one to four messages to the destination.

**#x**

Output the first x messages, for example, -r #100. -r \#0 copies all messages to the destination.

**-t**

Set the transaction message limit. If the optional **n** flag is not set, all the messages are done in a single transaction.

**n**

The message operations are split into groups of n messages. For example -t1000 deals with 1000 messages in a single transaction.

## -T

Allows message selection based on message age.

See [“Using message age” on page 71](#) for information on selection using message age.



**Attention:** The age is based on the **PutDate** and **PutTime** fields in the Message Descriptor (MQMD), compared to UTC for the system where the utility is running.

## -s or -e

Allows message selection based on message content.



On ASCII platforms (AIX, Linux, and Windows) use the **-s** option to search for a natively encoded string.



On EBCDIC platforms (z/OS) use the **-e** option to search for a natively encoded string.

See [“Using message content” on page 72](#) for information on selection using message content.

## -u

If you use the **-u** parameter to supply a user ID, you are prompted for a matching password.

If you have configured the CONNAUTH AUTHINFO record with CHCKLOCL(REQUIRED) or CHCKLOCL(REQDADM), you must use the **-u** parameter otherwise you will not be able to copy or move the contents of a queue.

If you specify this parameter and redirect stdin, a prompt will not be displayed and the first line of redirected input should contain the password.

## -w

Wait interval, in seconds, for consuming messages. If specified the program waits for messages to arrive, for the specified period, before ending.

For examples of using the utility, see [Examples of using the \*\*dmpmqmsg\*\* utility](#). If you store the output of the command in a file, see [“Meaning of three letter codes in dmpmqmsg output file” on page 72](#) for the meaning of the codes in the second column of the information in that file.

### Related concepts

[“Restoring messages from a data set to a queue \(LOAD\) on z/OS” on page 2767](#)

The LOAD function of CSQUTIL is complementary to the COPY or SCOPY function. LOAD restores messages from the destination data set of an earlier COPY or SCOPY operation. The queue manager must be running.

[“Restoring messages from a data set to a queue \(SLOAD\) on z/OS” on page 2769](#)

The SLOAD function of CSQUTIL is complementary to the COPY or SCOPY function. SLOAD restores messages from the destination data set of an earlier COPY or SCOPY operation. SLOAD processes a single queue.

### Related reference



[The IBM MQ for z/OS utilities](#)



### **Message selection for *dmpmqmsg***

Message selection for the **dmpmqmsg** command can be based on message age or message content.

## Using message age

You can choose to process only messages older than a certain time interval using the **-T** flag.

Time interval can be specified in Days, Hours and Minutes. The general format being [days:]hours:]minutes.

The parameter can take one or two times, **-T [OlderThanTime] [, YoungerThanTime]**.

For example:

- Display messages older than five minutes

```
dmpmqmsg -m QM1 -i Q1 -fstdout -T5
```

- Display messages younger than five minutes

```
dmpmqmsg -m QM1 -i Q1 -fstdout -T,5
```

- Display messages older than one day but younger than two days.

```
dmpmqmsg -m QM1 -i Q1 -fstdout -T1440,2880
```

- The following command copies messages older than one hour from Q1 to Q2.

```
dmpmqmsg -m QM1 -i Q1 -o Q2 -T1:0
```

- The following command moves messages older than one week from Q1 to Q2

```
dmpmqmsg -m QM1 -I Q1 -o Q2 -T7:0:0
```

## Using message content

You can specify a maximum of three of each search string. If multiple strings are used, they are treated as follows:

### Positive search strings

When multiple positive strings are used, then all the strings must be present for the search to match. For example, the command

```
dmpmqmsg -iMATCH -s LIVERPOOL -s CHELSEA
```

only returns messages that contain both strings.

### Negative search strings

When multiple negative strings are used, then none of the strings must be present for the search to match. For example, the command

```
dmpmqmsg -iMATCH -S HOME -S DRAW
```

only returns messages that contain neither string.

### Multi

## Meaning of three letter codes in dmpmqmsg output file

The mapping between the codes from **dmpmqmsg** and the attribute names from **amqsbcg**.

The order of the attributes in the following table is not alphabetical. Instead, the order reflects the sequence of the attribute names from **amqsbcg**.

File format attribute name (from <b>dmpmqmsg</b> )	Representation (from <b>amqsbcg</b> )
VER	Version
RPT	Report
MST	MsgType
EXP	Expiry
FDB	Feedback



Table 31. Mapping between the three letter codes in the output file from **dmpmqmsg** and the representation from **amqsbcg** (continued)

File format attribute name (from dmpmqmsg)	Representation (from amqsbcg)
ENC	Encoding
CCS	CodedCharSetId
FMT	Format PRI Priority
PER	Persistence
MSI	MsgId
COI	CorrelId
BOC	BackoutCount
RTQ	ReplyToQ
RTM	ReplyToQMgr
USR	UserIdentifier
ACC	AccountingToken
AIX	ApplIdentityData
PAT	PutApplType
PAN	PutApplName
PTD	PutDate
PTT	PutTime
AOX	ApplOriginData
GRP	GroupId
MSQ	MsgSeqNumber
OFF	Offset
MSF	MsgFlags
ORL	OriginalLength

### Related concepts


The Browser sample program

### **dspmq (display queue managers)**

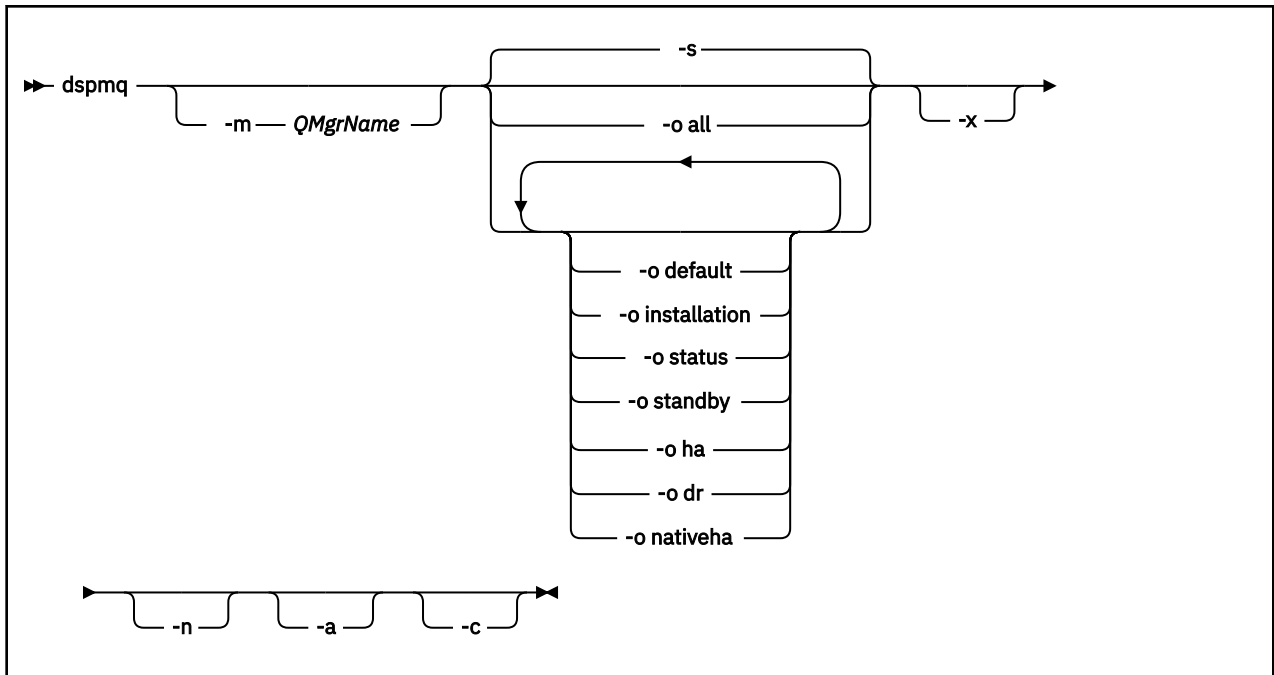
Display information about queue managers on Multiplatforms.

### Purpose

Use the dspmq command to display names and details of the queue managers on a system.

 The equivalent utility to dspmq on z/OS is [CSQUDSPM](#).

### Syntax



## Required parameters

None

## Optional parameters

### -a

Displays information about the active queue managers only.

A queue manager is active if it is associated with the installation from which the **dspsmq** command was issued and one or more of the following statements are true:

- The queue manager is running
- A listener for the queue manager is running
- A process is connected to the queue manager

### -m *QMgrName*

The queue manager for which to display details. If you give no name, all queue manager names are displayed.

### -n

Suppresses translation of output strings.

### -s

The operational status of the queue managers is displayed. This parameter is the default status setting.

The parameter *-o status* is equivalent to *-s*.

### -o **all**

The operational status of the queue managers is displayed, and whether any are the default queue manager.

**ALW** On AIX, Linux, and Windows, the installation name (INSTNAME), installation path (INSTPATH), and installation version (INSTVER) of the installation that the queue manager is associated with is also displayed.

**CP4I** On Native HA configurations, the current HA role (ROLE), the name of this instance (INSTANCE), readiness to take over active role (INSYNC) and quorum status (QUORUM) are also displayed.

**-o default**

Displays whether any of the queue managers are the default queue manager.

**ALW -o installation**

AIX, Linux, and Windows only.

Displays the installation name (INSTNAME), installation path (INSTPATH), and installation version (INSTVER) of the installation that the queue manager is associated with.

**-o status**

The operational status of the queue managers is displayed.

**-o standby**

Displays whether a queue manager currently permits starting a standby instance. The possible values are shown in Table 32 on page 75.

<i>Table 32. Standby values</i>	
<b>Value</b>	<b>Description</b>
Permitted	The queue manager is running and is permitting standby instances.
Not permitted	The queue manager is running and is not permitting standby instances.
Not applicable	The queue manager is not running. You can start the queue manager and this instance becomes active if it starts successfully.

**-o ha | HA**

Indicates whether a queue manager is an HA RDQM (high availability replicated data queue manager) or not. If the queue manager is an HA RDQM, one of the following responses is displayed:

**HA(Replicated)**

Indicates that the queue manager is an HA RDQM.

**HA()**

Indicates that the queue manager is not an HA RDQM.

For example:

```
dspmqr -o ha
QMNAME (RDQM8)           HA(Replicated)
QMNAME (RDQM9)           HA(Replicated)
QMNAME (RDQM7)           HA(Replicated)
QMNAME (QM7)             HA()
```

**-o dr | DR**

Indicates whether a queue manager is a DR RDQM (disaster recovery replicated data queue manager) or not. One of the following responses is displayed:

**DRROLE()**

Indicates that the queue manager is not configured for disaster recovery.

**DRROLE(Primary)**

Indicates that the queue manager is configured as the DR primary.

**DRROLE(Secondary)**

Indicates that the queue manager is configured as the DR secondary.

For example:

```
dspmqr -o dr
QMNAME (RDQM13)          DRROLE(Primary)
QMNAME (RDQM14)          DRROLE(Primary)
```

**CP4I -o nativeha | NATIVEHA**

Displays operational information for an instance in a Native HA configuration. Used on its own, displays ROLE, INSTANCE, INSYNC, and QUORUM fields. Combine with the -x parameter to view additional information on all the instances in the Native HA configuration (see [Native HA instance values](#)).

**-x**

Information about multi-instance queue manager instances is displayed. **CP4I** Displays information about Native HA queue manager instances if combined with the -o nativeha parameter.

The possible values for multi-instance queue manager instances are shown in [Table 33 on page 76](#).

<i>Table 33. Instance values</i>	
Value	Description
Active	The instance is the active instance.
Standby	The instance is a standby instance.

**CP4I** The possible values for Native HA queue manager instances are shown in [Native HA instance values](#)

<i>Table 34. Native HA instance values</i>	
Name	Description
ROLE	Specifies the current role of the instance and is one of Active, Replica, Unknown, or Not configured.
INSTANCE	The name provided for this instance of the queue manager when it was created using the <b>-lr</b> option of the <b>crtmqm</b> command.
INSYNC	Indicates whether the instance is able to take over as the active instance if required.
QUORUM	Reports the quorum status in the form <i>number_of_instances_in-sync/number_of_instances_configured</i> .
REPLADDR	The replication address of the queue manager instance.
CONNECTV	Indicates whether the instance is connected to the active instance.
BACKLOG	Indicates the number of KB that the node is behind.
CONNINST	Indicates whether the named instance is connected to this instance.
ALTDATE	Indicates the date on which this information was last updated (blank if it has never been updated).
ALTTIME	Indicates the time at which this information was last updated (blank if it has never been updated).

For examples of **dspmq** output for Native HA instances, see [Viewing the status of Native HA queue managers for IBM MQ containers](#).

**-c**

Shows the list of processes currently connected to the IPCC, QMGR, and PERSISTENT subpools for a queue manager.

For example, this list typically includes:

- Queue manager processes
- Applications, including those that are inhibiting shutdown
- Listeners

## Queue manager states

The different states that a queue manager can be in are as follows:

- Starting
- Running
- Running as standby
- Running elsewhere
- Quiescing
- Ending immediately
- Ending pre-emptively
- Ended normally
- Ended immediately
- Ended unexpectedly
- Ended pre-emptively
- Status not available

## Return codes

*Table 35. Return code identifiers and descriptions*

<b>Return code</b>	<b>Description</b>
0	Command completed normally
5	Queue manager running
36	Invalid arguments supplied
58	Inconsistent use of installations detected
71	Unexpected error
72	Queue manager name error

## Examples

1. The following command displays queue managers on this server:

```
dspmq -o all
```

2. The following command displays standby information for queue managers on this server that have ended immediately:

```
dspmqr -o standby
```

3. The following command displays standby information and instance information for queue managers on this server:

```
dspmqr -o standby -x
```

## **Multi** dspmqa (display object authorization)

dspmqa displays the authorizations of a specific IBM MQ object.

### Purpose

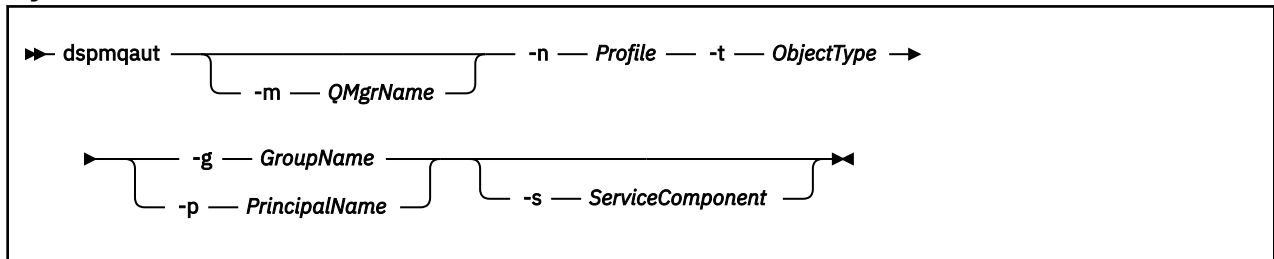
Use the **dspmqa** command to display the current authorizations to a specified object.

If a user ID is a member of more than one group, this command displays the combined authorizations of all the groups.

Only one group or principal can be specified.

For more information about authorization service components, see [Installable services](#), [Service components](#), and [Authorization service interface](#).

### Syntax



### Required parameters

#### **-n Profile**

The name of the profile for which to display authorizations. The authorizations apply to all IBM MQ objects with names that match the profile name specified.

This parameter is required, unless you are displaying the authorizations of a queue manager. In this case you must not include it and instead specify the queue manager name using the **-m** parameter.

#### **-t ObjectType**

The type of object on which to make the inquiry. Possible values are:

Object Type	Description
<b>authinfo</b>	An authentication information object, for use with TLS channel security
<b>channel</b> or <b>chl</b>	A channel
<b>clntconn</b> or <b>clcn</b>	A client connection channel
<b>listener</b> or <b>lstr</b>	A Listener

Table 36. The object type on which to make the inquiry. (continued)

Object Type	Description
<b>namelist</b> or <b>nl</b>	A namelist
<b>process</b> or <b>prcs</b>	A process
<b>queue</b> or <b>q</b>	A queue or queues matching the object name parameter
<b>qmgr</b>	A queue manager
<b>rqmname</b> or <b>rqmn</b>	A remote queue manager name
<b>service</b> or <b>srvc</b>	A service
<b>topic</b> or <b>top</b>	A topic

## Optional parameters

### -m *QMgrName*

The name of the queue manager on which to make the inquiry. This parameter is optional if you are displaying the authorizations of your default queue manager.

### -g *GroupName*

The name of the user group on which to make the inquiry. You can specify only one name, which must be the name of an existing user group.

**Windows** For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following formats:

```
GroupName@domain
domain\GroupName
```

### -p *PrincipalName*

The name of a user for whom to display authorizations to the specified object.

**Windows** For IBM MQ for Windows only, the name of the principal can optionally include a domain name, specified in the following format:

```
userid@domain
```

For more information about including domain names on the name of a principal, see [Principals and groups](#).

### -s *ServiceComponent*

If installable authorization services are supported, specifies the name of the authorization service to which the authorizations apply. This parameter is optional; if you omit it, the authorization inquiry is made to the first installable component for the service.

## Returned parameters

Returns an authorization list, which can contain none, one, or more authorization values. Each authorization value returned means that any user ID in the specified group or principal has the authority to perform the operation defined by that value.

[Table 37 on page 80](#) shows the authorities that can be given to the different object types.

Table 37. Specifying authorities for different object types

Authority	Queue	Process	Queue manager	Remote queue manager name	Namelist	Topic	Auth info	Clntcon n	Channel	Listener	Service
all	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
alladm	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
allmqi	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No
none	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
altusr	No	No	Yes	No	No	No	No	No	No	No	No
browse	Yes	No	No	No	No	No	No	No	No	No	No
chg	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
clr	Yes	No	No	No	No	Yes	No	No	No	No	No
connect	No	No	Yes	No	No	No	No	No	No	No	No
crt	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ctrl	No	No	No	No	No	Yes	No	No	Yes	Yes	Yes
ctrlx	No	No	No	No	No	No	No	No	Yes	No	No
dlt	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
dsp	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
get	Yes	No	No	No	No	No	No	No	No	No	No
pub	No	No	No	No	No	Yes	No	No	No	No	No
put	Yes	No	No	Yes	No	Yes	No	No	No	No	No
inq	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No
passall	Yes	No	No	No	No	Yes	No	No	No	No	No
passid	Yes	No	No	No	No	Yes	No	No	No	No	No
resume	No	No	No	No	No	Yes	No	No	No	No	No
set	Yes	Yes	Yes	No	No	No	No	No	No	No	No
setall	Yes	No	Yes	No	No	Yes	No	No	No	No	No
setid	Yes	No	Yes	No	No	Yes	No	No	No	No	No
sub	No	No	No	No	No	Yes	No	No	No	No	No
system	No	No	Yes	No	No	No	No	No	No	No	No

The following list defines the authorizations associated with each value:

Table 38. Authorizations associated with each value.

Value	Description
all	Use all operations relevant to the object. all authority is equivalent to the union of the authorities alladm, allmqi, and system appropriate to the object type.
alladm	Perform all administration operations relevant to the object
allmqi	Use all MQI calls relevant to the object



Table 38. Authorizations associated with each value. (continued)

Value	Description
altusr	Specify an alternative user ID on an MQI call
browse	Retrieve a message from a queue by issuing an MQGET call with the BROWSE option
chg	Change the attributes of the specified object, using the appropriate command set
clr	Clear a queue (PCF command Clear queue only) or a topic
ctrl	Start, and stop the specified channel, listener, or service, and ping the specified channel.
ctrlx	Reset or resolve the specified channel
connect	Connect the application to the specified queue manager by issuing an MQCONN call
crt	Create objects of the specified type using the appropriate command set
dlt	Delete the specified object using the appropriate command set
dsp	Display the attributes of the specified object using the appropriate command set
get	Retrieve a message from a queue by issuing an MQGET call
inq	Make an inquiry on a specific queue by issuing an MQINQ call
passall	Pass all context
passid	Pass the identity context
pub	Publish a message on a topic using the MQPUT call.
put	Put a message on a specific queue by issuing an MQPUT call
resume	Resume a subscription using the MQSUB call.
set	Set attributes on a queue from the MQI by issuing an MQSET call
setall	Set all context
setid	Set the identity context
sub	Create, alter, or resume a subscription to a topic using the MQSUB call.
system	Use queue manager for internal system operations

The authorizations for administration operations, where supported, apply to these command sets:

- Control commands
- MQSC commands
- PCF commands

## Return codes

Table 39. Return code identifiers and descriptions

Return code	Description
0	Successful operation
26	Queue manager running as a standby instance.
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
58	Inconsistent use of installations detected
69	Storage not available
71	Unexpected error
72	Queue manager name error
133	Unknown object name
145	Unexpected object name
146	Object name missing
147	Object type missing
148	Invalid object type
149	Entity name missing

### Examples

- The following example shows a command to display the authorizations on queue manager `saturn.queue.manager` associated with user group `staff`:

```
dspmqaout -m saturn.queue.manager -t qmgr -g staff
```

The results from this command are:

```
Entity staff has the following authorizations for object:  
  get  
  browse  
  put  
  inq  
  set  
  connect  
  altusr  
  passid  
  passall  
  setid
```

- The following example displays the authorities `user1` has for queue `a.b.c`:

```
dspmqaout -m qmgr1 -n a.b.c -t q -p user1
```

The results from this command are:

```
Entity user1 has the following authorizations for object:
```

get  
put

## Multi **dspmqcsv (display command server)**

The status of a command server is displayed

### Purpose

Use the **dspmqcsv** command to display the status of the command server for the specified queue manager.

The status can be one of the following:

- Starting
- Running
- Running with SYSTEM.ADMIN.COMMAND.QUEUE not enabled for gets
- Ending
- Stopped

You must use the **dspmqcsv** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmq -o installation` command.

### Syntax



### Required parameters

None

### Optional parameters

#### QMgrName

The name of the local queue manager for which the command server status is being requested.

### Return codes

Table 40. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

### Examples

The following command displays the status of the command server associated with `venus.q.mgr`:

```
dspmqcsv venus.q.mgr
```

## Related commands

Table 41. Related command names and descriptions

Command	Description
<code>strmqcsv</code>	Start a command server
<code>endmqcsv</code>	End a command server

### Related reference

[“Command server commands” on page 7](#)

A table of command server commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

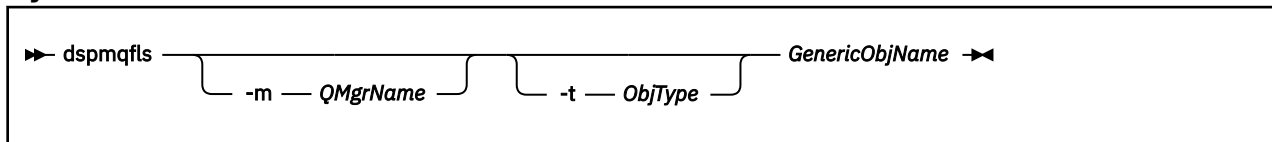
## **Multi** `dspmqls` (display file names)

Display the file names corresponding to IBM MQ objects.

### Purpose

Use the `dspmqls` command to display the real file system name for all IBM MQ objects that match a specified criterion. You can use this command to identify the files associated with a particular object. This command is useful for backing up specific objects. See [Understanding IBM MQ file names](#) for information about name transformation.

### Syntax



### Required parameters

#### GenericObjName

The name of the object. The name is a string with no flag and is a required parameter. Omitting the name returns an error.

This parameter supports an asterisk (\*) as a wildcard at the end of the string.

### Optional parameters

#### -m QMgrName

The name of the queue manager for which to examine files. If you omit this name, the command operates on the default queue manager.



#### -t ObjType

The object type. The following list shows the valid object types. The abbreviated name is shown first followed by the full name.

Table 42. Valid object types.

Object Type	Description
<b>*</b> or <b>all</b>	All object types; this parameter is the default
<b>authinfo</b>	Authentication information object, for use with TLS channel security
<b>channel</b> or <b>chl</b>	A channel
<b>clntconn</b> or <b>clcn</b>	A client connection channel
<b>catalog</b> or <b>ctlg</b>	An object catalog
<b>namelist</b> or <b>nl</b>	A namelist
<b>listener</b> or <b>lstr</b>	A listener
<b>process</b> or <b>prcs</b>	A process
<b>queue</b> or <b>q</b>	A queue or queues matching the object name parameter
<b>qalias</b> or <b>qa</b>	An alias queue
<b>qlocal</b> or <b>ql</b>	A local queue
<b>qmodel</b> or <b>qm</b>	A model queue
<b>qremote</b> or <b>qr</b>	A remote queue
<b>qmgr</b>	A queue manager object
<b>service</b> or <b>srvc</b>	A service

**Note:**

1. The **dspmqls** command displays the name of the directory containing the queue, not the name of the queue itself.
2.   On AIX and Linux, you must prevent the shell from interpreting the meaning of special characters, for example, an asterisk (\*). The way you do this depends on the shell you are using. It may involve the use of single quotation marks, double quotation marks, or a backslash.

**Return codes**

Table 43. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Command completed but not entirely as expected
20	An error occurred during processing

**Examples**

1. The following command displays the details of all objects with names beginning SYSTEM.ADMIN defined on the default queue manager.

```
dspmqls SYSTEM.ADMIN*
```

2. The following command displays file details for all processes with names beginning PROC defined on queue manager RADIUS.

```
dspmqls -m RADIUS -t prcs PROC*
```

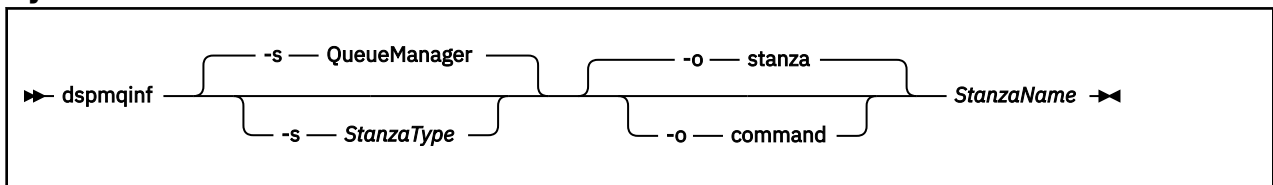
## Multi **dspmqlf (display configuration information)**

Display IBM MQ configuration information (AIX, Linux, and Windows only).

### Purpose

Use the **dspmqlf** command to display IBM MQ configuration information.

### Syntax



### Required parameters

#### StanzaName

The name of the stanza. That is, the value of the key attribute that distinguishes between multiple stanzas of the same type.

### Optional parameters

#### -s *StanzaType*

The type of stanza to display. If omitted, the QueueManager stanza is displayed.

The only supported value of *StanzaType* is QueueManager.

#### -o *stanza*

Displays the configuration information in stanza format as it is shown in the .ini files. This format is the default output format.

Use this format to display stanza information in a format that is easy to read.

#### -o *command*

Displays the configuration information as an **addmqinf** command.

Information about the installation associated with the queue manager is not displayed using this parameter. The **addmqinf** command does not require information about the installation.

Use this format to paste into a command shell.

### Return codes

Table 44. Return code identifiers and descriptions

Return code	Description
0	Successful operation
39	Bad command-line parameters

Table 44. Return code identifiers and descriptions (continued)

Return code	Description
44	Stanza does not exist
58	Inconsistent use of installations detected
69	Storage not available
71	Unexpected error
72	Queue manager name error

### Examples

```
dspmqinf QM.NAME
```

The command defaults to searching for a QueueManager stanza named QM.NAME and displays it in stanza format.

```
QueueManager:
  Name=QM.NAME
  Prefix=/var/mqm
  Directory=QM!NAME
  DataPath=/MQHA/qmgrs/QM!NAME
  InstallationName=Installation1
```

The following command gives the same result:

```
dspmqinf -s QueueManager -o stanza QM.NAME
```

The next example displays the output in **addmqinf** format.

```
dspmqinf -o command QM.NAME
```

The output is on one line:

```
addmqinf -s QueueManager -v Name=QM.NAME -v Prefix=/var/mqm -v Directory=QM!NAME
-v DataPath=/MQHA/qmgrs/QM!NAME
```

### Usage notes

Use **dspmqinf** with **addmqinf** to create an instance of a multi-instance queue manager on a different server.

To use this command you must be an IBM MQ administrator and a member of the mqm group.

### Related commands

Table 45. Related command names and descriptions

Command	Description
<a href="#">“addmqinf (add configuration information)”</a> on page 21	Add queue manager configuration information
<a href="#">“rmvmqinf (remove configuration information)”</a> on page 148	Remove queue manager configuration information

## Multi **dspmqinst (display IBM MQ installation)**

Display installation entries from `mqinst.ini` and license entitlement information on AIX, Linux, and Windows, and license entitlement information on IBM i.

### Purpose

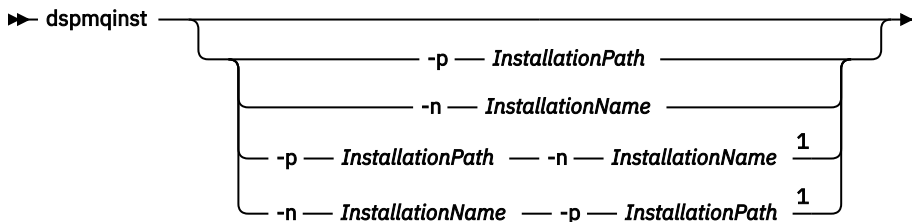
The `mqinst.ini` file contains information about all IBM MQ installations on a system. For more information about `mqinst.ini`, see [Installation configuration file, mqinst.ini](#).

**ULW** On AIX, Linux, and Windows, you can use the **dspmqinst** command to display `mqinst.ini` information from all installations on the system, or about specific installations (see [“Examples for AIX, Linux, and Windows”](#) on page 89).

**dspmqinst** also displays information about license entitlement for each installation. The command displays the license type (Production, Trial, Beta, or Developer) and the licensed entitlement required for the IBM MQ installation. The required entitlement is reported based on the components that are installed and the usage information that has been specified using the **setmqinst** command (see [“setmqinst \(set IBM MQ installation\)”](#) on page 209). For more information about license types and entitlement, see [IBM MQ license information](#).

**IBM i V 9.3.0** From IBM MQ 9.3.0, the **dspmqinst** command is supported on IBM MQ for IBM i without any options to display the license entitlement required for an IBM MQ installation. The required entitlement is reported based on the components that are installed and usage information that has been specified using the **setmqinst** command (see [“Examples for IBM i”](#) on page 90).

### Syntax



Notes:

<sup>1</sup> When specified together, the installation name and installation path must refer to the same installation.

### Required parameters

None

### Optional parameters

**ALW**

#### **-n InstallationName**

The name of the installation.

#### **-p InstallationPath**

The installation path.

**?**

Display usage information.



## Return codes

Table 46. Return code identifiers and descriptions

Return code	Description
0	Entry displayed without error
36	Invalid arguments supplied
44	Entry does not exist
59	Invalid installation specified
71	Unexpected error
89	.ini file error
96	Could not lock .ini file
131	Resource problem

### Examples for AIX, Linux, and Windows



1. Display details of all IBM MQ installations on the system:

```
dspmqinst
```

2. Query the entry for the installation named Installation3:

```
dspmqinst -n Installation3
```

3. Query the entry with an installation path of /opt/mqm:

```
dspmqinst -p /opt/mqm
```

4. Query the entry for the installation named Installation3. Its expected installation path is /opt/mqm:

```
dspmqinst -n Installation3 -p /opt/mqm
```

5. The following examples show the output of **dspmqinst** for different license types and entitlements:

- Output for an IBM MQ client installation:

```
InstName:      Installation1
InstDesc:      My installation
Identifier:    1
InstPath:      /opt/mqm
Version:       9.3.0.0
Primary:       No
State:         Available
License:       Production
Entitlement:    IBM MQ Client
```

- Output for a standard IBM MQ server installation:

```
InstName:      Installation1
InstDesc:      My installation
Identifier:    1
InstPath:      /opt/mqm
Version:       9.3.0.0
Primary:       No
State:         Available
```

```
License:      Production
Entitlement:   IBM MQ
```

- Output for an IBM MQ server installation that has been identified as a High Availability Replica:

```
InstName:     Installation1
InstDesc:     My installation
Identifier:    1
InstPath:     /opt/mqm
Version:      9.3.0.0
Primary:      No
State:        Available
License:      Production
Entitlement:   IBM MQ High Availability Replica
```

- Output for an IBM MQ Advanced Advanced server installation:

```
InstName:     Installation1
InstDesc:     My installation
Identifier:    1
InstPath:     /opt/mqm
Version:      9.3.0.0
Primary:      No
State:        Available
License:      Production
Entitlement:   IBM MQ Advanced
```

- Output for an IBM MQ Advanced server installation that has High Availability Replica entitlement:

```
InstName:     Installation1
InstDesc:     My installation
Identifier:    1
InstPath:     /opt/mqm
Version:      9.3.0.0
Primary:      No
State:        Available
License:      Production
Entitlement:   IBM MQ Advanced High Availability Replica
```

- Output for an IBM MQ Advanced server installation that has Non-Production entitlement:

```
InstName:     Installation1
InstDesc:     My installation
Identifier:    1
InstPath:     /opt/mqm
Version:      9.3.0.0
Primary:      No
State:        Available
License:      Production
Entitlement:   IBM MQ Advanced (Non-production)
```

## Examples for IBM i



From IBM MQ 9.3.0, the **dspmqinst** command is supported to be run without any options. In /QIBM/ProdData/mqm/bin, running **dspmqinst** displays the **InstName**, **InstPath**, **Version**, **LicenseType**, and **Entitlement**.

The following examples shows the output of **dspmqinst** for an IBM MQ Advanced server installation that has been identified as a High Availability Replica:

```
dspmqinst
InstName:     Installation1
InstPath:     /QIBM/ProdData/mqm
Version:      9.3.0.0
LicenseType:   Production
Entitlement:   IBM MQ Advanced High Availability Replica
```

**dspmqlc (display IBM MQ license)**

Display an IBM MQ license.

**Purpose**

On Linux (excluding IBM MQ Appliance) use the **dspmqlc** command to display the IBM MQ license in the appropriate language for the environment.

**Syntax**

```
▶▶ dspmqlc ◀◀
```

**Required parameters**

None

**Optional parameters**

None

**Return codes**

Table 47. Return code identifiers and descriptions

Return code	Description
0	The license file is displayed in some language
20	An error occurred

**Usage notes**

You can change the language can by setting the LANG environment variable. Note that you might need to install the necessary operating system language pack to obtain the required information in a language other than English.

**Related concepts**

[Accepting the license on IBM MQ for Linux](#)

**Related reference****MQLICENSE**

“mqlicense (accept license post installation)” on page 135

Use the mqlicense command on Linux to accept an IBM MQ license after installation.

**strmqm (start queue manager)**

Start a queue manager or ready it for standby operation.

**dspmqrte (display route information)**

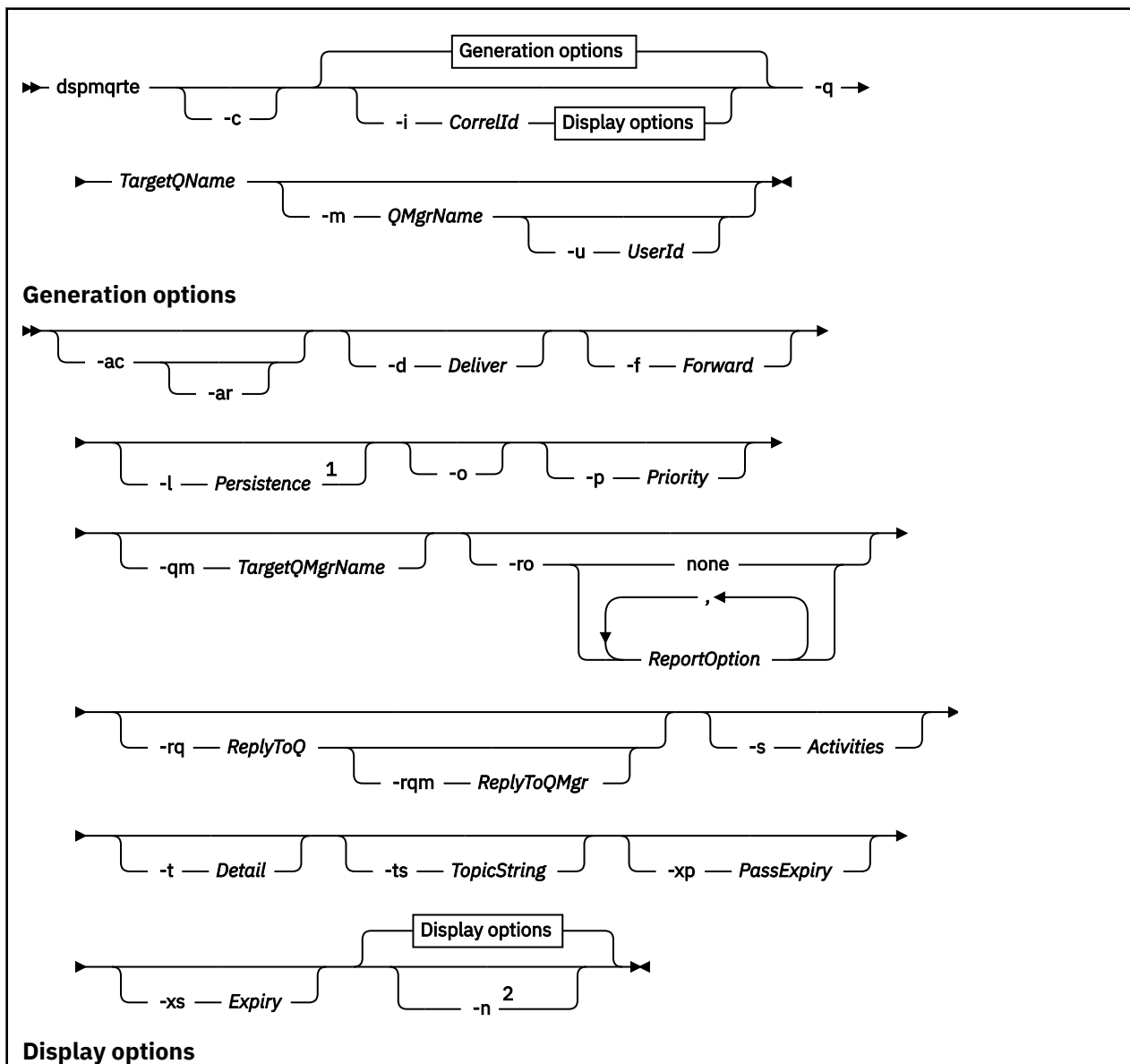
Determine the route that a message has taken through a queue manager network.

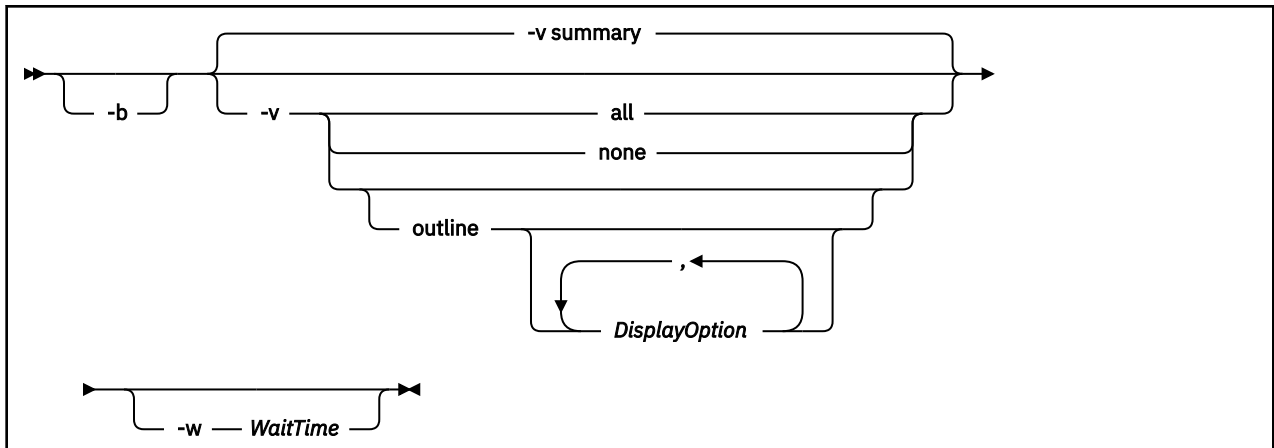
**Purpose**

The IBM MQ display route application (**dspmqrte**) command can be run on all platforms except z/OS. You can run the IBM MQ display route application as a client to an IBM MQ for z/OS queue manager by specifying the **-c** parameter when issuing the **dspmqrte** command.

The IBM MQ display route application generates and puts a trace-route message into a queue manager network. As the trace-route message travels through the queue manager network, activity information is recorded. When the trace-route message reaches its target queue, the activity information is collected by the IBM MQ display route application and displayed. For more information, and examples of using the IBM MQ display route application, see [IBM MQ display route application](#).

## Syntax





**Notes:**

- <sup>1</sup> If *Persistence* is specified as yes, and is accompanied by a request for a trace-route reply message ( *-ar* ), or any report generating options ( *-ro ReportOption* ), then you must specify the parameter *-rq ReplyToQ*. The reply-to queue must not resolve to a temporary dynamic queue.
- <sup>2</sup> If this parameter is accompanied by a request for a trace-route reply message ( *-ar* ), or any of the report generating options ( *-ro ReportOption* ), then a specific (non-model) reply-to queue must be specified using *-rq ReplyToQ* . By default, activity report messages are requested.

**Required parameters**

**-q TargetQName**

If the IBM MQ display route application is being used to send a trace-route message into a queue manager network, *TargetQName* specifies the name of the target queue.

If the IBM MQ display route application is being used to view previously gathered activity information, *TargetQName* specifies the name of the queue where the activity information is stored.

**Optional parameters**

**-c**

Specifies that the IBM MQ display route application connects as a client application. For more information about how to set up client machines, see:

- ▶ **AIX** [Installing an IBM MQ client on an AIX workstation](#)
- ▶ **Linux** [Installing an IBM MQ client on a Linux workstation](#)
- ▶ **Windows** [Installing an IBM MQ client on a Windows workstation](#)
- ▶ **IBM i** [Installing an IBM MQ client on an IBM i workstation](#)

This parameter can be used only if the client component is installed.

**-i CorrelId**

This parameter is used when the IBM MQ display route application is used to display previously accumulated activity information only. There can be many activity reports and trace-route reply messages on the queue specified by *-q TargetQName*. *CorrelId* is used to identify the activity reports, or a trace-route reply message, related to a trace-route message. Specify the message identifier of the original trace-route message in *CorrelId*.

The format of *CorrelId* is a 48 character hexadecimal string.

**-m QMgrName**

The name of the queue manager to which the IBM MQ display route application connects. The name can contain up to 48 characters.

If you do not specify this parameter, the default queue manager is used.

### Generation options

**The following parameters are used when the IBM MQ display route application is used to put a trace-route message into a queue manager network.**

#### **-ac**

Specifies that activity information is to be accumulated within the trace-route message.

If you do not specify this parameter, activity information is not accumulated within the trace-route message.

#### **-ar**

Requests that a trace-route reply message containing all accumulated activity information is generated when the number of activities performed on the trace-route message exceeds the value of specified in *-s Activities*

For more information about trace-route reply messages, see [Trace-route reply message reference](#).

If you do not specify this parameter, a trace-route reply message is not requested.

#### **-d Deliver**

Specifies whether the trace-route message is to be delivered to the target queue on arrival. Possible values for *Deliver* are:

Value	Description
yes	On arrival, the trace-route message is put to the target queue, even if the queue manager does not support trace-route messaging.
no	On arrival, the trace-route message is not put to the target queue.

If you do not specify this parameter, the trace-route message is not put to the target queue.

#### **-f Forward**

Specifies the type of queue manager that the trace-route message can be forwarded to. Queue managers use an algorithm when determining whether to forward a message to a remote queue manager. For details of this algorithm, see [The cluster workload management algorithm](#). The possible values for *Forward* are:

Value	Description
all	The trace-route message is forwarded to any queue manager.
supported	The trace-route message is only forwarded to a queue manager that honors the <i>Deliver</i> parameter from the <i>TraceRoute</i> PCF group.

If you do not specify this parameter, the trace-route message is only forwarded to a queue manager that honors the *Deliver* parameter.

#### **-l Persistence**

Specifies the persistence of the generated trace-route message. Possible values for *Persistence* are:

Value	Description
yes	The generated trace-route message is persistent. (MQPER_PERSISTENT).
no	The generated trace-route message is not persistent. (MQPER_NOT_PERSISTENT).

Table 50. Persistence parameter values. (continued)

Value	Description
q	The generated trace-route message inherits its persistence value from the queue specified by -q <i>TargetQName</i> . (MQPER_PERSISTENCE_AS_Q_DEF).

A trace-route reply message, or any report messages, returned shares the same persistence value as the original trace-route message.

If *Persistence* is specified as yes, you must specify the parameter -r<sub>q</sub> *ReplyToQ*. The reply-to queue must not resolve to a temporary dynamic queue.

If you do not specify this parameter, the generated trace-route message is not persistent.

**-o**

Specifies that the target queue is not bound to a specific destination. Typically this parameter is used when the trace-route message is to be put across a cluster. The target queue is opened with option MQOO\_BIND\_NOT\_FIXED.

If you do not specify this parameter, the target queue is bound to a specific destination.

**-p Priority**

Specifies the priority of the trace-route message. The value of *Priority* is either greater than or equal to 0, or MQPRI\_PRIORITY\_AS\_Q\_DEF. MQPRI\_PRIORITY\_AS\_Q\_DEF specifies that the priority value is taken from the queue specified by -q *TargetQName*.

If you do not specify this parameter, the priority value is taken from the queue specified by -q *TargetQName*.

**-qm TargetQMName**

Qualifies the target queue name; normal queue manager name resolution applies. The target queue is specified with -q *TargetQName*.

If you do not specify this parameter, the queue manager to which the IBM MQ display route application is connected is used as the reply-to queue manager.

**-ro none | ReportOption**

Table 51. ReportOption parameter values.

Value	Description
none	Specifies no report options are set.
<i>ReportOption</i>	Specifies report options for the trace-route message. Multiple report options can be specified using a comma as a separator. Possible values for <i>ReportOption</i> are: <b>activity</b> The report option MQRO_ACTIVITY is set. <b>coa</b> The report option MQRO_COA_WITH_FULL_DATA is set. <b>cod</b> The report option MQRO_COD_WITH_FULL_DATA is set. <b>exception</b> The report option MQRO_EXCEPTION_WITH_FULL_DATA is set. <b>expiration</b> The report option MQRO_EXPIRATION_WITH_FULL_DATA is set. <b>discard</b> The report option MQRO_DISCARD_MSG is set.

If -r<sub>o</sub> *ReportOption* or -r<sub>o</sub> none are not specified, then the MQRO\_ACTIVITY and MQRO\_DISCARD\_MSG report options are specified.

**-rq ReplyToQ**

Specifies the name of the reply-to queue that all responses to the trace-route message are sent to. If the trace-route message is persistent, or if the **-n** parameter is specified, a reply-to queue must be specified that is not a temporary dynamic queue.

If you do not specify this parameter, the system default model queue, SYSTEM.DEFAULT.MODEL.QUEUE is used as the reply-to queue. Using this model queue causes a temporary dynamic queue, for the IBM MQ display route application, to be created.

**-rqm ReplyToQMgr**

Specifies the name of the queue manager where the reply-to queue is located. The name can contain up to 48 characters.

If you do not specify this parameter, the queue manager to which the IBM MQ display route application is connected is used as the reply-to queue manager.

**-s Activities**

Specifies the maximum number of recorded activities that can be performed on behalf of the trace-route message before it is discarded. This parameter prevents the trace-route message from being forwarded indefinitely if caught in an infinite loop. The value of *Activities* is either greater than or equal to 1, or MQROUTE\_UNLIMITED\_ACTIVITIES. MQROUTE\_UNLIMITED\_ACTIVITIES specifies that an unlimited number of activities can be performed on behalf of the trace-route message.

If you do not specify this parameter, an unlimited number of activities can be performed on behalf of the trace-route message.

**-t Detail**

Specifies the activities that are recorded. The possible values for *Detail* are:

<i>Table 52. Detail parameter values.</i>	
Value	Description
low	Activities performed by user-defined application are recorded only.
medium	Activities specified in low are recorded. Additionally, activities performed by MCAs are recorded.
high	Activities specified in low, and medium are recorded. MCAs do not expose any further activity information at this level of detail. This option is available to user-defined applications that are to expose further activity information only. For example, if a user-defined application determines the route a message takes by considering certain message characteristics, the routing logic can be included with this level of detail.

If you do not specify this parameter, medium level activities are recorded.

**-ts TopicString**

Specifies a topic string to which the IBM MQ display route application is to publish a trace-route message, and puts this application into topic mode. In this mode, the application traces all of the messages that result from the publish request.

**-xp PassExpiry**

Specifies whether the report option MQRO\_DISCARD\_MSG and the remaining expiry time from the trace-route message is passed on to the trace-route reply message. Possible values for *PassExpiry* are:



Table 53. PassExpiry parameter values.

Value	Description
yes	<p>The report option MQRO_PASS_DISCARD_AND_EXPIRY is specified in the message descriptor of the trace-route message.</p> <p>If a trace-route reply message, or activity reports, are generated for the trace-route message, the MQRO_DISCARD_MSG report option (if specified), and the remaining expiry time are passed on.</p> <p>This parameter is the default value.</p>
no	<p>The report option MQRO_PASS_DISCARD_AND_EXPIRY is not specified.</p> <p>If a trace-route reply message is generated for the trace-route message, the discard option and remaining expiry time from the trace-route message are not passed on.</p>

If you do not specify this parameter, the MQRO\_PASS\_DISCARD\_AND\_EXPIRY report option is not specified in the trace-route message.

**-xs Expiry**

Specifies the expiry time for the trace-route message, in seconds.

If you do not specify this parameter, the expiry time is specified as 60 seconds.

**-n**

Specifies that activity information returned for the trace-route message is not to be displayed.

If this parameter is accompanied by a request for a trace-route reply message ( -a1), or any of the report generating options from ( -10 ReportOption), then a specific (non-model) reply-to queue must be specified using -1q ReplyToQ. By default, activity report messages are requested.

After the trace-route message is put to the specified target queue, a 48 character hexadecimal string is returned containing the message identifier of the trace-route message. The message identifier can be used by the IBM MQ display route application to display the activity information for the trace-route message at a later time. This can be done using the -i CorrelId parameter.

If you do not specify this parameter, activity information returned for the trace-route message is displayed in the form specified by the -v parameter.

**Display options**

**The following parameters are used when the IBM MQ display route application is used to display collected activity information.**

**-b**

Specifies that the IBM MQ display route application only browses activity reports or a trace-route reply message related to a message. This parameter allows activity information to be displayed again at a later time.

If you do not specify this parameter, the IBM MQ display route application gets activity reports and deletes them, or a trace-route reply message related to a message.

**-v summary | all | none | outline DisplayOption**

Table 54. DisplayOption parameter values.

Value	Description
summary	The queues that the trace-route message was routed through are displayed.
all	All available information is displayed.
none	No information is displayed.

Table 54. *DisplayOption* parameter values. (continued)

Value	Description
outline <i>DisplayOption</i>	<p>Specifies display options for the trace-route message. Multiple display options can be specified using a comma as a separator.</p> <p>If no values are supplied the subsequent information is displayed:</p> <ul style="list-style-type: none"> <li>• The application name</li> <li>• The type of each operation</li> <li>• Any operation-specific parameters</li> </ul> <p>Possible values for <i>DisplayOption</i> are:</p> <p><b>activity</b> All non-PCF group parameters in <i>Activity</i> PCF groups are displayed.</p> <p><b>identifiers</b> Values with parameter identifiers MQBACF_MSG_ID or MQBACF_CORREL_ID are displayed. This overrides <i>msgdelta</i>.</p> <p><b>message</b> All non-PCF group parameters in <i>Message</i> PCF groups are displayed. When this value is specified, you cannot specify <i>msgdelta</i>.</p> <p><b>msgdelta</b> All non-PCF group parameters in <i>Message</i> PCF groups, that have changed since the last operation, are displayed. When this value is specified, you cannot specify <i>message</i>.</p> <p><b>operation</b> All non-PCF group parameters in <i>Operation</i> PCF groups are displayed.</p> <p><b>traceroute</b> All non-PCF group parameters in <i>TraceRoute</i> PCF groups are displayed.</p>

If you do not specify this parameter, a summary of the message route is displayed.

**-w WaitTime**

Specifies the time, in seconds, that the IBM MQ display route application waits for activity reports, or a trace-route reply message, to return to the specified reply-to queue.

If you do not specify this parameter, the wait time is specified as the expiry time of the trace-route message, plus 60 seconds.

**-u UserId**

The ID of the user authorized to determine the route that a message has taken through a queue manager network.

**Return codes**


Table 55. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Invalid arguments supplied
20	An error occurred during processing

**Examples**



Table 56. `dspmqspl` command flags (continued)

Command flag	Explanation
<b>-export</b>	Adding this flag generates output which can easily be applied to a different queue manager.  The output is written to a DD named EXPORT

## Examples

The `dspmqspl` command shows the key reuse count for all policies. The following example is the output you receive on [Multiplatforms](#):

```
Policy Details:
Policy name: PROT
Quality of protection: PRIVACY
Signature algorithm: SHA256
Encryption algorithm: AES256
Signer DNs: -
Recipient DNs:
  CN=Name, O=Organization, C=Country
Toleration: 0
Key Reuse Count: 0
-----
Policy Details:
Policy name: PROT2
Quality of protection: CONFIDENTIALITY
Signature algorithm: NONE
Encryption algorithm: AES256
Signer DNs: -
Recipient DNs:
  CN=Name, O=Organization, C=Country
Toleration: 0
Key Reuse Count: 100
```

### Related reference


[“SET POLICY \(set security policy\) on Multiplatforms” on page 938](#)

Use the MQSC command SET POLICY to set a security policy.

[“DISPLAY POLICY \(display a security policy\) on Multiplatforms” on page 755](#)

Use the MQSC command **DISPLAY POLICY** to display a security policy.

[“setmqspl \(set security policy\)” on page 216](#)

Use the `setmqspl` command to define a new security policy, replace an already existing one, or remove an existing policy.  On z/OS you use the command with the CSQOUTIL utility.

## **dspmqttrc (display formatted trace)**

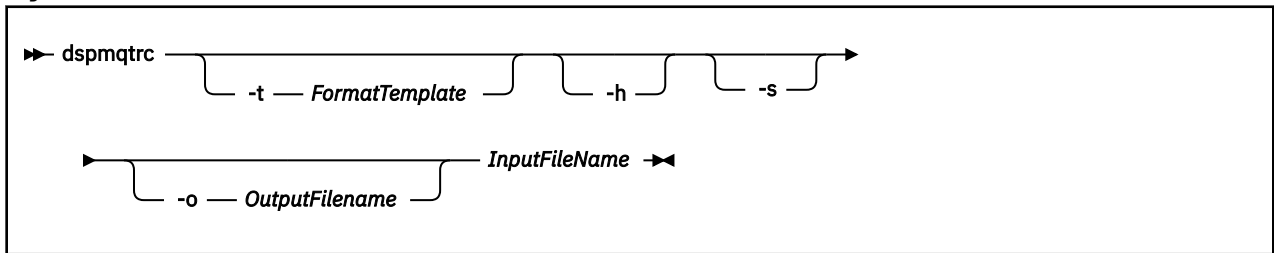
Format and display IBM MQ trace.

### Purpose

The `dspmqttrc` command is supported on AIX and Linux systems only. Use the `dspmqttrc` command to display IBM MQ formatted trace output.

The runtime TLS trace files have the names AMQ.SSL.TRC and AMQ.SSL.TRC.1. You cannot format any of the TLS trace files. The TLS trace files are binary files and, if they are transferred to IBM support by FTP, they must be transferred in binary transfer mode.

## Syntax



## Required parameters

### InputFileName

The name of the file containing the unformatted trace, for example:

```
/var/mqm/trace/AMQ12345.01.TRC
```

If you provide one input file, **dspmqtrc** formats it to the output file you name. If you provide more than one input file, any output file you name is ignored, and formatted files are named AMQ *yyyyy.zz.FMT*, based on the PID of the trace file.

## Optional parameters

### -t *FormatTemplate*

The name of the template file containing details of how to display the trace. If this parameter is not supplied, the default template file location is used:

**AIX** For AIX systems, the default value is as follows:

```
MQ_INSTALLATION_PATH/lib/amqtrc2.fmt
```

**Linux** For Linux, the default value is as follows:

```
MQ_INSTALLATION_PATH/lib/amqtrc.fmt
```

*MQ\_INSTALLATION\_PATH* represents the high-level directory in which IBM MQ is installed.

### -h

Omit header information from the report.

### -s

Extract trace header and put to stdout.

### -o *output\_filename*

The name of the file into which to write formatted data.

## Related commands

Table 57. Related command names and descriptions

Command	Description
<a href="#">endmqtrc</a>	End trace
<a href="#">“strmqtrc (start trace)” on page 242</a>	Start trace

### Related tasks

[Using trace](#)

## Related reference

Command sets comparison: Other commands

A table of other commands, showing the command description, and the equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

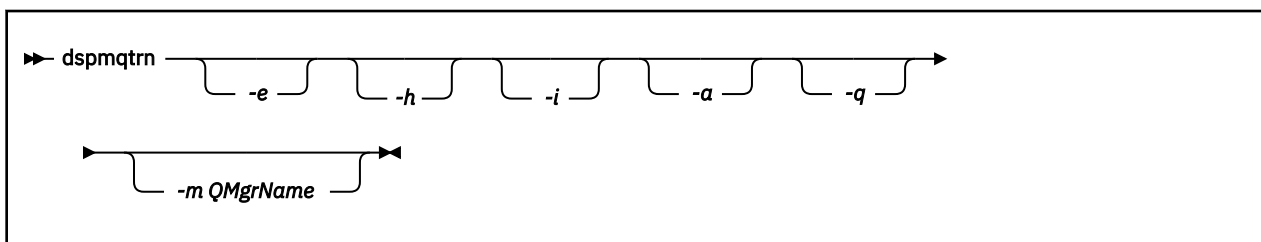
## Multi **dspmqtrn (display incomplete transactions)**

Display in-doubt and heuristically completed transactions.

## Purpose

Use the **dspmqtrn** command to display details of transactions. This command includes transactions coordinated by IBM MQ and by an external transaction manager.

## Syntax



## Optional parameters

**-e**

Requests details of externally coordinated, in-doubt XA transactions. Such transactions are those for which the queue manager (RM) has been asked to prepare to commit, but has not yet been informed by the TM of the transaction outcome (commit or rollback).

**-h**

Requests details of externally coordinated transactions that were resolved by the **rsvmqtrn** command, and the external transaction coordinator has yet to acknowledge with an `xa-forget` command. This transaction state is termed *heuristically completed* by X/Open.

**Note:** If you do not specify **-e**, **-h**, or **-i**, details of both internally and externally coordinated in-doubt transactions are displayed, but details of externally coordinated, heuristically completed transactions are not displayed.

**-i**

Requests details of internally coordinated, in-doubt XA transactions. Such transactions are those for which the queue manager (TM) has asked each resource manager (RM) to prepare to commit, but an error was reported by one of the resource managers (for example, a network connection broke). In this state, the queue manager (TM) has yet to inform all resource managers of the transaction outcome (commit or rollback), but stands ready to do so. For more information, see [Displaying outstanding units of work with the dspmqtrn command](#).

Information about the state of the transaction in each of its participating resource managers is displayed. This information can help you assess the affects of failure in a particular resource manager.

**Note:** If you do not specify **-e** or **-i**, details of both internally and externally coordinated in-doubt transactions are displayed.

**-a**

Requests a list of all transactions known to the queue manager. The returned data includes transaction details for all transactions known to the queue manager. If a transaction is currently

associated with an IBM MQ application connection, information related to that IBM MQ application connection is also returned. The data returned by this command might typically be correlated with the output of a `runmqsc "DISPLAY CONN (display application connection information)"` on page 725 command, and the output fields have the same meaning as in that command.

Not all of the fields are appropriate for all transactions. When the fields are not meaningful, they are displayed as blank. For example: The UOWLOG value when the command is issued against a circular logging queue manager.

#### **-q**

Specifying this parameter on its own is the same as specifying `-a -q`.

Displays all the data from the `-a` parameter and a list of up to 100 unique objects updated within the transaction. If more than 100 objects are updated in the same transaction, only the first 100 distinct objects are listed for each transaction.

#### **-m QMgrName**

The name of the queue manager for which to display transactions. If you omit the name, the transaction of the default queue manager are displayed.

## Return codes

Table 58. Return code identifiers and descriptions

Return code	Description
0	Successful operation
26	Queue manager running as a standby instance.
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
58	Inconsistent use of installations detected
69	Storage not available
71	Unexpected error
72	Queue manager name error
102	No transactions found

## Example

A typical use of the command is:

```
dspmqrtn -m QMgrName -q -a
```

## Related commands

Table 59. Related command names and descriptions

Command	Description
<a href="#">rsvmqtrn</a>	Resolve transaction

## Related information

[Displaying outstanding units of work with the dspmqrtn command](#)

## Multi **dspmqr (display version information)**

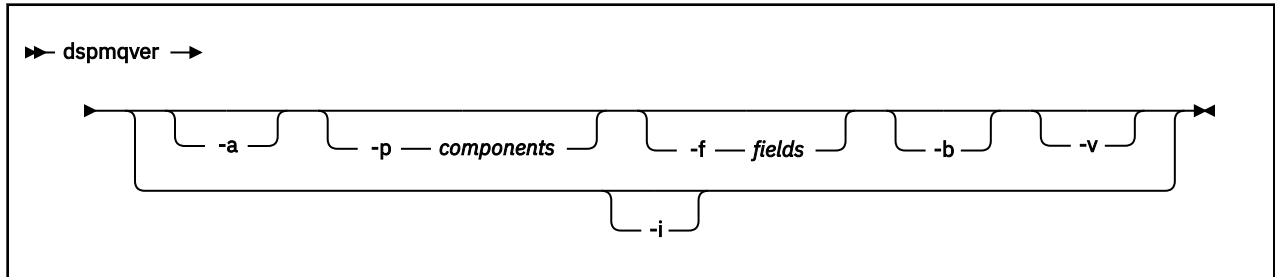
Display IBM MQ version and build information.

### Purpose

Use the **dspmqr** command to display IBM MQ version and build information.

By default, the **dspmqr** command displays details of the installation from which it was started. A note is displayed if other installations exist; use the **-i** parameter to display their details.






### Syntax



### Optional parameters

**-a**  
Display information about all fields and components.

**-p Components**  
Display information for the components specified by *component*. Either a single component or multiple components can be specified. Enter either the value of a single component or the sum of the values of all the required components. Available components and related values are as follows:

Value	Description
1	IBM MQ server, or client.
2	IBM MQ classes for Java.
4	IBM MQ classes for Java Message Service and    IBM MQ classes for Jakarta Messaging.
8	WebScale Distribution Hub
16 <small>"1" on page 105</small>	 IBM MQ custom channel for Windows Communication Foundation.
32	 IBM MQ Message Service Client (XMS) for .NET (XMS .NET) - this component is only available on Windows.
64	GSKit For IBM MQ 9.1 and earlier, when 32-bit support is missing from a 64-bit system, the <b>dspmqr -a</b> command might issue a message suggesting that the 32 bit version of GSKit is not installed. For more information, see the "Command failure" section of this topic. This issue is fixed in IBM MQ 9.2.
128	Advanced Message Security
256	IBM MQ AMQP Service
512	IBM MQ Telemetry Service



Value	Description
1024	Other bundled components that are used by IBM MQ
2048	WebSphere Liberty profile
4096	IBM MQ Java runtime environment
8192	IBM MQ Replicated Data Queue Managers (RDQM)

**Notes:**

1. **Windows** Supported by IBM MQ for Windows only. If you have not installed Microsoft.NET 3 or later, the following error message is displayed:

Title: WMQWCFCustomChannelLevel.exe - Application Error

The application failed to initialize properly (0x0000135).

The default value is 1.

**-f Fields**

Display information for the fields specified by *field*. Specify either a single field or multiple fields. Enter either the value of a single field or the sum of the values of all the required fields. Available fields and related values are as follows:

Value	Description
1	Name
2	Version, in the form V . R . M . F: Where V =Version, R =Release, M =Modification, and F =Fix pack
4	Level
8	Build type
16	Platform
32	Addressing mode
64	Operating system
128	Installation path
256	Installation description
512	Installation name
1024	Maximum command level
2048	Primary installation
4096	Data Path
8192	License type
<b>V9.3.3</b> 16384	Release type, can be: <ul style="list-style-type: none"> <li>• Long Term Support (LTS) and Continuous Delivery (CD) if the Modification and Fix pack is zero</li> <li>• Long Term Support (LTS) if the Modification is zero and Fix pack is not zero.</li> <li>• Continuous Delivery (CD) otherwise</li> </ul>

Value	Description
	See IBM MQ release types and versioning.

Information for each selected field is displayed on a separate line when the **dspmqr** command is run.

The default value is 65535. This displays information for all fields.

**-b**

Omit header information from the report.

**-v**

Display verbose output.

**-i**

Display information about all installations. You cannot use this option with other options. The installation from which the **dspmqr** command was issued is displayed first. For any other installations, only the following fields are displayed: Name, Version, Installation name, Installation description, Installation path, and Primary installation.

## Return codes

Table 60. Return code identifiers and descriptions

Return code	Description
0	Command completed normally.
10	Command completed with unexpected results.
20	An error occurred during processing.

## Examples

The following command displays IBM MQ version and build information, using the default settings for **-p** and **-f**:

```
dspmqr
```

The following command displays information about all fields and components and is the equivalent of specifying `dspmqr -p 63 -f 4095`:

```
dspmqr -a
```

The following command displays version and build information for the IBM MQ classes for Java:

```
dspmqr -p 2
```

The following command displays the Common Services for Java Platform Standard Edition, IBM MQ, Java Message Service Client, IBM MQ classes for Java Message Service, and **JM 3.0** IBM MQ classes for Jakarta Messaging:

```
dspmqr -p 4
```

The following command displays the build level of the WebScale Distribution Hub:

```
dspmqr -p 8 -f 4
```

**Windows** The following command displays the name and build type for IBM MQ custom channel for Windows Communication Foundation:

```
dspmqr -p 16 -f 9
```

The following command displays information about installations of IBM MQ.

```
dspmqr -i
```

Example output for MQ.NET Standard classes:

```
Name:      IBM Message Service Client for .NET Standard
Version:   9.1.1.0
Level:    p911-LXXXX
Build Type: Production
```

## Command failure

### *Failure when 32-bit support is missing from a 64-bit system*

In IBM MQ 8.0, IBM MQ 9.0, and IBM MQ 9.1, the 32-bit and 64-bit versions of IBM Global Security Kit for IBM MQ (GSKit) are bundled together. When you run `dspmqr -a` or `dspmqr -p 64`, the command checks both versions of GSKit. When 32-bit support is missing from a 64-bit system, you might get a message suggesting that the 32-bit version of GSKit is not installed. The GSKit issue is fixed in IBM MQ 9.2. For information about 64-bit Linux distributions that might no longer support 32-bit applications by default, and guidance on manually loading the 32-bit libraries for these platforms, see [Hardware and software requirements on Linux systems](#).

### *Failure when viewing the IBM MQ classes for Java*

The **dspmqr** command can fail if you try to view version or build information for the IBM MQ classes for Java, and you have not correctly configured your environment, or if the IBM MQ JRE component is not installed, and an alternative JRE could not be located.

For example, you might see the following message:

```
[root@blade883 ~]# dspmqr -p 2
AMQ8351: IBM MQ Java environment has not been configured
correctly, or the IBM MQ JRE feature has not been installed.
```

To resolve this problem, consider installing the IBM MQ JRE component if it is not already installed, or ensure that the path is configured to include the JRE, and that the correct environment variables are set.

**Linux** **AIX** For example, on AIX and Linux, you can use one of the following scripts to resolve this problem:

- **JMS 2.0** For JMS 2.0, use `setjmsenv` or `setjmsenv64`. For example:

```
export PATH=$PATH:/opt/mqm/java/jre/bin
cd /opt/mqm/java/bin/
./setjmsenv64

[root@blade883 bin]# dspmqr -p 2
Name:      IBM MQ classes for Java
Version:   8.0.0.0
Level:    k000-L110908
Build Type: Production
```

- ▶ **JM 3.0** For Jakarta Messaging 3.0, use `setjms30env` or `setjms30env64`. For example:

```
export PATH=$PATH:/opt/mqm/java/jre/bin
cd /opt/mqm/java/bin/
. ./setjms30env64

[root@blade883 bin]# dspmqver -p 2
Name:      IBM MQ classes for Java
Version:   8.0.0.0
Level:    k000-L110908
Build Type: Production
```

#### Notes:

- ▶ **Linux** ▶ **AIX** The `setjmsenv` and `setjms30env` scripts apply to AIX and Linux only.
- ▶ **Windows** On Windows, the `runjms` scripts provide similar functionality to that provided by the `setjmsenv` and `setjms30env` scripts on AIX and Linux.
- ▶ **Windows** For the IBM MQ classes for Java on Windows, if the IBM MQ JRE component is installed, you need to issue the `setmqenv` command to resolve error [AMQ8351](#).

#### Related tasks

[Setting environment variables for IBM MQ classes for JMS](#)

#### Related reference

[Scripts provided with IBM MQ classes for JMS](#)

## dspmqweb properties (display mqweb server configuration properties)

Display information about the configurable properties of the mqweb server that is used to support the IBM MQ Console and REST API. That is, the command displays properties that are configurable by the user and the properties that have been modified.

### Purpose

Use the `dspmqweb properties` command to view details of the configuration of the mqweb server. It is not necessary for the mqweb server to be running.

### Using the command on z/OS

▶ **z/OS**

Before you issue either the `setmqweb` or `dspmqweb` commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, enter the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where `WLP_user_directory` is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

You must also set the `JAVA_HOME` environment variable to reference a 64-bit version of Java on your system.

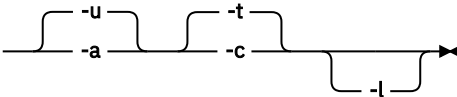
### Using the command in a stand-alone IBM MQ Web Server installation

▶ **V 9.3.5** ▶ **Linux**

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ\_OVERRIDE\_DATA\_PATH** environment variable to the IBM MQ Web Server data directory.

The user ID running the command needs write access to the data directory and its subdirectories.

## Syntax

► **dspmqweb properties** 

## Optional parameters

- u**  
Displays only the configurable properties that have been modified by the user.
- a**  
Displays all available configurable properties, including those properties which have been modified by the user.
- t**  
Formats the output as text name-value pairs.
- c**  
Formats the output as command text, which can be used as input to the corresponding **setmqweb properties** command.
- l**  
Enable verbose logging. Diagnostic information is written to a mqweb server log-file.

## Properties that can be returned by the command

The following properties can be returned by the **dspmqweb properties** command on all platforms, including the IBM MQ Appliance. **V 9.3.5** Some of the following properties are not available in a stand-alone IBM MQ Web Server installation.

### ltpaExpiration

This configuration property is used to specify the time, in seconds, before the LTPA token expires.

### maxTraceFiles

This configuration property is used to specify the maximum number of mqweb server log files that are generated by the mqweb server.

### maxTraceFileSize

This configuration property is used to specify the maximum size, in MB, that each mqweb server log file can reach.

### **V 9.3.4** mqConsoleEnableDashboardBrowse

This configuration property is used to enable or disable queue browsing that is used by the IBM MQ Console to obtain some of the information that is displayed in the [MQ Console dashboard](#).

### **V 9.3.4** mqConsoleEnableSystemTopicMonitoring

This configuration property is used to enable or disable system topic monitoring that is used to display system information in the IBM MQ Console.

### **V 9.3.0** mqConsoleMaxMsgCharsToDisplay

This configuration property is used to specify the maximum characters to retrieve from each message when you browse a queue by using the IBM MQ Console.

### **V 9.3.0** `mqConsoleMaxMsgRequestSize`

This configuration property is used to specify the maximum size, in MB, a browse request can be across all messages when you browse queues by using the IBM MQ Console.

### **V 9.3.0** `mqConsoleMaxMsgsPerRequest`

This configuration property is used to specify the total number of messages to retrieve from a queue when you browse by using the IBM MQ Console.

### **mqRestCorsAllowedOrigins**

This configuration property is used to specify the origins that are allowed to access the REST API. For more information about CORS, see [Configuring CORS for the REST API](#).

### **mqRestCorsMaxAgeInSeconds**

This configuration property is used to specify the time, in seconds, that a web browser can cache the results of any CORS pre-flight checks.

### **mqRestCsrftValidation**

This configuration property is used to specify whether CSRF validation checks are performed. A value of `false` removes the CSRF token validation checks.

### **mqRestGatewayEnabled**

This configuration property is used to specify whether the administrative REST API gateway is enabled.

**V 9.3.5** This property is not returned in a stand-alone IBM MQ Web Server installation as the administrative REST API is not available in this environment.

### **mqRestGatewayQmgr**

This configuration property is used to specify the name of the queue manager to use as the gateway queue manager. This queue manager must be in the same installation as the mqweb server. A blank value indicates that no queue manager is configured as the gateway queue manager.

**V 9.3.5** This property is not returned in a stand-alone IBM MQ Web Server installation as the administrative REST API is not available in this environment.

### **V 9.3.2** `mqRestMessagingAdoptWebUserContext`

This configuration property is used to specify the user context that is used for authorization when you send, publish, receive, or browse messages by using the messaging REST API. That is, it specifies which user ID is used for authorization.

If the value is set to `true`, the ID that is used for authorization is the user ID that is logged in to the REST API.

If the value is set to `false`, the ID that is used for authorization is the user ID that is used to start the mqweb server.

### **mqRestMessagingEnabled**

This configuration property is used to specify whether the messaging REST API is enabled.

### **mqRestMessagingFullPoolBehavior**

This configuration property is used to specify the behavior of the messaging REST API when all connections in the connection pool are in use.

If the value is set to `block`, wait for a connection to become available if all the connections in the pool are in use. When this option is used, the wait for a connection is indefinite.

If the value is set to `error`, return an error if all connections in the pool are in use.


If the value is set to `overflow`, create a nonpooled connection to use if all the connections in the pool are in use.

### **mqRestMessagingMaxPoolSize**

This configuration property is used to specify the maximum connection pool size for each queue manager connection pool.


### **mqRestMftCommandQmgr**

This configuration property is used to specify the name of the command queue manager to which create transfer and create, delete, or update resource monitor requests are submitted by the REST API for MFT.

 This property is not returned in a stand-alone IBM MQ Web Server installation as the REST API for MFT is not available in this environment.


### **mqRestMftCoordinationQmgr**

This configuration property is used to specify the name of the coordination queue manager from which transfer details are retrieved by the REST API for MFT.

 This property is not returned in a stand-alone IBM MQ Web Server installation as the REST API for MFT is not available in this environment.


### **mqRestMftEnabled**

This configuration property is used to specify whether the REST API for MFT is enabled.

 This property is not returned in a stand-alone IBM MQ Web Server installation as the REST API for MFT is not available in this environment.

### **mqRestMftReconnectTimeoutInMinutes**

This configuration property is used to specify the length of time, in minutes, after which the REST API for MFT stops trying to connect to the coordination queue manager.




 This property is not returned in a stand-alone IBM MQ Web Server installation as the REST API for MFT is not available in this environment.

### **mqRestRequestTimeout**

This configuration property is used to specify the time, in seconds, before a REST request times out.

### **traceSpec**

This configuration property is used to specify the level of trace that is generated by the mqweb server. For a list of possible values, see [Configuring logging for the IBM MQ Console and REST API](#).

  The following properties are the additional properties that can be returned by the **dspmweb properties** command on z/OS, AIX, Linux, and Windows.  Some of the following properties are not available in a stand-alone IBM MQ Web Server installation.

### **httpHost**

This configuration property is used to specify the HTTP hostname as an IP address, domain name server (DNS) hostname with domain name suffix, or the DNS hostname of the server where IBM MQ is installed.

An asterisk specifies all available network interfaces, and the value `localhost` allows only local connections.

### **httpPort**

This configuration property is used to specify the HTTP port number that is used for HTTP connections.

If the value is set to `-1` the port is disabled.

### **httpsPort**

This configuration property is used to specify the HTTPS port number that is used for HTTPS connections.

If the value is set to `-1` the port is disabled.

### **ltpaCookieName**

This configuration property is used to specify the name of the LTPA token cookie name.

By default, the value of this property is `LtpaToken2_${env.MQWEB_LTPA_SUFFIX}` on AIX, Linux, and Windows, or `LtpaToken2_${httpsPort}` on z/OS. The variable after the `LtpaToken2_` prefix is used by the mqweb server to generate a unique name for the cookie. You cannot set this variable, but you can change the `ltpaCookieName` to a value of your choosing.

### **maxMsgTraceFiles**

This configuration property is used to specify the maximum number of messaging trace files that are generated by the mqweb server for the IBM MQ Console.

### **maxMsgTraceFileSize**

This configuration property is used to specify the maximum size, in MB, that each messaging trace file can reach.

This property applies only to the IBM MQ Console.

### **mqConsoleAutostart**

This configuration property is used to specify whether the IBM MQ Console automatically starts when the mqweb server starts.

### **mqConsoleFrameAncestors**

This configuration property is used to specify the list of origins of web pages that can embed the IBM MQ Console in an IFrame.

#### **V 9.3.0 mqConsoleRemoteAllowLocal**

This configuration property is used to specify whether remote and local queue managers are visible in the IBM MQ Console when remote queue manager connections are allowed. When this property is set to true, both local and remote queue managers are displayed.

**V 9.3.5** This property is not returned in a stand-alone IBM MQ Web Server installation as the IBM MQ Console can be used only with remote queue managers in this environment.

#### **V 9.3.0 mqConsoleRemotePollTime**

This configuration property is used to specify the time, in seconds, before the remote queue manager connections list is refreshed. On refresh, unsuccessful connections are retried.

#### **V 9.3.0 mqConsoleRemoteSupportEnabled**

This configuration property is used to specify whether the IBM MQ Console allows remote queue manager connections. When this property is set to true, remote queue manager connections are allowed.

**V 9.3.5** This property is not returned in a stand-alone IBM MQ Web Server installation as the IBM MQ Console can be used only with remote queue managers in this environment.

#### **V 9.3.0 mqConsoleRemoteUIAdmin**

This configuration property is used to specify whether remote queue managers can be added to the IBM MQ Console by using the Console, or if remote queue managers can be added only by using the **setmqweb remote** command. When this property is set to true, remote queue managers can be added by using the IBM MQ Console.

### **mqRestAutostart**

This configuration property is used to specify whether the REST API automatically starts when the mqweb server starts.

#### **V 9.3.3 mqRestMessagingConnectionMode**

This configuration property is used to specify whether the messaging REST API can send messages to queue managers that are not in the same installation as the mqweb server.

If the value is set to `local`, the messaging REST API can send messages only to queue managers that are in the same installation as the mqweb server.

If the value is set to `remote`, the messaging REST API can send messages to any queue manager that is configured for use by the messaging REST API.

**V 9.3.5** This property is not returned in a stand-alone IBM MQ Web Server installation as the messaging REST API can be used only with remote queue managers in this environment.



### V 9.3.0 remoteKeyfile

This configuration property is used to specify the location of the key file that contains the initial encryption key that is used to decrypt the passwords that are stored in the remote queue manager connection information.

### secureLtpa

This configuration property is used to specify whether the LTPA token is secured for all requests. An unsecured LTPA token is required in order to send HTTP requests from a browser.

**ALW** The following properties are the additional properties that can be returned by the **dspmweb properties** command on AIX, Linux, and Windows:

### managementMode

This configuration property is used to specify whether queue managers and listeners are able to be created, deleted, started, and stopped by the IBM MQ Console.

If the value is set to `standard`, queue managers and listeners can be created and administered in the IBM MQ Console.

If the value is set to `externallyprovisioned`, queue managers and listeners cannot be created in the IBM MQ Console. Only queue managers and listeners that are created outside of the IBM MQ Console can be administered.

## Return codes

Table 61. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

For a full list of server command exit codes, see [Liberty:server command options](#) in the WebSphere Application Server documentation.

## Related commands

Table 62. Related commands and descriptions

Command	Description
<a href="#">dspmweb status</a>	Display the mqweb server status
<a href="#">dspmweb remote</a>	Display the mqweb server remote queue manager configuration
<a href="#">strmqweb</a>	Start the mqweb server.
<a href="#">endmqweb</a>	Stop the mqweb server.
<a href="#">setmqweb pid</a>	Configure the product ID that the mqweb server runs under on z/OS.
<a href="#">“setmqweb properties (set mqweb server configuration properties)” on page 222</a>	Configure the mqweb server properties.
<a href="#">“setmqweb remote (set mqweb server remote queue manager configuration)” on page 230</a>	Configure the mqweb server remote queue manager connections.

## V 9.3.0 **dspmqweb remote (display mqweb server remote queue manager configuration)**

Display the remote queue manager connection information for the mqweb server. The remote queue manager connection information is used for the IBM MQ Console, and the messaging REST API to connect to remote queue managers.

### Purpose

Use the **dspmqweb remote** command to view details of the remote queue manager connections that are configured for use with the IBM MQ Console and the messaging REST API.

### Using the command on z/OS



Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, enter the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP\_user\_directory* is the name of the directory that is passed to **crtmqweb**. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

You must also set the `JAVA_HOME` environment variable to reference a 64-bit version of Java on your system.

### Using the command in a stand-alone IBM MQ Web Server installation



Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the `MQ_OVERRIDE_DATA_PATH` environment variable to the IBM MQ Web Server data directory.

The user ID running the command needs write access to the data directory and its subdirectories.

### Syntax

```
➔ dspmqweb remote { -li | -a | -uniqueName uniqueName } -l
```

### Optional parameters

**-li**

Displays all the remote queue manager unique IDs.

**-a**

Displays all the remote queue manager connection definitions and properties.

**-uniqueName *uniqueName***

Displays the remote queue manager connection definition and properties for the queue manager that is associated with the unique name.

**-l**

Enable verbose logging. Diagnostic information is written to a mqweb server log-file.

## Properties that can be returned by the command

The following global properties can be returned when the **-a** parameter is used:

### **globalTrustStorePath**

The path to the truststore JKS file. This truststore is used for all remote connections unless it is overridden by specific remote queue manager connection information in the **trustStorePath** entry.

### **globalTrustStorePassword**

The password for the global truststore.

This value is encrypted and is displayed as a series of asterisks.

### **globalKeyStorePath**

The path to the keystore JKS file. This keystore is used for all remote connections unless it is overridden by a specific remote queue manager connection information in the **keyStorePath** entry.

### **globalKeyStorePassword**

The password for the global keystore.

This value is encrypted and is displayed as a series of asterisks.

For each remote queue manager connection, the following properties can be returned:

### **uniqueName**

The unique name for the remote queue manager connection.

### **qmgrName**

The name of the queue manager.

### **ccdtURL**

The path to the CCDT file that is associated with the remote queue manager connection.

### **V 9.3.3 group**

Whether this remote queue manager connection is part of a queue manager group. If the value is **true**, then the **uniqueName** specifies the group name.

This property is valid only for the messaging REST API.

### **V 9.3.3 visibility**

Whether this remote queue manager connection can be used by the messaging REST API, the IBM MQ Console, or both.

### **username**

The username that is used for the remote queue manager connection.

### **password**

The password that is associated with the username that is used for the remote queue manager connection.

This value is encrypted in the remote queue manager connection information and is displayed as a series of asterisks.

### **enableMutualTLS**

Whether this remote queue manager connection uses mutual TLS.

### **trustStorePath**

The path to the truststore JKS file. This value overrides the global truststore value.

### **trustStorePassword**

The password for the truststore file.

This value is encrypted in the remote queue manager connection information and is displayed as a series of asterisks.

### **keyStorePath**

The path to the keystore JKS file. This value overrides the global keystore value.

### **keyStorePassword**

The password for the keystore file.

This value is encrypted in the remote queue manager connection information and is displayed as a series of asterisks.

## Return codes

Table 63. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

For a full list of server command exit codes, see [Liberty:server command options](#) in the WebSphere Application Server documentation.

## Related commands

Table 64. Related commands and descriptions

Command	Description
<a href="#">dspmqweb status</a>	Display the mqweb server status
<a href="#">dspmqweb properties</a>	Display the mqweb server configuration properties
<a href="#">strmqweb</a>	Start the mqweb server.
<a href="#">endmqweb</a>	Stop the mqweb server.
<a href="#">setmqweb pid</a>	Configure the product ID that the mqweb server runs under on z/OS.
<a href="#">“setmqweb properties (set mqweb server configuration properties)” on page 222</a>	Configure the mqweb server properties.
<a href="#">“setmqweb remote (set mqweb server remote queue manager configuration)” on page 230</a>	Configure the mqweb server remote queue manager connections.

## dspmqweb status (display mqweb server status)

Display information about the status of the mqweb server that is used to support the IBM MQ Console and REST API.

### Purpose

Use the **dspmqweb status** command to view information about the status of the mqweb server.

The mqweb server must be running to use the IBM MQ Console or the REST API. If the server is running then the available root context URLs and associated ports that are used by the IBM MQ Console and REST API are displayed by the **dspmqweb status** command.

### Using the command on z/OS



Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the WLP\_USER\_DIR environment variable so that the variable points to your mqweb server configuration.

To set the WLP\_USER\_DIR environment variable, enter the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP\_user\_directory* is the name of the directory that is passed to **crtmqweb**. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

You must also set the JAVA\_HOME environment variable to reference a 64-bit version of Java on your system.

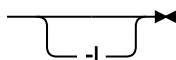
## Using the command in a stand-alone IBM MQ Web Server installation

V 9.3.5 Linux

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ\_OVERRIDE\_DATA\_PATH** environment variable to the IBM MQ Web Server data directory.

The user ID running the command needs write access to the data directory and its subdirectories.

### Syntax

►► **dspmqweb status** 

### Optional parameters

**-l**

Enable verbose logging. Diagnostic information is written to a mqweb server log-file.

### Return codes

Table 65. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

For a full list of server command exit codes, see [Liberty:server command options](#) in the WebSphere Application Server documentation.

### Related commands

Table 66. Related commands and descriptions

Command	Description
<a href="#">dspmqweb properties</a>	Display the mqweb server configuration properties
<a href="#">dspmqweb remote</a>	Display the mqweb server remote queue manager configuration
<a href="#">strmqweb</a>	Start the mqweb server.
<a href="#">endmqweb</a>	Stop the mqweb server.
<a href="#">setmqweb pid</a>	Configure the product ID that the mqweb server runs under on z/OS.

Table 66. Related commands and descriptions (continued)

Command	Description
<a href="#">“setmqweb properties (set mqweb server configuration properties)”</a> on page 222	Configure the mqweb server properties.
<a href="#">“setmqweb remote (set mqweb server remote queue manager configuration)”</a> on page 230	Configure the mqweb server remote queue manager connections.

## Multi **endmqscsv (end command server)**

Stop the command server for a queue manager.

### Purpose

Use the **endmqscsv** command to stop the command server on the specified queue manager.

You must use the **endmqscsv** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmqs -o` installation command.

If the queue manager attribute, SCMDSERV, is specified as QMGR then changing the state of the command server using **endmqscsv** does not effect how the queue manager acts upon the SCMDSERV attribute at the next restart.

### Syntax



### Required parameters

#### QMGrName

The name of the queue manager for which to end the command server.

### Optional parameters

#### -c

Stops the command server in a controlled manner. The command server can complete the processing of any command message that it has already started. No new message is read from the command queue.

This parameter is the default.

#### -i

Stops the command server immediately. Actions associated with a command message currently being processed might not complete.

### Return codes

Table 67. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

## Examples

1. The following command stops the command server on queue manager `saturn.queue.manager`:

```
endmqcsv -c saturn.queue.manager
```

The command server can complete processing any command it has already started before it stops. Any new commands received remain unprocessed in the command queue until the command server is restarted.

2. The following command stops the command server on queue manager `pluto` immediately:

```
endmqcsv -i pluto
```

## Related commands

Table 68. Related command names and descriptions

Command	Description
<a href="#">strmqcsv</a>	Start a command server
<a href="#">dspmqcsv</a>	Display the status of a command server

## Related reference

[“Command server commands” on page 7](#)

A table of command server commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

## **Windows** endmqdnm (stop .NET monitor)

Stop the .NET monitor for a queue ( Windows only).

## Purpose

**Note:** The `endmqdnm` command applies to IBM MQ for Windows only.

Use the `endmqdnm` control command to stop a .NET monitor.

## Syntax

```
endmqdnm -q QueueName -m QMgrName
```

## Required parameters

### **-q *QueueName***

The name of the application queue that the .NET monitor is monitoring.

## Optional parameters

### **-m *QMGrName***

The name of the queue manager that hosts the application queue.

If omitted, the default queue manager is used.

## Return codes

Table 69. Return code identifiers and descriptions

Return code	Description
0	Successful operation
36	Invalid arguments supplied
40	Queue manager not available
58	Inconsistent use of installations detected
71	Unexpected error
72	Queue manager name error
133	Unknown object name error

## Related tasks

[Using the .NET monitor](#)

## **endmq1sr (end listener)**

End all listener process for a queue manager.

## Purpose

The **endmq1sr** command ends all listener processes for the specified queue manager.

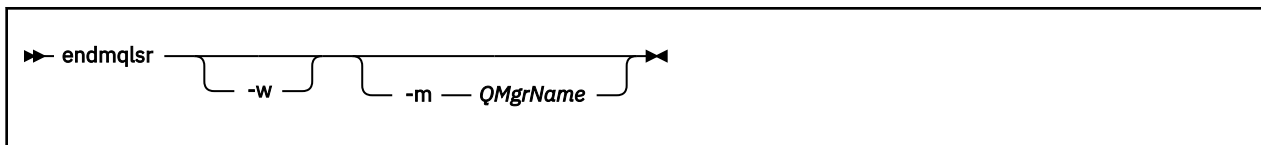
You must use the **endmq1sr** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmqr -o installation` command.

You do not have to stop the queue manager before issuing the **endmq1sr** command. If any of the listeners are configured to have inbound channels running within the **runmq1sr** listener process, rather than within a pool process, the request to end that listener might fail if channels are still active. In this case a message is written indicating how many listeners were successfully ended and how many listeners are still running.

If the listener attribute, CONTROL, is specified as QMGR then changing the state of the listener using **endmq1sr** does not effect how the queue manager acts upon the CONTROL attribute at the next restart.

## Syntax





## Optional parameters

### -m *QMgrName*

The name of the queue manager. If you omit this parameter, the command operates on the default queue manager.

### -w

Wait before returning control.

Control is returned to you only after all listeners for the specified queue manager have stopped.

## Return codes

Table 70. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

### Related tasks

[Applying maintenance level updates to multi-instance queue managers on AIX](#)

[Applying maintenance level updates to multi-instance queue managers on Linux](#)

[Applying maintenance level updates to multi-instance queue managers on Windows](#)

### Related reference

[“Listener commands” on page 12](#)

A table of listener commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

## **Multi** **endmqm (end queue manager)**

Stop a queue manager or switch to a standby queue manager or to a replica queue manager.

### Purpose

Use the **endmqm** command to end (stop) a specified queue manager. This command stops a queue manager in the following modes:

- Controlled or quiesced shutdown
- Immediate shutdown
- Pre-emptive shut down
- Wait shutdown

The **endmqm** command stops all instances of a multi-instance queue manager in the same way as it stops a single instance queue manager. You can issue the **endmqm** on either the active instance, or one of the standby instances of a multi-instance queue manager. You must issue **endmqm** on the active instance to end the queue manager.

If you issue the **endmqm** command on the active instance of a multi-instance queue manager, you can permit a standby instance to switch over to being the new active instance when the current active instance completes its shutdown.

If you issue the **endmqm** command on a standby instance of a multi-instance queue manager, you can end the standby instance by adding the **-x** option, and leave the active instance running. The queue manager reports an error if you issue **endmqm** on the standby instance without the **-x** option.

**CP4I** You can issue the **endmqm** command on the active or replica nodes of a Native HA group. A check is performed to see if stopping the specified instance breaks the group's quorum, and the command fails if it does. If you issue **endmqm -s** on the active instance, that instance stops and one of the replicas becomes the active instance. If you issue **endmqm -x** on a replica instance, the instance is stopped.

Issuing the **endmqm** command will affect any client application connected through a server-connection channel. The effect varies depending on the parameter used, but it is as though a **STOP CHANNEL** command was issued in one of the three possible modes. See [Stopping MQI channels](#), for information about the effects of **STOP CHANNEL** modes on server-connection channels. The **endmqm** optional parameter descriptions state which **STOP CHANNEL** mode they will be equivalent to.

If you issue **endmqm** to stop a queue manager, reconnectable clients do not try to reconnect. To override this behavior, specify either the **-x** or **-s** option to enable clients to start trying to reconnect.

**Note:** If a queue manager or a channel ends unexpectedly, reconnectable clients start trying to reconnect.

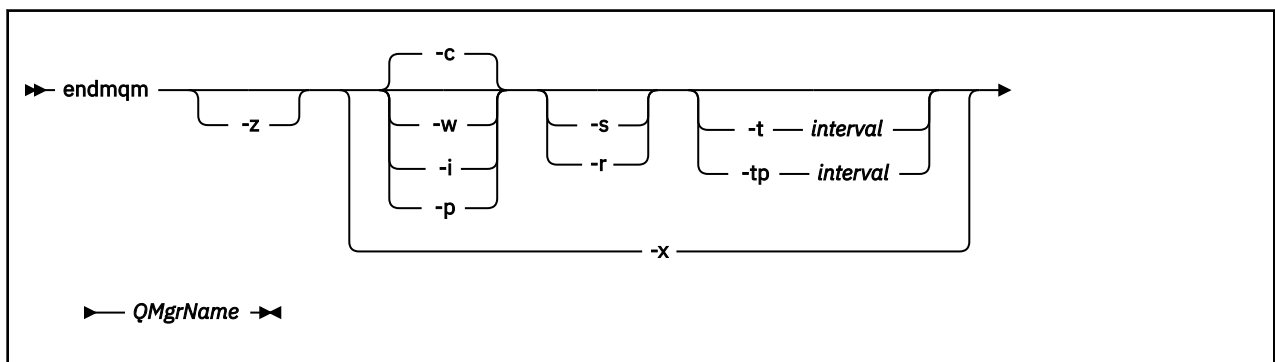
**Note:** The client might not reconnect to this queue manager. Depending on the MQCONNX reconnect option the client has used, and the definition of the queue manager group in the client connection table, the client might reconnect to a different queue manager. You can configure the client to force it to reconnect to the same queue manager.

You must use the **endmqm** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the **dspmqr -o** installation command.

The attributes of the queue manager and the objects associated with it are not affected by the **endmqm** command. You can restart the queue manager using the **strmqm** (Start queue manager) command.

To delete a queue manager, stop it and then use the **dlmqm** (Delete queue manager) command.

## Syntax



## Required parameters

### QMgrName

The name of the message queue manager to be stopped.

## Optional parameters

**-c**

Controlled (or quiesced) shutdown. This parameter is the default.

The queue manager stops, but only after all applications have disconnected. Any MQI calls currently being processed are completed. In the unlikely event that a “[dspmq \(display queue managers\)](#)” on [page 73](#) command is issued in the small timeframe between the applications disconnecting and the queue manager actually stopping, the “[dspmq \(display queue managers\)](#)” on [page 73](#) command might transiently report the status as `Ending immediately`, even though a controlled shutdown was requested.

Control is returned to you immediately and you are not notified of when the queue manager has stopped.

The effect on any client applications connected through a server-connection channel is equivalent to a **STOP CHANNEL** command issued in QUIESCE mode.

**-i**

Immediate shutdown. The queue manager stops after it has completed all the MQI calls currently being processed. Any MQI requests issued after the command has been issued fail. Any incomplete units of work are rolled back when the queue manager is next started.

Control is returned after the queue manager has ended.

The effect on any client applications connected through a server-connection channel is equivalent to a **STOP CHANNEL** command issued in FORCE mode.

**-p**

Pre-emptive shutdown.

**Important:** Use this type of shutdown only in exceptional circumstances, for example, when a queue manager does not stop as a result of a normal **endmqm** command.

The queue manager might stop without waiting for applications to disconnect or for MQI calls to complete. This can give unpredictable results for IBM MQ applications. The shutdown mode is set to *immediate shutdown*. If the queue manager has not stopped after a few seconds, the shutdown mode is escalated, and all remaining queue manager processes are stopped.


The effect on any client applications connected through a server-connection channel is equivalent to a **STOP CHANNEL** command issued in TERMINATE mode.

**-r**

Start trying to reconnect reconnectable clients. This parameter has the effect of reestablishing the connectivity of clients to other queue managers in their [queue manager group](#).

**-s**

For a multi-instance queue manager, switch over to a standby queue manager instance after shutting down. The command checks that there is a standby instance running before ending the active instance. It does not wait for the standby instance to start before ending. Connections to the queue manager are broken by the active instance shutting down. Reconnectable clients start trying to reconnect. You can configure the reconnection options of a client to reconnect only to another instance of the same queue manager, or to reconnect to other queue managers in the queue manager group.

 For the active instance of a Native HA group, switch over to a replica instance after shutting down.

- Specify the `-r` option to help client applications to reconnect to another instance.
- If this instance is not the active instance in the Native HA group then the command fails.
- If ending this active instance would cause the group quorum to fail then the command fails. (If other instances end or become unavailable at the same time as you run this command, the quorum check

might not detect this, the Native HA group ends and can only be restarted when enough instances are available.)

**-w**

Wait shutdown.

This type of shutdown is equivalent to a controlled shutdown except that control is returned to you only after the queue manager has stopped. You receive the message `Waiting for queue manager qmName to end` while shutdown progresses. In the unlikely event that a “[dspmq \(display queue managers\)](#)” on page 73 command is issued in the small timeframe between the applications disconnecting and the queue manager actually stopping, the “[dspmq \(display queue managers\)](#)” on page 73 command might transiently report the status as `Ending immediately`, even though a controlled shutdown was requested.

The effect on any client applications connected through a server-connection channel is equivalent to a **STOP CHANNEL** command issued in QUIESCE mode.

**-x**

For a multi-instance queue manager, end a standby instance of the queue manager, without ending the active instance of the queue manager.

**CP41** For a replica instance of a Native HA group, stop the instance:

- If this instance is the active instance in the Native HA group then the command fails.
- If ending this replica instance would cause the group quorum to fail then the command fails. (If other instances end or become unavailable at the same time as you run this command, the quorum check might not detect this, the Native HA group ends and can only be restarted when enough instances are available.)

**-z**

Suppresses error messages on the command.

**-t <interval>**

The target time in which ending the queue manager within *<interval>* seconds is attempted, escalating the phases of application disconnection. Essential queue manager maintenance tasks are allowed to complete, which might prolong the phase of the queue manager ending. (See [Ending a queue manager within a target time](#) for more details.)

**-tp <interval>**

The target time in which ending the queue manager within *<interval>* seconds is attempted, escalating the phases of application disconnection. Essential queue manager maintenance tasks are interrupted if necessary. (See [Ending a queue manager within a target time](#) for more details.)

These maintenance tasks include the attempt to retain non-persistent messages, when NPMCLASS is set to HIGH on a queue.

## Return codes

Table 71. Return code identifiers and descriptions

Return code	Description
0	Queue manager ended
3	Queue manager being created
<b>V 9.3.2</b>	
<b>V 9.3.2</b>	Queue Manager is being started
4	
16	Queue manager does not exist

Table 71. Return code identifiers and descriptions (continued)

Return code	Description
39	Invalid parameter specified
40	Queue manager not available
49	Queue manager stopping
58	Inconsistent use of installations detected
62	The queue manager is associated with a different installation
69	Storage not available
71	Unexpected error
77	IBM MQ queue manager cannot switch over
79	Active instance of IBM MQ queue manager <i>QmgrName</i> not ended
90	Standby instance of IBM MQ queue manager <i>QmgrName</i> not ended
119	Permission denied

### Examples

The following examples show commands that stop the specified queue managers.

1. This command ends the queue manager named `mercury.queue.manager` in a controlled way. All applications currently connected are allowed to disconnect.

```
endmqm mercury.queue.manager
```

2. This command ends the queue manager named `saturn.queue.manager` immediately. All current MQI calls complete, but no new ones are allowed.

```
endmqm -i saturn.queue.manager
```

The results of issuing **endmqm** to the local instance of a multi-instance queue manager are shown in the previous table. The results of the command depend on whether the `-s` or `-x` switch is used, and on the running status of local and remote instances of the queue manager. The following table shows the results of issuing **endmqm** to Native HA instances.

endmqm option	Local machine	Remote machine	RC	Message	Result
	Active	None	0	-	Queue manager ended.
		Standby			Queue manager ended, including the standby instance.
	Standby	Active	90	AMQ8368	Standby instance of IBM MQ queue manager <i>QmgrName</i> not ended.

Table 72. endmqm actions (continued)

endmqm option	Local machine	Remote machine	RC	Message	Result
CP4I	Native HA Active	-	0	-	Active instance ended. Remaining instances communicate and elect a new Active, if they can find a quorum.
	Native HA Replica	-	0	-	Replica instance ended. Remaining instances continue, if they have quorum.
-s	Active	None	77	AMQ7276	IBM MQ queue manager cannot switch over.
		Standby	0	-	Queue manager QMNAME ended, permitting switchover to a standby instance.
	Standby	Active	90	AMQ8368	Standby instance of IBM MQ queue manager <i>QmgrName</i> not ended.
CP4I -s	Native HA Active	Quorum would remain	0	-	Active instance ended.
	Native HA Active	Quorum would be broken	79	AMQ7275	Instance has not ended because it would cause quorum to be lost.
	Native HA Replica	-	90	AMQ7277	This operation is not valid on a Replica instance.
-x	Active	None	79	AMQ8367	Active instance of IBM MQ queue manager <i>QmgrName</i> not ended.
		Standby			
	Standby	Active	0	-	Standby instance of queue manager QMNAME ended.
CP4I -x	Native HA Active	-	79	AMQ8367	Active instance of IBM MQ queue manager 'MG' not ended.
	Native HA Replica	Quorum would remain	0	-	Replica instance ended.
	Native HA Replica	Quorum would be broken	90	AMQ7275	Instance has not ended because it would cause quorum to be lost.

**Related tasks**

Stopping a queue manager

[Applying maintenance level updates to multi-instance queue managers on AIX](#)

[Applying maintenance level updates to multi-instance queue managers on Linux](#)

[Applying maintenance level updates to multi-instance queue managers on Windows](#)

### Related reference

[crtmqm \(create queue manager\)](#)

Create a queue manager.

[endmqm \(end queue manager\)](#)

Stop a queue manager or switch to a standby queue manager or to a replica queue manager.

[dlmqm \(delete queue manager\)](#)

Delete a queue manager.

## endmqsvc (end IBM MQ service)

End the IBM MQ service on Windows.

### Purpose

The command ends the IBM MQ service on Windows. Run the command on Windows only.

If you are running IBM MQ on Windows systems with User Account Control (UAC) enabled, you must invoke **endmqsvc** with elevated privileges. To open an elevated command prompt, right click the command prompt icon and select **Run as administrator** (see [Authority to administer IBM MQ on AIX, Linux, and Windows](#)).

Run the command to end the service, if the service is running.

Restart the service for IBM MQ processes to pick up a new environment, including new security definitions.

### Syntax

**endmqsvc**

### Parameters

The **endmqsvc** command has no parameters.

You must set the path to the installation that contains the service. Either make the installation primary, run the [setmqenv](#) command, or run the command from the directory containing the **endmqsvc** binary file.

### Related reference

[“strmqsvc \(start IBM MQ service\)” on page 237](#)

Start the IBM MQ service on Windows.

## endmqtrc (end trace)

End trace for some or all of the entities that are being traced.

### Purpose

Use the **endmqtrc** command to end tracing for the specified entity or all entities. The **endmqtrc** command ends only the trace that is described by its parameters. Using **endmqtrc** with no parameters ends early tracing of all processes.

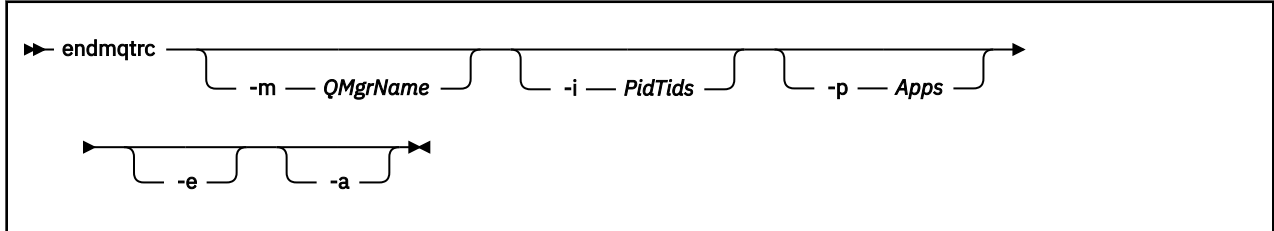
All **endmqtrc** commands set the type of output to *mqm* on [strmqtrc](#).



**Attention:** There can be a slight delay between the **endmqtrc** command ending, and all trace operations actually completing. This is because IBM MQ processes are accessing their own trace files. As each process becomes active at different times, their trace files close independently of one another.

## Syntax

The syntax of this command is as follows:



## Optional parameters

### -m *QMgrName*

The name of the queue manager for which to end tracing.

The *QMgrName* supplied must match exactly the *QMgrName* supplied on the **strmqtrc** command. If the **strmqtrc** command used wildcards, the **endmqtrc** command must use the same wildcard specification including the escaping of any wildcard characters to prevent them being processed by the command environment.

A maximum of one -m flag and associated queue manager name can be supplied on the command.

### -i *PidTids*

Process identifier (PID) and thread identifier (TID) for which to end tracing. You cannot use the **-i** flag with the **-e** flag. If you try to use the **-i** flag with the **-e** flag, then an error message is issued. This parameter must only be used under the guidance of IBM Service personnel.

### -p *Apps*

The named processes for which to end tracing. *Apps* is a comma-separated list. You must specify each name in the list exactly as the program name would be displayed in the "Program Name" FDC header. Asterisk (\*) or question mark (?) wildcards are allowed. You cannot use the **-p** flag with the **-e** flag. If you try to use the **-p** flag with the **-e** flag, then an error message is issued.

### -e

Ends early tracing of all processes.

Using **endmqtrc** with no parameters has the same effect as **endmqtrc -e**. You cannot specify the **-e** flag with the **-m** flag, the **-i** flag, or the **-p** flag.

### -a

Ends all tracing.

**Important:** This flag must be specified alone.

## Return codes

Table 73. Return code identifiers and descriptions

Return code	Description
-------------	-------------

AMQ5611	This message is issued if you supply invalid arguments to the command.
---------	--

58	Inconsistent use of installations detected
----	--



## Examples

This command ends tracing of data for a queue manager called QM1.

```
endmqtrc -m QM1
```

The following examples are a sequence that shows how the `endmqtrc` command ends only the trace that is described by its parameters.

1. The following command enables tracing for queue manager QM1 and process `amqxxx.exe`:

```
strmqtrc -m QM1 -p amqxxx.exe
```

2. The following command enables tracing for queue manager QM2:

```
strmqtrc -m QM2
```

3. The following command ends tracing for queue manager QM2 only. Tracing of queue manager QM1 and process `amqxxx.exe` continues:

```
endmqtrc -m QM2
```

## Related commands

Table 74. Related command names and descriptions

Command	Description
<a href="#">dspmqtrc</a>	Display formatted trace output
<a href="#">“strmqtrc (start trace)” on page 242</a>	Start trace

### Related tasks

[Using trace](#)

### Related reference

Command sets comparison: Other commands

A table of other commands, showing the command description, and the equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.



## **endmqweb (end mqweb server)**

Stop the `mqweb` server that is used to support the IBM MQ Console and REST API.

### Purpose

Use the **endmqweb** command to stop the `mqweb` server. If you stop the `mqweb` server, you cannot use the IBM MQ Console or the REST API.

### Usage notes

  Before you issue the **endmqweb** command in a stand-alone IBM MQ Web Server installation, you must set the **MQ\_OVERRIDE\_DATA\_PATH** environment variable to the IBM MQ Web Server data directory.

## Syntax

▶▶ endmqweb ▶▶

## Optional parameters

None.

## Return codes

Table 75. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

For a full list of server command exit codes, see [Liberty:server command options](#) in the WebSphere Application Server documentation.

## Related commands

Table 76. Related command names and descriptions

Command	Description
<a href="#">dspmqweb</a>	Display the status of the mqweb server.
<a href="#">strmqweb</a>	Start the mqweb server.

## ALW **migmqlog (migrate IBM MQ logs)**

The **migmqlog** command migrates logs, and can also change the type of your queue manager logs from linear to circular or from circular to linear.

**IBM i** **z/OS** **migmqlog** is not supported on IBM i or z/OS.

## Usage notes

**Windows** On Windows, running **migmqlog** enables you to move your queue manager logs to an Advanced Format disk

**migmqlog** can only run when the queue manager is inactive.

If the running of **migmqlog** is interrupted by, for example, a power failure, you should be rerun the same command until it completes normally.

A partially migrated log can not be used to start a queue manager, and the result of attempting to do so is not well defined.

**migmqlog** migrates logs 'in place', or migrates logs to a new location. When logs are migrated to a new log location, no change is made to any existing log files and all valid recovery log files in the old location are migrated to the new location.

**migmqlog** updates the `qm.ini` file to reflect the new log configuration, that is, **LogType** and **LogPath**, as needed.

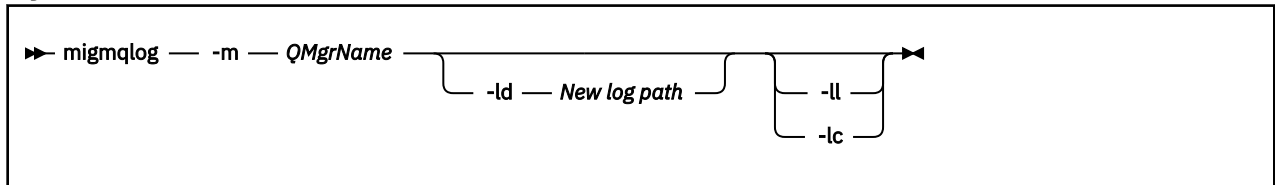
Following any log migration, the log is configured such that all future log writes occur with 4096 byte alignment, at minimum.

## Windows

For further information on migrating logs on Windows to be Advanced Format, see [Migrating logs to an Advanced Format disk](#).

See [Types of logging](#) for more information about linear and circular logging.

## Syntax



## Required parameters

### -m *QMgrName*

The name of the queue manager on which to migrate logs.

## Optional parameters

### -ld *New log path*

If you specify **-ld** and do not point to the existing log location, migration will be to a new log location.

If you do not specify **-ld**, or you specify **-ld** and point to the existing log location, migration will be 'in place'.

### -ll

If you pass **-ll** to the command, and the queue manager is currently defined to be using circular logging, the queue manager will be reconfigured to use linear logging.

### -lc

If you pass **-lc** to the command, and the queue manager is currently defined to be using linear logging, the queue manager will be reconfigured to use circular logging.

## Related tasks

[Migrating the log of your queue manager from linear to circular](#)

[Migrating the log of your queue manager from circular to linear](#)

## Multi

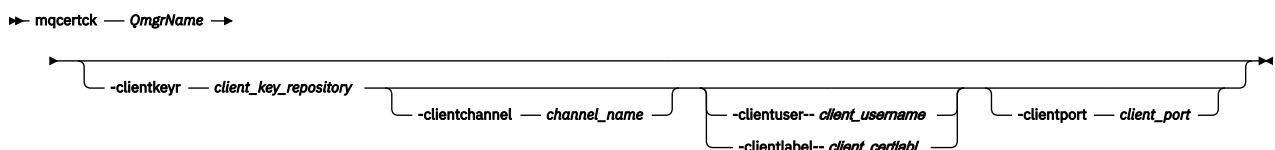
## mqcertck (certify TLS setup)

Use the **mqcertck** command to diagnose potential TLS problems with your queue managers.

## Purpose

The command can be used as a first check to determine why a connection using TLS has been unable to successfully connect to queue managers within your enterprise, and works with multiple certificates.

## Syntax



## Required parameters

### *QmgrName*

Name of the queue manager to check for TLS errors.

## Optional parameters

### **-clientkeyr** *client\_key\_repository*

Required if you supply the **-clientuser**, **-clientlabel**, **-clientchannel**, or **-clientport** parameters.

Location of the client key repository used by a client application connecting to the referenced queue manager.

**Important:** You must supply the name without the `.kdb` extension.

### **-clientuser** *client\_username*

Cannot be used if you supplied the **-clientlabel** parameter.

User running the client application that connects to the referenced queue manager. If supplied, requires **-clientkeyr**.

### **-clientlabel** *client\_certlbl*

Cannot be used if you supplied the **-clientuser** parameter.

Certificate label that is given to the client that connects to the referenced queue manager by using one of the IBM MQ MQI client CERTLABL methods. If supplied, requires **-clientkeyr**.

### **-clientchannel** *channel\_name*

Name of the channel on the referenced queue manager to check for TLS errors. If supplied, requires **-clientkeyr**.

### **-clientport** *port\_number*

Specify a specific port to use when testing the client.

The value must be:

- An integer value between 1 and 65535 inclusive.
- A port number, which must be a free port that **mqcertck** can use during its client checks.
- Not be a port that is in use by the queue manager, or any other process on the machine running **mqcertck**.

If you do not specify a value, port 5857 is used. If supplied, requires **-clientkeyr**.

## Examples

### Example 1

After configuring an IBM MQ queue manager for TLS connections, you can use **mqcertck** to verify that no mistakes have been made, before attempting to start your channels.

The information returned in the example shows that no certificate has been found for queue manager `qmgr`.

```
[mqm@mq-host ~]$ mqcertck qmgr
5724-H72 (C) Copyright IBM Corp. 1994, 2024.
+-----+
| IBM MQ TLS Configuration Test tool
+-----+

ERROR:
No Certificate could be found for the Queue Manager qmgr

EXPLANATION:
Queue managers will use a certificate with the label set in the Queue Manager's
CERTLABL attribute. There is no certificate with the label ibmwebspheremqqmgr
in the key repository being used by the queue manager The Key repository being
used is located at /var/mqm/qmgrs/qmgr/ssl/key.kdb.

ACTION:
A valid certificate with the label ibmwebspheremqqmgr needs to be added to the
key repository.
+-----+
```

This application has ended. See above for any problems found.  
If there are problems then resolve these and run this tool again.  
+-----

## Example 2

After creating a key repository, certificate, and exchanging certificates for a client application, you can use **mqcertck** to verify that a client application is able to connect to a queue manager.

To do this, you need to run **mqcertck** on the machine where the IBM MQ queue manager is running, and have access to the client key repository.

You can do this in a variety of ways, for example, a file system mount. After you have set up your machine, run the following command:

```
mqcertck QmgrName -clientkeyr Location_of_Client_Key_Repository  
                -clientlabel Client_certificate_label
```

For example:

```
mqcertck qmgr -clientkeyr /var/mqm/qmgrs/qmgr/ssl/key  
                -clientlabel ibmwebspheremqmgr
```

Check the output for any problems identified with your configuration.

Note that, if you are planning on having your clients connect anonymously, you can run the preceding command without the **-clientlabel** parameter.

## Linux AIX **mqconfig (check system configuration)**

Checks that the system configuration meets the requirements to run IBM MQ (AIX and Linux platforms only).

### Purpose

The **mqconfig** command is run to verify the system configuration matches or exceeds that which is required by an IBM MQ queue manager environment. The configuration values are minimum values, and large installations might require values greater than those checked by this command.

For further information about configuring your system for IBM MQ, see the *Operating system configuration and tuning information for IBM MQ* on the platform, or platforms, that your enterprise uses.

### Syntax



### Optional parameters

#### **-v Version**

The system requirements vary between different versions of IBM MQ. Specify the version of IBM MQ for which you need to verify the current system configuration.

The default value, if **-v** is not specified, is the current version.

## Example

The following output is an example of what the command produces on a Linux system:

```
# mqconfig -v 8.0
mqconfig: V3.7 analyzing Red Hat Enterprise Linux Server release 6.5
(Santiago) settings for IBM MQ V8.0

System V Semaphores
semmsl (sem:1) 500 semaphores          IBM>=32      PASS
semms  (sem:2) 35 of 256000 semaphores (0%) IBM>=4096 PASS
semopm (sem:3) 250 operations          IBM>=32      PASS
semmni  (sem:4) 3 of 1024 sets          (0%) IBM>=128 PASS

System V Shared Memory
shmmax  68719476736 bytes              IBM>=268435456 PASS
shmmni  1549 of 4096 sets               (37%) IBM>=4096 PASS
shmall  7464 of 2097152 pages           (0%) IBM>=2097152 PASS

System Settings
file-max 4416 of 524288 files           (1%) IBM>=524288 PASS

Current User Limits (root)
nofile (-Hn) 10240 files                 IBM>=10240 PASS
nofile (-Sn) 10240 files                 IBM>=10240 PASS
nproc  (-Hu) 11 of 30501 processes       (0%) IBM>=4096 PASS
nproc  (-Su) 11 of 4096 processes        (1%) IBM>=4096 PASS
```

**Note:** Any values listed in the **Current User Limits** section are resource limits for the user who ran **mqconfig**. If you normally start your queue managers as the mqm user, you should switch to mqm and run **mqconfig** there.

If other members of the mqm group (and perhaps root) also start queue managers, all those members should all run **mqconfig**, to ensure that their limits are suitable for IBM MQ.

The limits displayed by **mqconfig** are not applied to queue managers on Linux started with **systemd**.

## Related tasks

[Configuring and tuning the operating system on Linux](#)

## Windows Linux MQExplorer (launch IBM MQ Explorer)

Start IBM MQ Explorer (Windows and Linux x86-64 platforms only).

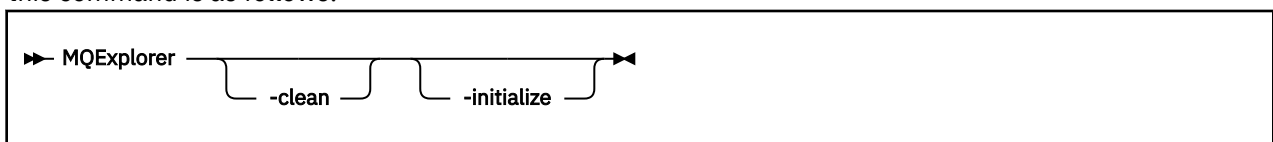
## Purpose

You can start IBM MQ Explorer by using the **MQExplorer** command in the installation directory.

**V 9.3.0** **V 9.3.0** The **MQExplorer** command is in `MQ_EXPLORER_INSTALLATION_PATH`, where `MQ_EXPLORER_INSTALLATION_PATH` is the installation path for the stand-alone IBM MQ Explorer. For more information, see [Installing and uninstalling IBM MQ Explorer as a stand-alone application on Linux and Windows](#).

## Syntax

**MQExplorer.exe** (the MQExplorer command) supports standard Eclipse runtime options. The syntax of this command is as follows:



## Optional parameters

### -clean

Cleans the caches used by the Eclipse runtime to store bundle dependency resolution and eclipse extension registry data. Using this option forces Eclipse to reinitialize these caches.

## -initialize

Initializes the configuration being run. All runtime-related data structures and caches are refreshed. Any user/plugin defined configuration data is not purged.

No application is run, any product specifications are ignored and no UI is presented (for example, the splash screen is not drawn).

## Related tasks

[Launching IBM MQ Explorer](#)

## Linux: **mqlicense (accept license post installation)**

Use the `mqlicense` command on Linux to accept an IBM MQ license after installation.

## Purpose

On Linux (excluding IBM MQ Appliance) use the **mqlicense** command to accept the IBM MQ license post installation.

**Note:** You must have the appropriate privileges to run this command on your system, typically root access on Linux.

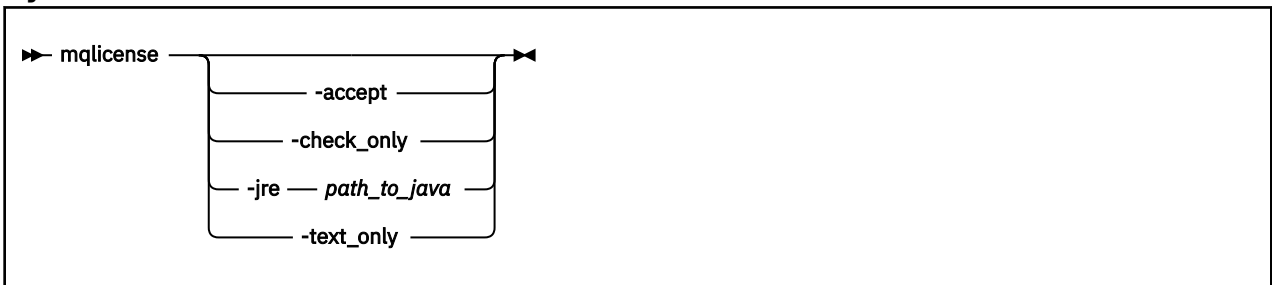
The license agreement is displayed in a language appropriate to your environment and you are prompted to accept or decline the terms of the license.

If possible, **mqlicense** opens an X-window to display the license.

If you need the license to be presented as text in the current shell, which can be read by a screen reader, type the following command:

```
mqlicense -text_only
```

## Syntax



## Required parameters

None

## Optional parameters

### -accept

Accept the IBM MQ license without it being displayed.

**V 9.3.5**

### -check\_only

Check whether the IBM MQ license has already been accepted without it being displayed.

### -jre

Path to the Java executable used to display the license.

### -text\_only

Display a text-only version of the license, which can be read by a screen-reader.

## Return codes

Table 77. Return code identifiers and descriptions

Return code	Description
0	Successful completion. You can accept or decline the result, depending upon what you chose.
10	A warning occurred
20	An error occurred

## Usage notes

Note that running this command, with the environment variable **MQLICENSE=accept**, has the same effect as specifying the **-accept** parameter.

### Related concepts

[Accepting the license on IBM MQ for Linux](#)

### Related reference

[MQLICENSE](#)

[“dspmqlic \(display IBM MQ license\)” on page 91](#)

Display an IBM MQ license.

## Multi **mqrc (display return code and AMQ message information)**

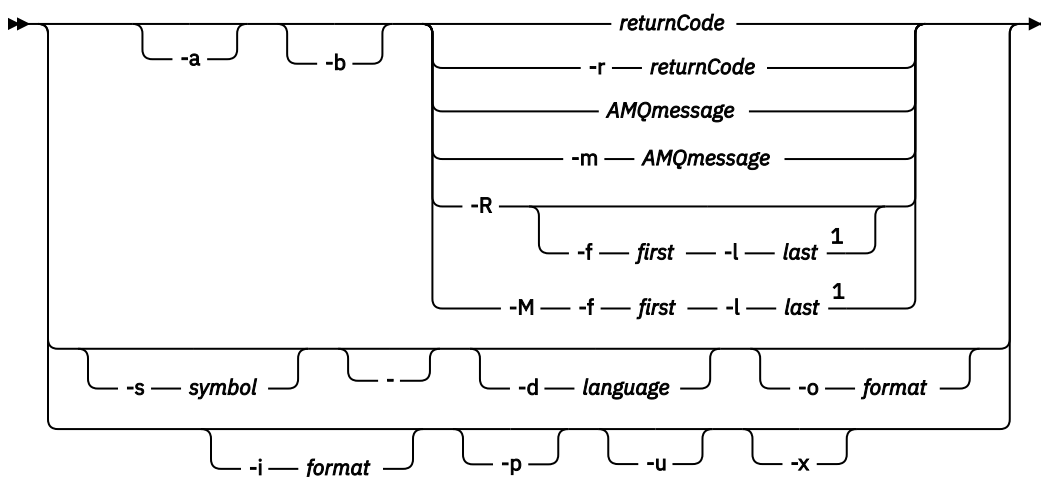
Display information about return codes and AMQ messages.

## Purpose

You can use the **mqrc** command to display information about symbols, return codes, and AMQ messages. You can specify a range of return codes or AMQ messages, as well as specifying specific return codes or AMQ messages.

Numeric arguments are interpreted as decimal if they start with a digit 1 - 9, or hex if prefixed with 0x.

## Syntax



### Notes:

<sup>1</sup> If there is a problem with a message within a range, an indication is displayed before the message text. ? is displayed if there are no matching return codes for the message. ! is displayed if the message severity is different to the return code severity.



## Parameters

### **returnCode**

The return code to display

### **AMQmessage**

The AMQ message to display

### **symbol**

The symbol to display

### **-a**

Try all severities to find message text

### **-b**

Display messages without extended information

### **-f first**

First number in a range

### **-l last**

Last number in a range

### **-m AMQmessage**

The AMQ message to list

### **-M**

Display AMQ messages in a range

### **-r returnCode**

The return code to display

### **-R**

Display all return codes. If used with the **-f** and **-l** parameters, **-R** displays the return codes within a range.

### **-s symbol**

The symbol to display

-

If a - is given as a trailing parameter, it indicates that further input will come from stdin.

### **-d language**

Display the message in the specified language, for example, Fr\_FR.

### **-i format**

Determine the message to display from a message in the specified format, which must be one of the following:

#### **text**

The textual format of the **QLErrorLog** service, including the Insert attributes.

#### **json**

JSON format diagnostic messages, specified in UTF-8.

### **-o format**

Display the message in the specified format, which must be one of the following:

#### **mqr**

The format used by **mqr** in previous versions of the product.

#### **text**

The textual format of the **QLErrorLog** service.

#### **json**

The JSON format, described in JSON format diagnostic messages.

### **-p**

Display the message explanation only. For example:

```
mqr -p AMQ8118
```

displays

The queue manager `insert_5` does not exist.

**ALW** **-u**

Display the user response only. For example:

```
mqrc -u AMQ8118
```

displays

Either create the queue manager (`crtmqm` command) or correct the queue manager name used in the command and then try the command again.

**-x**

Display extended message information, including the message severity. For example, the following message has an error (**E**) severity of 30:

```
mqrc -x AMQ8118
536903960 0x20008118 E 30 urcMS_MQCONN_FAILED
536903960 0x20008118 E 30 zrc_CSPRC_Q_MGR_DOES_NOT_EXIST
```

MESSAGE:  
IBM MQ queue manager does not exist.

EXPLANATION:  
The queue manager `<insert three>` does not exist.

ACTION:  
Either create the queue manager (`crtmqm` command) or correct the queue manager name used in the command and then try the command again.

## Examples

1. This command displays AMQ message 5005:

```
mqrc AMQ5005
```

2. This command displays return codes in the range 2505 - 2530:

```
mqrc -R -f 2505 -l 2530
```

3. Running the following command, where `AMQERR01.json` contains JSON formatted messages in any language, converts all messages into US English in the original textual **QMErrorLog** format:

```
cat AMQERR01.json | mqrc -d En_US -i json -o text -
```

Alternatively, you could take `AMQERR01.LOG` and convert it to JSON:

```
cat AMQERR01.LOG | mqrc -i text -o json -
```

4. Running the following command, where `AMQERR01.LOG` contains text formatted messages in any language, converts messages into US English:

```
cat AMQERR01.LOG | mqrc -d En_US -i text -o text -
```

## Multi **rctdmqimg (record media image)**

Write the image of an object or group of objects to the log for media recovery.

### Purpose

Use the **rctdmqimg** command to write an image of an object, or group of objects, to the log for use in media recovery. This command can be used only when using linear logging. See [Types of logging](#) for more information about linear logging. Use the associated command **rctrmqobj** to re-create the object from the image.

Before IBM MQ 9.1.0, or when using **LogManagement=Manual**, the command does not run automatically as it must be run in accordance with, and as determined by, the usage of each individual customer of IBM MQ.

After IBM MQ 9.1.0, when using **LogManagement=Automatic** or **Archive**, the queue manager automatically records media images, however **rctdmqimg** can also be run manually as well, if required.

Running **rctdmqimg** moves the log sequence number (LSN) forwards and frees up old log files for archival or deletion.

When determining when and how often to run **rctdmqimg**, consider these factors:

### Disk space

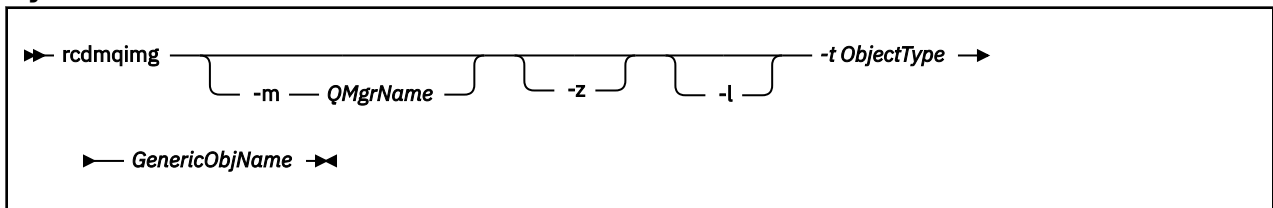
If disk space is limited, regular running of **rctdmqimg** releases log files for archive or deletion.

### Impact on normal system performance

**rctdmqimg** activity can take a long time if the queues on the system are deep. At this time, other system usage is slower and disk utilization increases because data is being copied from the queue files to the logs. Therefore, the ideal time to run **rctdmqimg** is when the queues are empty and the system is not being heavily used.

You use this command with an active queue manager. Further activity on the queue manager is logged so that, although the image becomes out of date, the log records reflect any changes to the object.

### Syntax



### Required parameters

#### GenericObjName

The name of the object to record. This parameter can have a trailing asterisk to record that any objects with names matching the portion of the name before the asterisk.

This parameter is required unless you are recording a queue manager object or the channel synchronization file. Any object name you specify for the channel synchronization file is ignored.



#### -t ObjectType

The types of object for which to record images. Valid object types are:

Object Type	Description
<b>all</b> and <b>*</b>	All the object types; <b>ALL</b> for objtype and <b>*</b> for GenericObjName
<b>authinfo</b>	Authentication information object, for use with TLS channel security

Table 78. Valid object types (continued)

Object Type	Description
<b>channel</b> or <b>chl</b>	Channels
<b>clntconn</b> or <b>clcn</b>	Client connection channels
<b>catalog</b> or <b>ctlg</b>	An object catalog
<b>listener</b> or <b>lstr</b>	Listeners
<b>namelist</b> or <b>nl</b>	Namelists
<b>process</b> or <b>prcs</b>	Processes
<b>queue</b> or <b>q</b>	All types of queue
<b>qalias</b> or <b>qa</b>	Alias queues
<b>qlocal</b> or <b>ql</b>	Local queues
<b>qmodel</b> or <b>qm</b>	Model queues
<b>qremote</b> or <b>qr</b>	Remote queues
<b>qmgr</b>	Queue manager object
<b>service</b> or <b>srvc</b>	Service
<b>syncfile</b>	Channel synchronization file.
<b>topic</b> or <b>top</b>	Topics

**Note:**   When using IBM MQ for AIX or Linux systems, you must prevent the shell from interpreting the meaning of special characters, for example, an asterisk (\*). How you do this depends on the shell you are using, but might involve the use of single quotation marks ('), double quotation marks ("), or a backslash (\).

## Optional parameters

### **-m QMgrName**

The name of the queue manager for which to record images. If you omit this parameter, the command operates on the default queue manager.

### **-z**

Suppresses error messages.

### **-l**

Writes messages containing the names of the oldest log files required to restart the queue manager and to perform media recovery. The messages are written to the error log and the standard error destination. (If you specify both the **-z** and **-l** parameters, the messages are sent to the error log, but not to the standard error destination.)

When issuing a sequence of **rcdmqimg** commands, include the **-l** parameter only on the last command in the sequence, so that the log file information is gathered only once.

## Return codes

Table 79. Return code identifiers and descriptions

Return code	Description
0	Successful operation
26	Queue manager running as a standby instance.

Table 79. Return code identifiers and descriptions (continued)

Return code	Description
28	Object not media recoverable.
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
58	Inconsistent use of installations detected
68	Media recovery not supported
69	Storage not available
71	Unexpected error
72	Queue manager name error
119	User not authorized
128	No objects processed
131	Resource problem
132	Object damaged
135	Temporary object cannot be recorded

### When are log extents deleted

Log extents are only deleted when the queue manager determines that they can be deleted. Note that log extents are not deleted immediately after recording the media image.

For example, if the starting media extent is 04, the queue manager does not delete this extent until the extent number moves forward, and the queue manager might or might not delete extents 01 to 04.

The logger event messages, and the IBM MQ queue manager error logs, show the log extents required for queue manager restart and media recovery.

### Examples

The following command records an image of the queue manager object `saturn.queue.manager` in the log.

```
rcdmqimg -t qmgr -m saturn.queue.manager
```

### Related commands

Table 80. Related command names and descriptions

Command	Description
<a href="#">rcrmqobj</a>	Re-create a queue manager object

#### **rcrmqobj (re-create object)**

Re-create an object, or group of objects, from their images contained in the log.

### Purpose

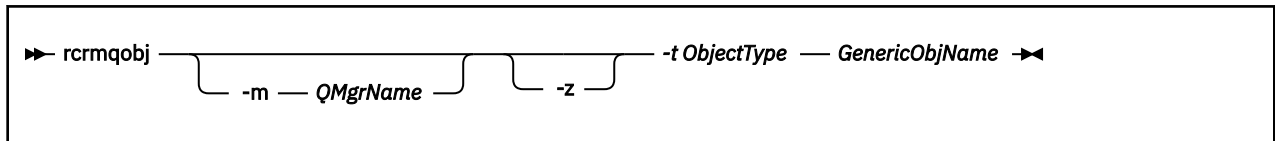
Use the **rcrmqobj** command to re-create an object, or group of objects, from their images.

**Note:** Use this command on a running queue manager.

- With *ObjectType* argument of `clchltab` or `syncfile`, this command re-creates the object files from the internal queue manager state.
- For other *ObjectType* arguments, the command can only be used when the queue manager is configured to use linear logging. Use the associated command, `rcdmqimg`, to record the object images to the log. The object is re-created from images in the log.

All activity on the queue manager after the image was recorded is logged. To re-create an object, replay the log to re-create events that occurred after the object image was captured.

## Syntax



## Required parameters

### GenericObjName

The name of the object to re-create. This parameter can have a trailing asterisk to re-create any objects with names matching the portion of the name before the asterisk.

This parameter is required, unless the object type is the channel synchronization file; any object name supplied for this object type is ignored.



### -t *ObjectType*

The types of object to re-create. Valid object types are:

<i>Table 81. Valid object types.</i>	
Object Type	Description
* or all	All object types
authinfo	Authentication information object, for use with TLS channel security
channel or chl	Channels
clntconn or clcn	Client connection channels
clchltab	Client channel table
comminfo	Communication information object
listener or lstr	Listener
namelist or nl	Namelists
process or prcs	Processes
queue or q	All types of queue
qalias or qa	Alias queues
qlocal or ql	Local queues
qmodel or qm	Model queues
qremote or qr	Remote queues
service or srvc	Service

Table 81. Valid object types. (continued)

Object Type	Description
syncfile	Channel synchronization file. You can use this option when circular logs are configured but the syncfile fails if the channel scratchpad files, which are used to rebuild syncfile, are damaged or missing. You might want to do this if your system has reported the error message <a href="#">AMQ7353 (krcE_SYNCFILE_UPDATE_FAILED)</a> .
topic or top	Topics

**Note:**   When using IBM MQ for AIX or Linux systems, you must prevent the shell from interpreting the meaning of special characters, for example, an asterisk (\*). How you do this depends on the shell you are using, but might involve the use of single quotation marks ('), double quotation marks ("), or a backslash (\).

### Optional parameters

**-m *QMgrName***

The name of the queue manager for which to re-create objects. If omitted, the command operates on the default queue manager.

**-z**

Suppresses error messages.

### Return codes

Table 82. Return code identifiers and descriptions

Return code	Description
0	Successful operation
26	Queue manager running as a standby instance.
28	Object not media recoverable.
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
58	Inconsistent use of installations detected
66	Media image not available
68	Media recovery not supported
69	Storage not available
71	Unexpected error
72	Queue manager name error
119	User not authorized
128	No objects processed
135	Temporary object cannot be recovered
136	Object in use

## Examples

1. The following command re-creates all local queues for the default queue manager:

```
rcrmqobj -t ql *
```

2. The following command re-creates all remote queues associated with queue manager store:

```
rcrmqobj -m store -t qr *
```

## Related commands

Table 83. Related command names and descriptions

Command	Description
<a href="#">rcdmqimg</a>	Record an object in the log

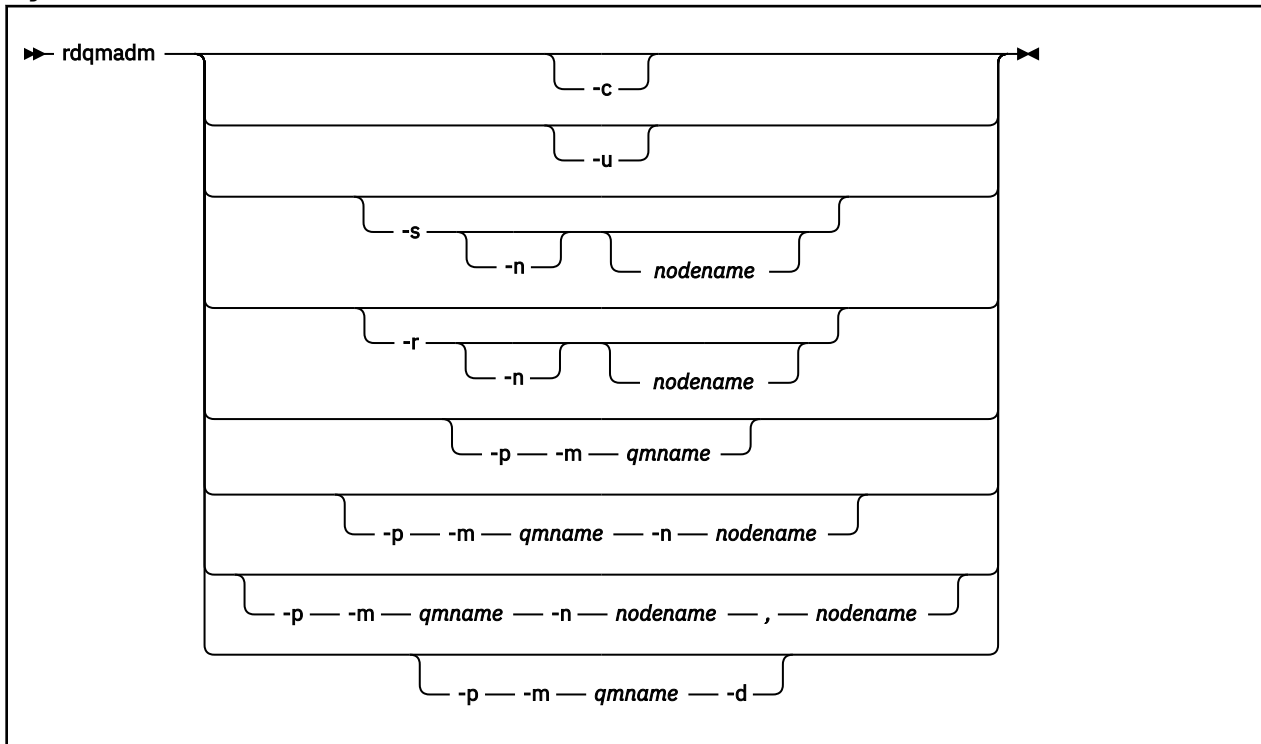
### Linux **rdqmadm (administer replicated data queue manager cluster)**

Administer the cluster in a high availability RDQM configuration.

#### Purpose

Use the **rdqmadm** command to administer the Pacemaker cluster used in RDQM high availability configurations. (This command is not required for disaster recovery RDQM configurations.)

#### Syntax





## Optional parameters

**-c**

Initialize the Pacemaker cluster, using the settings specified in the `/var/mqm/rdqm.ini` file. The same command must be run on each of the three nodes by the `root` user. (You can also run this command as a user in the `mqm` group if you have configured `sudo`, see [Requirements for RDQM HA solution](#).) The command fails if the node is already part of a Pacemaker cluster. A node cannot be a member of two Pacemaker clusters.

**-u**

Delete the Pacemaker cluster configuration. The same command must be run on each of the three nodes by the `root` user. (You can also run this command as a user in the `mqm` group if you have configured `sudo`, see [Requirements for RDQM HA solution](#).) The Pacemaker cluster configuration cannot be deleted if any replicated data queue managers (RDQMs) exist.

**-s [-n *nodename*]**

Suspend the local node (or the specified node if the `-n nodename` argument is supplied). The command can be run on any of the three nodes by a user in the `haclient` group, or by `root`. The node is taken offline. Any replicated data queue managers (RDQMs) running on that node are stopped and restarted on an active node. Queue manager data does not replicate to the offline node. The command fails if the specified node is the last active node.

**-r [-n *nodename*]**

Resume the local or specified node. The command can be run on any of the three nodes by a user in the `haclient` group, or by `root`. The node is brought online. If the node is the preferred location for any replicated data queue managers (RDQMs), the queue managers are stopped and restarted on this node.

**-p -m *qmname* [-n *nodename*],*nodename*]**

Assign the local or specified node as the Preferred Location for the named queue manager. If the Pacemaker cluster is in a normal state and the Preferred Location is not the current primary node, the queue manager is stopped and restarted on the new Preferred Location. You can specify a comma-separated list of two node names to assign a second preference of Preferred Location.

**-p -m *qmname* -d**

Clear the Preferred Location so that the queue manager does not automatically return to a node when it is restored.

Linux

V 9.3.0

## **rdqmclean (clear failed resource actions)**

You use the **rdqmclean** command to clear failed resource actions from RDQM HA configurations.

### Purpose

Failed resource actions arise when the Pacemaker component of an RDQM high availability configuration encounters some problem with a resource on one of the nodes in an HA group. Some failed resource actions prevent the resource from running on one or all nodes and must be cleared before Pacemaker can restart the resource. You must also resolve the cause of the resource failure.

### Syntax



## Optional parameters

**-m *qmname***

Specify the name of the queue manager for which you are clearing failed resource actions.

**-a**

Clear all failed resource actions in the RDQM HA configuration.

## Related concepts

[Failed resource actions](#)

## Related tasks

Linux

[Viewing RDQM and HA group status](#)

Linux

## rdqmdr (manage DR RDQM instances)

Change a primary disaster recovery replicated data queue manager (DR RDQM) to a secondary instance, or change a secondary instance to a primary.

## Purpose

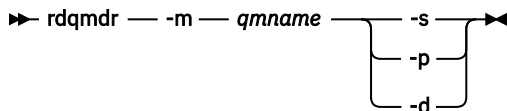
Use the **rdqmdr** command to control whether an instance of a DR RDQM has the primary or secondary role.

You can also use **rdqmdr** on the node where you created a primary DR RDQM to retrieve the command that you need to create the secondary instance on the recovery node.

You must be `root` or a user in the `mqm` group with `sudo` privileges to use this command.

## Syntax

```
rdqmdr -m qmname [-s] [-p] [-d]
```



## Parameters

### -m *qmname*

Specify the name of the DR RDQM that you are issuing the command for.

### -s

Specify `-s` to make a DR RDQM that is currently in the primary role into the secondary.

### -p

Specify `-p` to make a DR RDQM that is currently in the secondary role into the primary. This command fails if the primary instance of the queue manager is still running and the DR replication link is still functioning.

### -d

Specify `-d` to return the **crtmqm** command required to create a secondary instance of the specified DR RDQM.

Linux

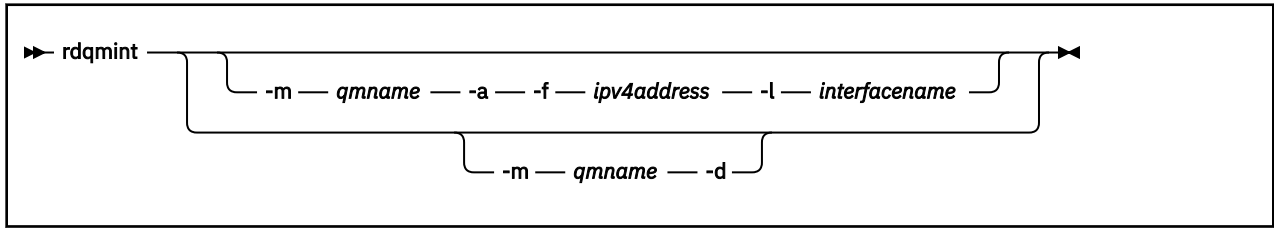
## rdqmint (add or delete floating IP address for RDQM)

Add or delete the floating IP address used to connect to a high availability replicated data queue manager (HA RDQM).

## Purpose

Use the **rdqmint** command to add or delete the floating IP address that is used to connect to an HA RDQM regardless of which node in the high availability (HA) group is actually running the RDQM. (This command is not applicable to disaster recovery RDQM configurations.)

## Syntax



## Optional parameters

### **-m *qmname***

Specify the name of the RDQM for which you are adding or deleting a floating IP address.

### **-a**

Specify this option to add a floating IP address.

### **-d**

Specify this option to delete a floating IP address.

### **-f *ipv4address***

The IP address in dot decimal format.

The floating IP address must be a valid IPv4 address that is not already defined on any HA node, and it must belong to the same subnet as the static IP addresses defined for the local interface.

### **-l *interfacename***

The name of the physical interface that the floating IP address is bound to.

## Examples

To specify a floating IP address for the queue manager RDQM1, enter the following command:

```
rdqmint -m RDQM1 -a 192.168.7.5 -l MQIF
```

To delete the floating IP address for the queue manager RDQM1, enter the following command:

```
rdqmint -m qmname -d
```

## Linux **rdqmstatus (display RDQM status)**

Display the status of all the replicated data queue managers (RDQMs) on a node or detailed status of specified individual RDQMs. You can also display the online/offline status of the nodes in an HA group.

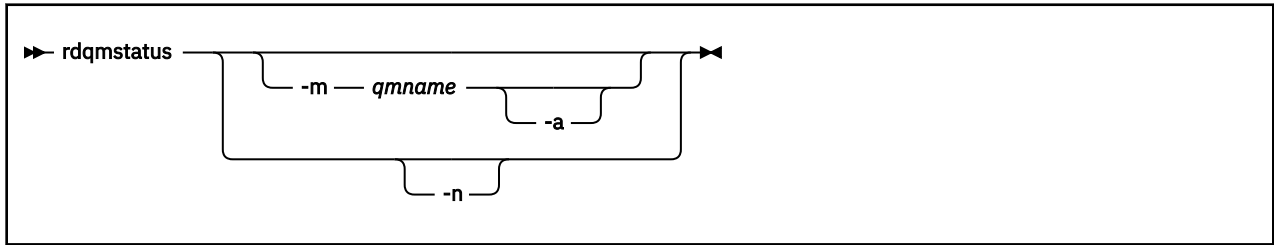
## Purpose

Use the **rdqmstatus** command on its own to view a summary of the status of all RDQM queue managers on a node. You can specify a queue manager name to view detailed status for that RDQM, including details of failed resource actions. You can also view the availability status of all nodes in an HA group.

You can enter the command on any node in an HA group, or either node in a DR pair, or any node in a DR/HA configuration.

For examples of the output of the **rdqmstatus** command, see [Viewing RDQM and HA group status](#), and [Viewing DR RDQM status](#), and [Viewing DR/HA RDQM and HA group status](#).

## Syntax



## Optional parameters

### **-m *qmname***

Specify the name of the RDQM for which you are requesting status.

### **V 9.3.0 -a**

Optionally use with `-m qmname` to view failed resource actions associated with the specified queue manager (see [Failed resource actions](#)).

### **-n**

Specify `-n` to list the three nodes in the HA group, and their current online or offline status.

## Related tasks

**Linux** [Viewing RDQM and HA group status](#)

**Linux** [Viewing DR RDQM status](#)

**Linux** [Viewing DR/HA RDQM and HA group status](#)

## **Multi** **rmvmqinf (remove configuration information)**

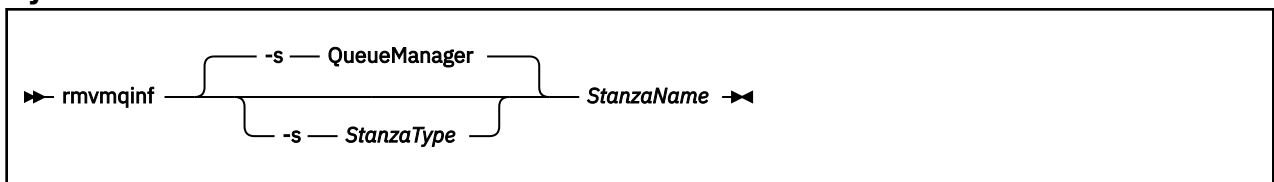
Remove IBM MQ configuration information (AIX, Linux, and Windows only).

## Purpose

Use the **rmvmqinf** command to remove IBM MQ configuration information.

You must use the **rmvmqinf** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmq -o` installation command.

## Syntax



## Required parameters

### **StanzaName**

The name of the stanza. That is, the value of the key attribute that distinguishes between multiple stanzas of the same type.

## Optional parameters

### **-s *StanzaType***

The type of stanza to remove. If omitted, a QueueManager stanza is removed.

The only supported value of *StanzaType* is QueueManager.

## Return codes

Table 84. Return code identifiers and descriptions

Return code	Description
0	Successful operation
5	Queue manager is running
26	Queue manager is running as a standby instance
39	Bad command line parameters
44	Stanza does not exist
49	Queue manager is stopping
58	Inconsistent use of installations detected
69	Storage is not available
71	Unexpected error
72	Queue manager name error

### Example

```
rmvmqinf QM.NAME
```

### Usage notes

Use `rmvmqinf` to remove an instance of a multi-instance queue manager.

To use this command you must be an IBM MQ administrator and a member of the `mqm` group.

### Related commands

Table 85. Related command names and descriptions

Command	Description
<a href="#">“addmqinf (add configuration information)” on page 21</a>	Add queue manager configuration information
<a href="#">“dspmqinf (display configuration information)” on page 86</a>	Display queue manager configuration information

## **rsvmqtrn (resolve transactions)**

Resolve in-doubt and heuristically completed transactions

### Purpose

The **rsvmqtrn** command is used to resolve two different transaction states.

#### in-doubt transactions

Use the **rsvmqtrn** command to commit or back out internally or externally coordinated in-doubt transactions.

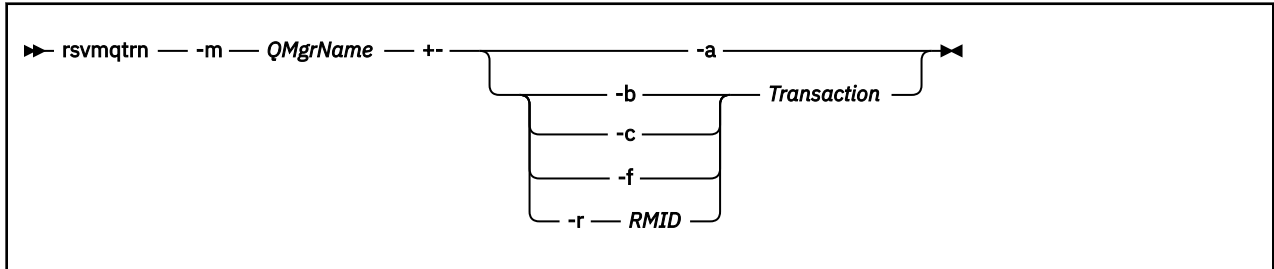
**Note:** Use this command only when you are certain that transactions cannot be resolved by the normal protocols. Issuing this command might result in the loss of transactional integrity between resource managers for a distributed transaction.

## heuristically completed transactions

Use the **rsvmqtrn** command with the **-f** parameter for IBM MQ to remove all information about externally coordinated transactions that were previously resolved manually using the **rsvmqtrn** command, but the resolution has not been acknowledged by the transaction coordinator using the **xa-forget** command. Transactions that are manually resolved by a resource manager and unacknowledged by the transaction manager, are known as *heuristically completed* transactions by X/Open.

**Note:** Only use the **-f** option if the external transaction coordinator is permanently unavailable. The queue manager, as a resource manager, remembers the transactions that are committed or backed out manually by the **rsvmqtrn** command.

## Syntax



## Required parameters

### -m *QMGrName*

The name of the queue manager.



**Attention:** The following parameters are mutually exclusive. You must supply the **-a** parameter on its own, or one of the other parameters together with its transaction number.

## Optional parameters

### -a

The queue manager resolves all internally coordinated, in-doubt transactions (that is, all global units of work).

### -b

Backs out the named transaction. This flag is valid for externally coordinated transactions (that is, for external units of work) only.

### -c

Commits the named transaction. This flag is valid for externally coordinated transactions (that is, external units of work) only.

### -f

Forgets the named heuristically completed transaction. This flag is valid only for externally coordinated transactions (that is, external units of work) that are resolved, but unacknowledged by the transaction coordinator.

**Note:** Use only if the external transaction coordinator is never going to be able to acknowledge the heuristically completed transaction. For example, if the transaction coordinator has been deleted.

### -r *RMID*

The participation of the resource manager in the in-doubt transaction can be ignored. This flag is valid for internally coordinated transactions only, and for resource managers that have had their resource manager configuration entries removed from the queue manager configuration information.

**Note:** The queue manager does not call the resource manager. Instead, it marks the participation of the resource manager in the transaction as being complete.

## Transaction

The transaction number of the transaction being committed or backed out. Use the **dspmqrtrn** command to find the relevant transaction number. This parameter is required with the **-b**, **-c**, **-f**, and **-r RMID** parameters and must be the last parameter when used.

## Return codes

Table 86. Return code identifiers and descriptions

Return code	Description
0	Successful operation
26	Queue manager running as a standby instance.
32	Transactions could not be resolved
34	Resource manager not recognized
35	Resource manager not permanently unavailable
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
58	Inconsistent use of installations detected
69	Storage not available
71	Unexpected error
72	Queue manager name error
85	Transactions not known


## Related commands

Table 87. Related command names and descriptions

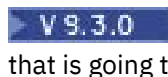
Command	Description
<a href="#">dspmqrtrn</a>	Display list of prepared transactions

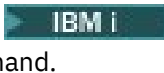
### **runamscred: protect AMS keywords**

The **runamscred** command protects passwords inside AMS configuration files.

 There are two variants of this command:

- An MQI variant located in <IBM MQ installation root>/bin
- A Java variant located in <IBM MQ installation root>/java/bin

 When using **runamscred** to protect AMS keywords, use the same variant for the AMS client that is going to use the AMS keywords. For example, use the Java variant to protect Java keywords.

**Note:**  When running on IBM MQ for IBM i, use **runamscri** as the name of this control command.

## Purpose

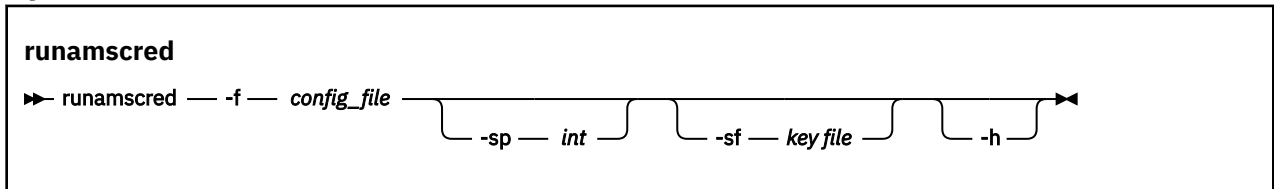
The **runamscred** command uses the encryption key contained in the file, indicated by one of four options. In order of priority, these are the:

1. **-sf** parameter.
2. **MQS\_AMSCRED\_KEYFILE** environment variable.
3. **amscred.keyfile** parameter in the configuration file .
4. Default initial key file if none of the above options is specified.



**Attention:** **V 9.3.0** From IBM MQ 9.3.0, you should not use the default initial key.

## Syntax



## Parameters

### **-f config\_file**

Required. Path to the keystore configuration file to protect

### **-sp int**

Optional. Algorithm to use for protecting passwords. The value can be:

**0**

Use the deprecated credentials protection method.

**V 9.3.0** Not applicable for MQI clients

**1**

**V 9.3.0** The IBM MQ 9.2.0 password protection algorithm.

**V 9.3.0** **2**

Default: Use the more secure credentials protection method.

### **-sf keyfile**

Optional. Path to a file containing the initial key.

### **-h**

Optional. Displays command syntax.

## Examples

**Linux** **AIX** To encrypt a password in the `/home/alice/keystore.conf` configuration file using the new algorithm, and store it in the new format, issue the following command:

```
runamscred -f /home/alice/keystore.conf
```

**Linux** **AIX** To encrypt a password in the `/home/alice/keystore.conf` configuration file, using an initial key in the `/etc/secure/alice_initial.key` file, together with the new algorithm, and store it in the new format, issue the following command:

```
runamscred -sf /etc/secure/alice_initial.key -f /home/alice/keystore.conf
```



**Windows** To encrypt a password in the C:\Users\alice\keystore.conf configuration file using the new algorithm, and store it in the new format, issue the following command:

```
runamscred -f C:\Users\alice\keystore.conf
```

**Windows** To encrypt a password in the C:\Users\alice\keystore.conf configuration file, using an initial key in the C:\secure\alice\_initial.key file, together with the new algorithm, and store it in the new format, issue the following command:

```
runamscred -sf C:\secure\alice_initial.key -f C:\Users\alice\keystore.conf
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related tasks

[Protecting passwords in IBM MQ configuration files](#)

## Related information

[Setting up AMS password protection for configuration files](#)

## **Future** **ALW** **runmqakm (manage keys, certificates, and key requests)**

You can use the **runmqakm** command line options to manage keys, certificates, and certificate requests. **runmqakm** provides functions similar to those of **gskitcapicmd**.

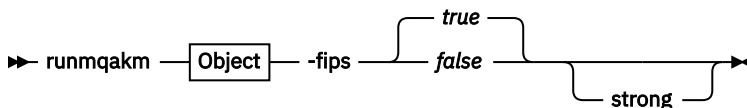
## Purpose

Use the **runmqakm** command to support:

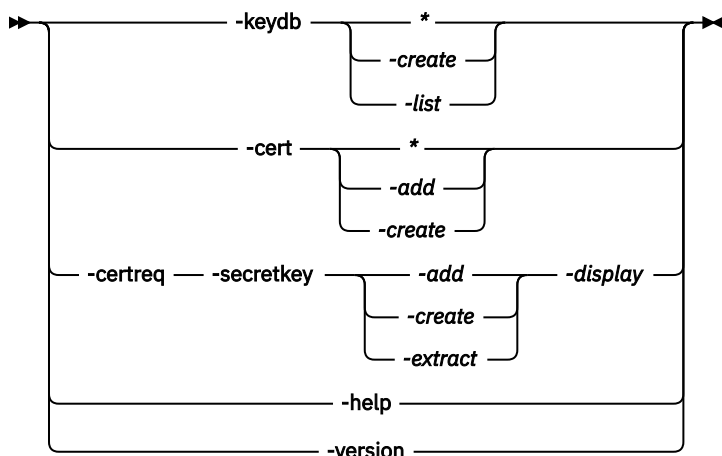
- The creation of certificates and certificate requests with Elliptic Curve public keys
- Stronger encryption of the key repository file than the **runmqckm** command through the **-strong** parameter

**runmqakm** has been certified as FIPS 140-2 compliant, and can be configured to operate in a FIPS compliant manner, using the **-fips** parameter.

## Syntax



## Object



## Required parameters

### Object

One of -keydb, -cert, -certreq, -help, and -version

The -keydb, -cert, and -certreq options also require an action.

See [Commands for CMS or PKCS#12 key databases on AIX, Linux, and Windows](#) and [Commands for cryptographic device operations on AIX, Linux, and Windows](#) for the possible actions.

See [runmqckm and runmqakm options on AIX, Linux, and Windows](#) for further information on the options available.

### -fips

Enable or disable forced Federal Information Processing Standards (FIPS) mode. The program runs in FIPS mode by default.

In FIPS mode, **gsk8capicmd** initializes the underlying cryptographic provider in FIPS mode so that it only uses algorithms that have been FIPS 140-2 validated.

If however the **-fips** true is set and the provider cannot be initialized in FIPS mode the **gsk8capicmd** operation fails. If FIPS mode is not forced and the provider cannot be initialized in FIPS mode then the utility falls back to a non **fips** mode of operation.

## Optional Parameters

### -strong

Control what kind of passwords are generated automatically by GSKit or enforce when a password is set, that it meets some requirements.

The minimum requirements for a password are as follows:

- The minimum password length is 14 characters.
- A password must have at least one lower case character, one uppercase character, and one digit or special character (for example, \*\$#% etc). A space is classified as a special character.
- Each character must not occur more than three times in a password.
- No more than two consecutive characters of the password can be identical.
- All characters are in the standard ASCII printable character set within the range from 0x20 to 0x7E inclusive.

## Deprecated Linux MQ Adv. VUIE runmqbcb (run IBM MQ Bridge to blockchain)

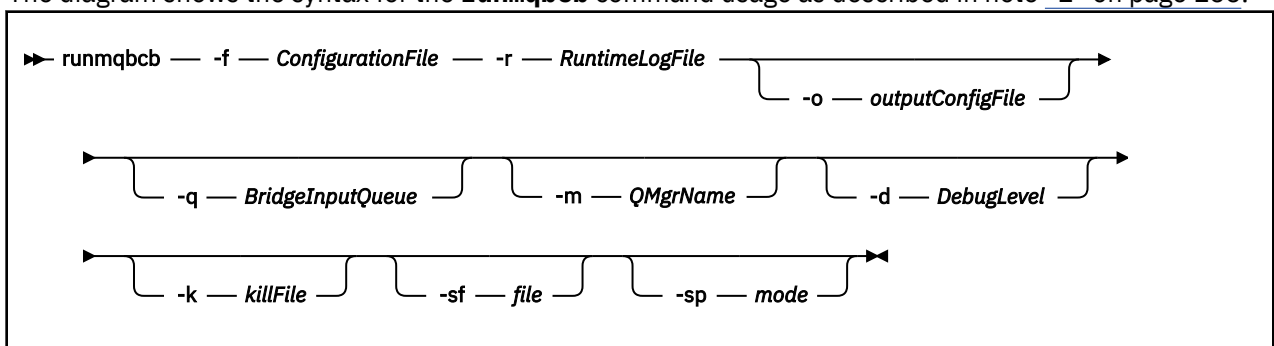
Configure and run the IBM MQ Bridge to blockchain on a Hyperledger Fabric network.

### Notes:

- **Deprecated** The IBM MQ Bridge to blockchain is deprecated across all releases from November 22 2022 (see [US Announcement letter 222-341](#)). Blockchain connectivity can be achieved with IBM App Connect or through App Connect capabilities available with IBM Cloud Pak® for Integration.
- **Removed** **V 9.3.2** For Continuous Delivery, the IBM MQ Bridge to blockchain is removed from the product at IBM MQ 9.3.2.
- **LTS** IBM intends to remove the capability from Long Term Support releases in upcoming fix packs. If you have applications that will be impacted by this change, please contact IBM Support.
- [Syntax](#)
- [Usage notes](#)
- [Command line parameters](#)
- [Configuration parameters](#)

## Syntax

The diagram shows the syntax for the **runmqbcb** command usage as described in note “1” on page 155.



## Usage notes

There are two available authentication mechanisms for the bridge to connect to Hyperledger Fabric, both of which require that you configure a username. This username will be associated with any operations processed through the IBM MQ Bridge to blockchain.

The first approach allows a Wallet (file) to be supplied from the administrator. The Wallet is a container holding certificates and so on.

The second approach is based on an administrator just providing certificates to you and not a standalone wallet. The configuration then requires the location of the certificate (typically a PEM file), along with a password to access it, and an associated organization name.

1. You can run the **runmqbcb** command to start the IBM MQ Bridge to blockchain and connect to Hyperledger Fabric and IBM MQ.

When the connections are made, the bridge is ready to receive and process request messages that are put on the queue manager input queue, send the correctly formatted queries and updates to the blockchain network, receive, process and put replies from the blockchain to the reply queue.

```
runmqbcb -f ConfigFile -q BridgeInputQueue -m QMgrName -d DebugLevel -k killFile -r
RuntimeLogFile
```

When you use the command for runtime processing, the required parameters are **-f**, with the name of the previously created configuration file, and **-r** with the name of the log file. When the other command parameters are also given on the command line, they override the values in the configuration file. The same configuration file can be used by multiple bridges.

2. You can also use the **runmqbcb** command to generate a configuration file that is used to define the parameters that are needed for the bridge to connect to Hyperledger Fabric and IBM MQ.

When you are creating the configuration file, the **-f** parameter is optional.

```
runmqbcb -f inputConfigFile -o outputConfigFile [-b]
```

When you run the command in this way, you are prompted to enter values for each of the configuration parameters. To keep an existing value press **Enter**. To remove an existing value press **Space**, then **Enter**. For more information, see [“Configuration parameters” on page 157](#).

3. From IBM MQ 9.2.0, the usage of the enhanced protection parameters is as follows:

#### **Configuration mode**

Newly entered passwords are written to the output configuration file protected with the new key. Note that preexisting passwords are not changed in format.

A warning is issued when the default key is being used, that is, you have not provided a keyfile.

Provision of passwords in the batch configuration mode, using environment variables, continues unchanged; that is, the value of the environment variables is given in plain text.

#### **Runtime mode**

When a password is decrypted, warning messages are issued if the password is found to be in the old format, and the name of the parameter causing the warning is given in the message to encourage you to migrate. However, the bridge continues processing commands.

**Note:** The warning message is not issued if you have specified the **-sp 0** parameter on the command line, as you explicitly wanted to use old formats.

A warning is also issued when the default key is being used, that is, you have not provided a keyfile.

Errors occur if a password cannot be decrypted, for example, if you have specified the wrong keyfile.

## **Command line parameters**

### **-f ConfigurationFile**

Configuration file. The **-f** parameter is required when you are running the **runmqbcb** command to start the IBM MQ Bridge to blockchain, as described in usage note [“1” on page 155](#). You can optionally use the **-f** parameter to reuse some of the values from an existing *inputConfigFile*, as described in usage note [“2” on page 155](#), and also enter some of the new values. If you do not specify the **-f** parameter when you are creating the configuration file, all the values for the parameters you are prompted for are empty.

### **-r RuntimeLogFile**

Required. Location and name of the log file for trace information. You can specify the log file path and name in the configuration file or on the command line.

### **-o outputConfigFile**

New configuration file. When you run the command with the **-o** parameter, **runmqbcb** command loads existing configuration values from the **-f** file and prompts for new values for each configuration parameter.

### **-q BridgeInputQueue**

Name of the queue that the bridge waits for messages on.

### **-m QMgrName**

Queue manager name.

### **-d debugLevel**

Debug level, 1, or 2.

**1**

Terse debug information is displayed.

**2**

Verbose debug information is displayed.

**-k killFile**

A file to cause the bridge to exit. When you run the command with the **-k** parameter and specify a file, if the file exists, it causes the bridge program to exit. Using this file is an alternative way to stop the program when you don't want to use `Ctrl+C` or **kill** command. The file is deleted by the bridge on startup in case it exists. If the deletion fails, the bridge abends but monitors for the recreation of the file.

**-b**

Use environment variables during configuration.

**-sf file**

File containing password protection key.

**-sp mode**

Password protection mode. The values can be:

**2** `V9.3.0`

Use the latest password protection mode. This is the default value from IBM MQ 9.3.0.

**1**

Use the IBM MQ 9.2 password protection mode for compatibility with versions earlier than IBM MQ 9.3.0. This is the default value in versions earlier than IBM MQ 9.3.0.

**0**

Use the deprecated password protection mode that is compatible with versions earlier than IBM MQ 9.2.

## Configuration parameters

When you run the **runmqbc** command to create the configuration file, the parameters are stepped through in six groups. Passwords are obfuscated and are not displayed as you type. The generated configuration file is in JSON format. You must use the **runmqbc** command to create the configuration file. You cannot edit the passwords and security certificate information directly in the JSON file.

### Connection to queue manager

Parameters relating to the IBM MQ queue manager.

#### IBM MQ Queue manager

Required. The IBM MQ Advanced queue manager that you are using with the IBM MQ Bridge to blockchain.

#### Bridge input queue

`SYSTEM.BLOCKCHAIN.INPUT.QUEUE` is the default queue where applications put request messages, this can be overridden in the configuration file or on the **runmqbc** command line. User applications must have appropriate authorisation to put messages to this queue.

#### IBM MQ Channel

The bridge requires a `svrcon` channel to connect to a z/OS queue manager remotely.

#### IBM MQ Conname

Uses standard connection name format of "host(port), host(port)" to enable multiple destinations such as for multi-instance queue managers.

#### IBM MQ CCDT URL

If a TLS connection is required to the queue manager, you must use a JNDI or CCDT definition.

#### JNDI implementation class name

The class name of your JNDI provider. The "queue manager name" parameter refers to the connection factory name when you are using JNDI.

#### JNDI provider URL

The endpoint of your JNDI service.

**IBM MQ UserId**

The **UserId** that is running the bridge must have permission to set identity context on the messages it sends as replies, these have the requester **UserId** set in the message. The bridge user must therefore have appropriate access to put to the reply queue.

**IBM MQ Password**

Password for the IBM MQ **UserId** that the bridge is using.

**User identification**

Parameters relating to user authentication details that the bridge uses to connect to the Hyperledger Fabric REST server

**Userid**

The User Id provided by the bridge to Hyperledger Fabric must be known and authorized to connect to the Hyperledger Fabric endpoint, based on the user authentication configuration of the Hyperledger Fabric REST server.

**Password**

The password for the User Id that the bridge is using to connect to Hyperledger Fabric.

**API path for login**

The URL path to provide user credentials to the Hyperledger Fabric REST server. Note that this URL differs, depending on the type of security provider configured.

**Fabric server**

Attributes applicable to the Hyperledger Fabric server.

**Wallet**

A file containing credentials for the user, usually supplied by a Hyperledger Fabric administrator.

**User Name**

Mandatory parameter.

**User Certificate**

If no **Wallet** is provided, you must supply your certificate, private key and organization.

**User Private Key**

Your private key. You must supply this along with your certificate and organization if no **Wallet** has been provided.

**User Organization**

Your organization. You must supply this along with your certificate and private key if no **Wallet** has been provided.

**Network Configuration File**

A JSON-format file, usually supplied by the Hyperledger Fabric administrator or tooling that describes the various servers, addresses and so on. The file must exist.

**Commit Timeout**

Timeout for update operations in seconds.

The default value is 15 seconds.

**Discovery**

Whether to enable discovery of unknown networks that are not listed in the network configuration file.

The value can be *Y* or *N*.

**Updates sent to all peers**

Whether update responses are needed from all peers. or just one.

The value can be *Y* or *N*. The default value is *Y*.

**Updates sent to all organizations in the network**

Should updates be sent to all of the organizations listed in the configuration, or just to the specific MSPID organization.

The value can be *Y* for all organizations, or *N* for the specific organization. The default value is *N*.

### Location of PEM file for IBM Blockchain certificate

When using a TLS connection to the Hyperledger Fabric REST server, a single PEM file is used to hold the Hyperledger certificates to authenticate the bridge with the Hyperledger Fabric REST server. This PEM file must be copied to the system where the IBM MQ Bridge to blockchain is running, and specified in the configuration file.

### Certificate stores for TLS connections

Parameters relating to certificate stores for TLS connections.

#### Personal keystore for TLS certificates

Keystore for security certificates that are used for IBM MQ.

#### Keystore password

Password for the keystore.

#### Trusted store for signer certificates

If you do not add the trusted store, the personal keystore for TLS certificates is used.

#### Trusted store password

If the personal keystore for TLS certificates is used, this is the password for the keystore for TLS certificates.

#### Use TLS for MQ connection

The bridge can use TLS when it connects to the queue manager.

#### Timeout for Blockchain operations

If you don't provide a truststore parameter, the keystore is used for both roles. The stores can be the same as the one configured for the IBM MQ connection in the CCDT or JNDI.

### Behavior of bridge program

Parameters relating to the behavior of the IBM MQ Bridge to blockchain.

#### Required. Runtime logfile for copy of stdout/stderr

Path to and name of the log file for the tracing information.

The configuration is only read on startup of the bridge process. Changes to the configuration require a restart of the bridge.

## **Multi** `runmqccred (obfuscate passwords for mqccred exit)`

Obfuscate passwords in the `.ini` file used by the `mqccred` security exit.

### Purpose

Use the `runmqccred` command to process the `mqccred` exit `.ini` file to change all plain text passwords into an obfuscated form. This command should be run before using the `.ini` with the exit to ensure the exit runs successfully.

### Syntax

```
►► runmqccred -f -p
```

### Optional Parameters

#### **-f**

Specify a specific file to edit, other than the default file.

By default, the program locates the `.ini` file in the same way as the channel exit.

#### **-p**

By default the program fails with an error, if the filemode enables others to access the file you edited.

Use the `-p` flag to continue processing even when the error appears.

This might be necessary in situations where you might, for example, have mounted a UNIX file system onto your Windows machine using NFS, or some other protocol, and are trying to use the `.ini` file from there (perhaps to share the same `.ini` file across multiple accounts).

Since NFS does not support the Windows NT FS Access Control Lists, the exit would fail unless you bypass the permissions check.

## Usage notes

The **runmqccred** program locates the ini file in the same way as the channel exit. The program also writes console messages saying which file is being modified, and any success or failure status.

Note that the channel exit can work with either **Password** or **OPW** attributes, but the expectation is that you will protect passwords.

**Important:** The **runmqccred** program works only from IBM MQ 8.0 or later. You must run the program on an IBM MQ 8.0 or later system and then transfer the output `.ini` file manually to a system running a previous version if you want to use clients there.

By default the exit only works when there are no plain text passwords in the file. You can override this by using the **NOCHECKS SCYDATA** option.

The **runmqccred** program also checks that the `.ini` file does not have excessive permissions set that allow other users to access it. By default the program fails with an error if the filemode enables others to access it. Use the **-p** flag to continue processing even when the error appears.

The **runmqccred** program is installed in the following folder:

**Linux** **AIX** **AIX and Linux**  
The `MQ_INSTALLATION_PATH/usr/mqm/samp/mqccred/`

**Windows** **Windows platforms**  
The `MQ_INSTALLATION_PATH\Tools\c\Samples\mqccred\`

If the file permissions are not secure enough **runmqccred** produces this message:

```
Configuration file 'C:\Users\User1\mqc\mqccred.ini' is not secure.  
Other users may be able to read it. No changes have been made to the file.  
Use the -p option for runmqccred to bypass this error.
```

You can bypass this issue with the **-p** flag, but the exit will fail to run when put into production if you have not resolved this issue. When **runmqccred** runs successfully it informs you how many passwords have been obfuscated.

```
File 'C:\Users\User1\mqc\mqccred.in' processed successfully.  
Plaintext passwords found: 3
```

## **Multi** **runmqchi (run channel initiator)**

Run a channel initiator process to automate starting channels.

### Purpose

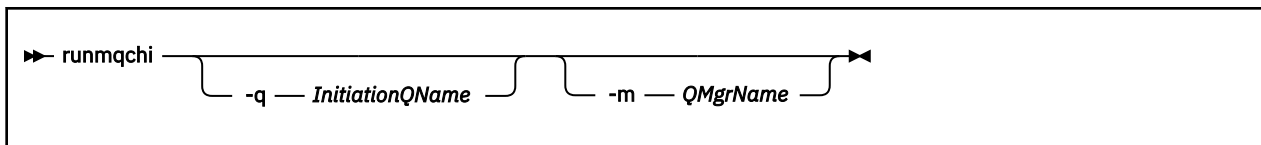
Use the **runmqchi** command to run a channel initiator process.

You must use the **runmqchi** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the **dspmqr -o installation** command.

The channel initiator is started by default as part of the queue manager.

### Syntax





## Optional parameters

### -q *InitiationQName*

The name of the initiation queue to be processed by this channel initiator. If you omit it, SYSTEM.CHANNEL.INITQ is used.

### -m *QMGrName*

The name of the queue manager on which the initiation queue exists. If you omit the name, the default queue manager is used.

## Return codes

Table 88. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

If errors occur that result in return codes of either 10 or 20, review the queue manager error log that the channel is associated with for the error messages, and the system error log for records of problems that occur before the channel is associated with the queue manager. For more information about error logs, see [Error log directories](#).

## Multi **runmqchl (run channel)**

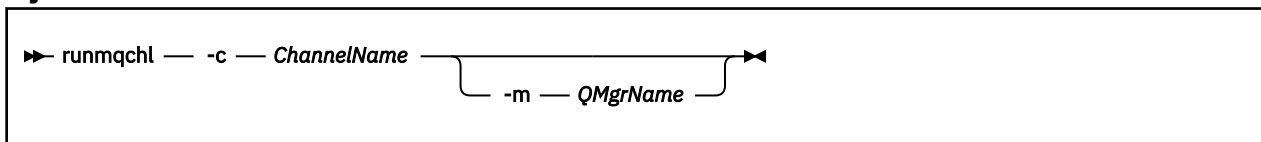
Start a sender or requester channel

### Purpose

Use the **runmqchl** command to run either a sender (SDR) or a requester (RQSTR) channel.

The channel runs synchronously. To stop the channel, issue the MQSC command **STOP CHANNEL**.

### Syntax



## Required parameters

### -c *ChannelName*

The name of the channel to run.

## Optional parameters

### -m *QMgrName*

The name of the queue manager with which this channel is associated. If you omit the name, the default queue manager is used.

## Return codes

Table 89. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

If return codes 10 or 20 are generated, review the error log of the associated queue manager for the error messages, and the system error log for records of problems that occur before the channel is associated with the queue manager.

## **runmqckm (manage keys, certificates, and key requests)**

You can use the **runmqckm** command line options to manage keys, certificates, and certificate requests. **runmqckm** provides functions similar to those of **iKeycmd**.

### Purpose

The **runmqckm** command supports the JKS and JCEKS key repository file formats.

If you require:

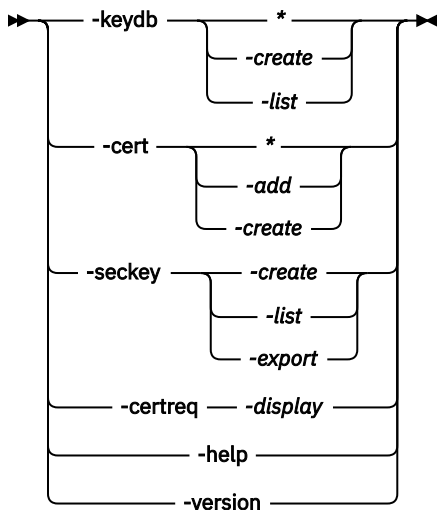
- The creation of certificates and certificate requests with Elliptic Curve public keys
- Stronger encryption of the key repository file than the **runmqckm** command through the **-strong** parameter

use the **runmqakm** command

### Syntax

►► runmqckm — Object ◀◀

### Object



## Required parameters

### Object

One of -keydb, -cert, -certreq, -seckey, -help, and -version

The -keydb, -cert, and -certreq options also require an action.

See [Commands for CMS or PKCS#12 key databases on AIX, Linux, and Windows](#) and [Commands for cryptographic device operations on AIX, Linux, and Windows](#) for the possible actions.

See [runmqckm and runmqakm options on AIX, Linux, and Windows](#) for further information on the options available.

## Multi **runmqdlq (run dead-letter queue handler)**

Start the dead-letter queue handler to monitor and process messages on the dead-letter queue.

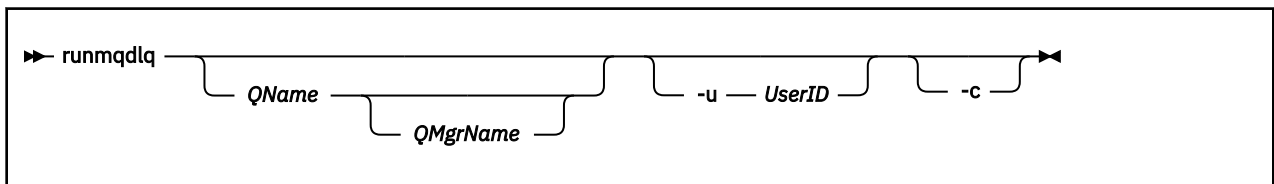
### Purpose

Use the **runmqdlq** command to start the dead-letter queue (DLQ) handler, which monitors and handles messages on a dead-letter queue.

Before IBM MQ 9.3.0, this command is used on servers. If you want client mode you should compile **amqsd1q** in client mode. See [The sample DLQ handler amqsd1q](#) for more information.

**V 9.3.0** From IBM MQ 9.3.0, you can use **runmqdlq** with the **-c** parameter to specify that it should connect to a queue manager by using a client connection.

### Syntax



### Description

Use the dead-letter queue handler to perform various actions on selected messages by specifying a set of rules that can both select a message and define the action to be performed on that message.

The **runmqdlq** command takes its input from `stdin`. When the command is processed, the results and a summary are put into a report that is sent to `stdout`.

By taking `stdin` from the keyboard, you can enter **runmqdlq** rules interactively.

By redirecting the input from a file, you can apply a rules table to the specified queue. The rules table must contain at least one rule.

If you use the DLQ handler without redirecting `stdin` from a file (the rules table), the DLQ handler reads its input from the keyboard:

- **Linux** **AIX** On AIX and Linux, the DLQ handler does not start to process the named queue until it receives an `end_of_file` (Ctrl+D) character.
- **Windows** On Windows, the DLQ handler does not start to process the named queue until you press the following sequence of keys: Ctrl+Z, Enter, Ctrl+Z, Enter.

For more information about rules tables and how to construct them, see [The DLQ handler rules table](#).

## Optional parameters

The MQSC command rules for comment lines and for joining lines also apply to the DLQ handler input parameters.

### QName

The name of the queue to be processed.

If you omit the name, the dead-letter queue defined for the local queue manager is used. If you enter one or more blanks ( ' '), the dead-letter queue of the local queue manager is explicitly assigned.

### QMgrName

The name of the queue manager that owns the queue to be processed.

If you omit the name, the default queue manager for the installation is used. If you enter one or more blanks ( ' '), the default queue manager for this installation is explicitly assigned.

### -u UserID

If you use the **-u** parameter to supply a user ID, you are prompted for a matching password.

If you have configured the `CONNAUTH AUTHINFO` record with `CHCKLOCL (REQUIRED)` or `CHCKLOCL (REQDADM)`, you must use the **-u** parameter otherwise you will not be able to start a dead-letter queue handler for your queue manager with **runmqdlq**.

If you specify this parameter and redirect `stdin`, a prompt will not be displayed and the first line of redirected input should contain the password.

### V9.3.0 -c

Modifies the **runmqdlq** command to connect to a queue manager by using a client connection. The client channel definitions used to connect to the queue manager are located using the following environment variables in this order of precedence: **MQSERVER**, **MQCHLLIB**, and **MQCHLTAB**.

This option requires the client to be installed. If it is not installed an error message reporting the missing client libraries is issued.



**Attention:** **V9.3.5** For Continuous Delivery from IBM MQ 9.3.5, the default permissions of **runmqdlq** have been changed to remove the `setuid` bit. When running **runmqdlq**, the tool runs under the context of the user that invokes the command.

For Long Term Support and Continuous Delivery before IBM MQ 9.3.5 **runmqdlq** is a `setuid` application that runs as the 'mqm' user regardless of which user started the application. If you use a CCDT file, the 'mqm' group must have permission to read the CCDT file as well as 'execute' permission on the directory structure. Failure to grant the correct permissions results in **runmqdlq** failing with an AMQ9516 error.

## Related concepts

[Dead-letter queues](#)

## Related tasks

[Invoking the dead-letter queue handler](#)

[Undelivered messages troubleshooting](#)

## **Windows** runmqdnm (run .NET monitor)

Start processing messages on a queue using the .NET monitor ( Windows only).

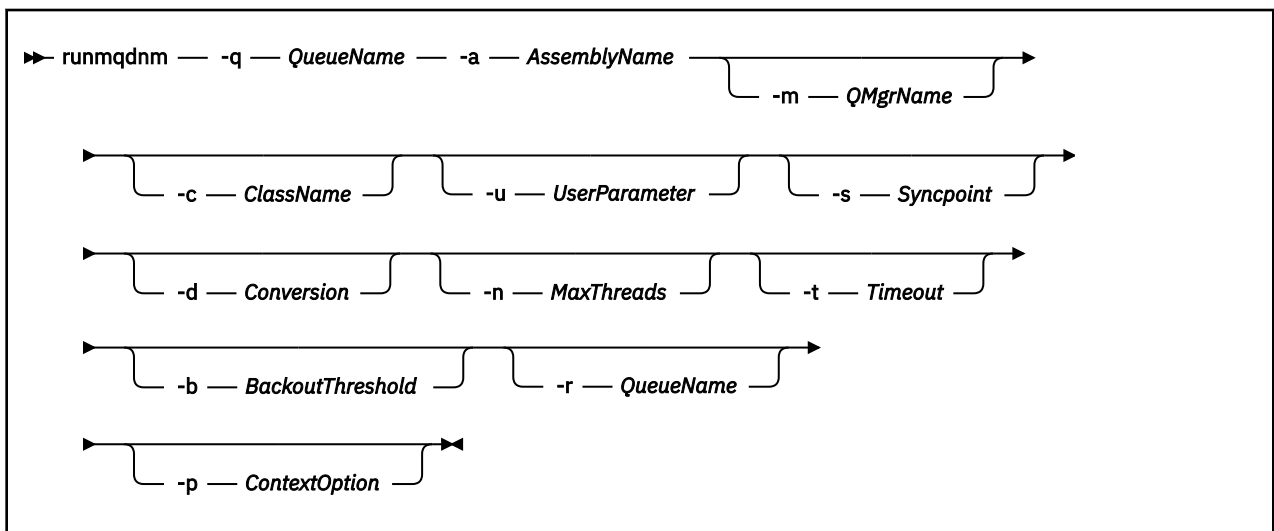
## Purpose

**Note:** The `runmqdnm` command applies to IBM MQ for Windows only.

`runmqdnm` can be run from the command line, or as a triggered application.

Use the `runmqdnm` control command to start processing messages on an application queue with a .NET monitor.

## Syntax



## Required parameters

### **-q QueueName**

The name of the application queue to monitor.

### **-a AssemblyName**

The name of the .NET assembly.

## Optional parameters

### **-m QMgrName**

The name of the queue manager that hosts the application queue.

If omitted, the default queue manager is used.

### **-c ClassName**

The name of the .NET class that implements the `IMQObjectTrigger` interface. This class must reside in the specified assembly.

If omitted, the specified assembly is searched to identify classes that implement the `IMQObjectTrigger` interface:

- If one class is found, then *ClassName* takes the name of this class.
- If no classes or multiple classes are found, then the .NET monitor is not started and a message is written to the console.

**-u UserData**

User-defined data. This data is passed to the Execute method when the .NET monitor calls it. User data must contain ASCII characters only, with no double quotation marks, NULLs, or carriage returns.

If omitted, null is passed to the Execute method.

**-s Syncpoint**

Specifies whether sync point control is required when messages are retrieved from the application queue. Possible values are:

Table 90. Syncpoint parameter values.	
Value	Description
YES	Messages are retrieved under sync point control (MQGMO_SYNCPOINT).
NO	Messages are not retrieved under sync point control (MQGMO_NO_SYNCPOINT).
PERSISTENT	Persistent messages are retrieved under sync point control (MQGMO_SYNCPOINT_IF_PERSISTENT).

If omitted, the value of *Syncpoint* is dependent on your transactional model:

- If distributed transaction coordination (DTC) is being used, then *Syncpoint* is specified as YES.
- If distributed transaction coordination (DTC) is not being used, then *Syncpoint* is specified as PERSISTENT.

**-d Conversion**

Specifies whether data conversion is required when messages are retrieved from the application queue. Possible values are:

Table 91. Conversion parameter values.	
Value	Description
YES	Data conversion is required (MQGMO_CONVERT).
NO	Data conversion is not required (no get message option specified).

If omitted, *Conversion* is specified as NO.

**-n MaxThreads**

The maximum number of active worker threads.

If omitted, *MaxThreads* is specified as 20.

**-t Timeout**

The time, in seconds, that the .NET monitor waits for further messages to arrive on the application queue. If you specify -1, the .NET monitor waits indefinitely.

If omitted when run from the command line, the .NET monitor waits indefinitely.

If omitted when run as a triggered application, the .NET monitor waits for 10 seconds.

**-b BackoutThreshold**

Specifies the backout threshold for messages retrieved from the application queue. Possible values are:

<i>Table 92. BackoutThreshold parameter values.</i>	
<b>Value</b>	<b>Description</b>
-1	The backout threshold is taken from the application queue attribute, BOTHRESH.
0	The backout threshold is not set.
1 or more	Explicitly sets the backout threshold.

If omitted, *BackoutThreshold* is specified as -1.

**-r QueueName**

The queue to which messages, with a backout count exceeding the backout threshold, are put.

If omitted, the value of *QueueName* is dependent on the value of the BOQNAME attribute from the application queue:

- If BOQNAME is non-blank, then *QueueName* takes the value of BOQNAME.
- If BOQNAME is blank, then *QueueName* is specified as the queue manager dead letter queue. If a dead letter queue has not been assigned to the queue manager, then backout processing is not available.

**-p ContextOption**

Specifies whether context information from a message that is being backed out is passed to the backed out message. Possible values are:

<i>Table 93. ContextOption parameter values.</i>	
<b>Value</b>	<b>Description</b>
NONE	No context information is passed.
IDENTITY	Identity context information is passed only.
ALL	All context information is passed.

If omitted, *ContextOption* is specified as ALL.

**Return codes**

*Table 94. Return code identifiers and descriptions*

<b>Return code</b>	<b>Description</b>
0	Successful operation
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
58	Inconsistent use of installations detected
71	Unexpected error
72	Queue manager name error
133	Unknown object name error

**Related tasks**

[Using the .NET monitor](#)

## runmqicred (protect IBM MQ client passwords)

The **runmqicred** command protects passwords that are used by the IBM MQ client libraries. For example, the TLS keystore password. It is also used to protect passwords that are used to protect log replication traffic for Native HA configurations.

### Purpose

The **runmqicred** command prompts for the password to be encrypted to be entered. The initial key that is used to encrypt the password can be specified in a file. The path to the file that contains the initial key is specified by using one of the following options, in order of priority:

1. The **-sf** parameter to the **runmqicred** command.
2. The `MQS_MQI_KEYFILE` environment variable.

If the initial key file is not specified by using either of these options, a default initial key is used to encrypt the password.



**Attention:** Do not use the default initial key as it does not protect passwords securely.

After the password is encrypted, **runmqicred** displays the encrypted password string.

Store the encrypted password in the appropriate property.

- For IBM MQ clients, store the encrypted password in either the appropriate property of the `mqclient.ini` file, or the `MQKEYRPWD` environment variable.
- **V 9.3.2** For Native HA configurations, store the encrypted password in the appropriate property of the **NativeHALocalInstance** stanza of the `qm.ini` file.

### Syntax

```

runmqicred -sf keyfile -sp protection_mode

```

### Optional Parameters

#### **-sf keyfile**

The path to the file that contains the initial key that is used to encrypt the password. If specified, the file must contain at least one character, and only one line.

If this parameter is not specified, a default initial key is used.

#### **-sp protection\_mode**

The password protection mode to be used by the command. One of the following values can be specified:

**1**

Use the IBM MQ 9.2.0 password protection algorithm.

**2**

Use the latest password protection mode. This mode is the most secure credentials protection method.

This value is the default.

### Examples

```
>runmqicred
```

5724-H72 (C) Copyright IBM Corp. 1994, 2024.  
 Credentials are encrypted using the default encryption key. For more secure protection of stored credentials, use a custom, strong encryption key. Enter password:



\*\*\*\*\*

<MQI>!2!+uIepF0e70/R7CUCe/46ToTo5MucJCWgLZKCSYwLix4=!+6AG1pYrphCo/dlfSt8N3g====

```
>runmqicred -sf InitialKey.file
```

5724-H72 (C) Copyright IBM Corp. 1994, 2024.

Enter password:

\*\*\*\*\*

<MQI>!2!STHVy96FWSEwPkwNQfR2Nuoe6/uWl/EAqy10jav9qs=!l+2y9yB/SjpszssrpGd+wJw=====

## Return codes

0

Command completed successfully.

1

Command completed unsuccessfully.

## Multi **runmqlsr (run listener)**

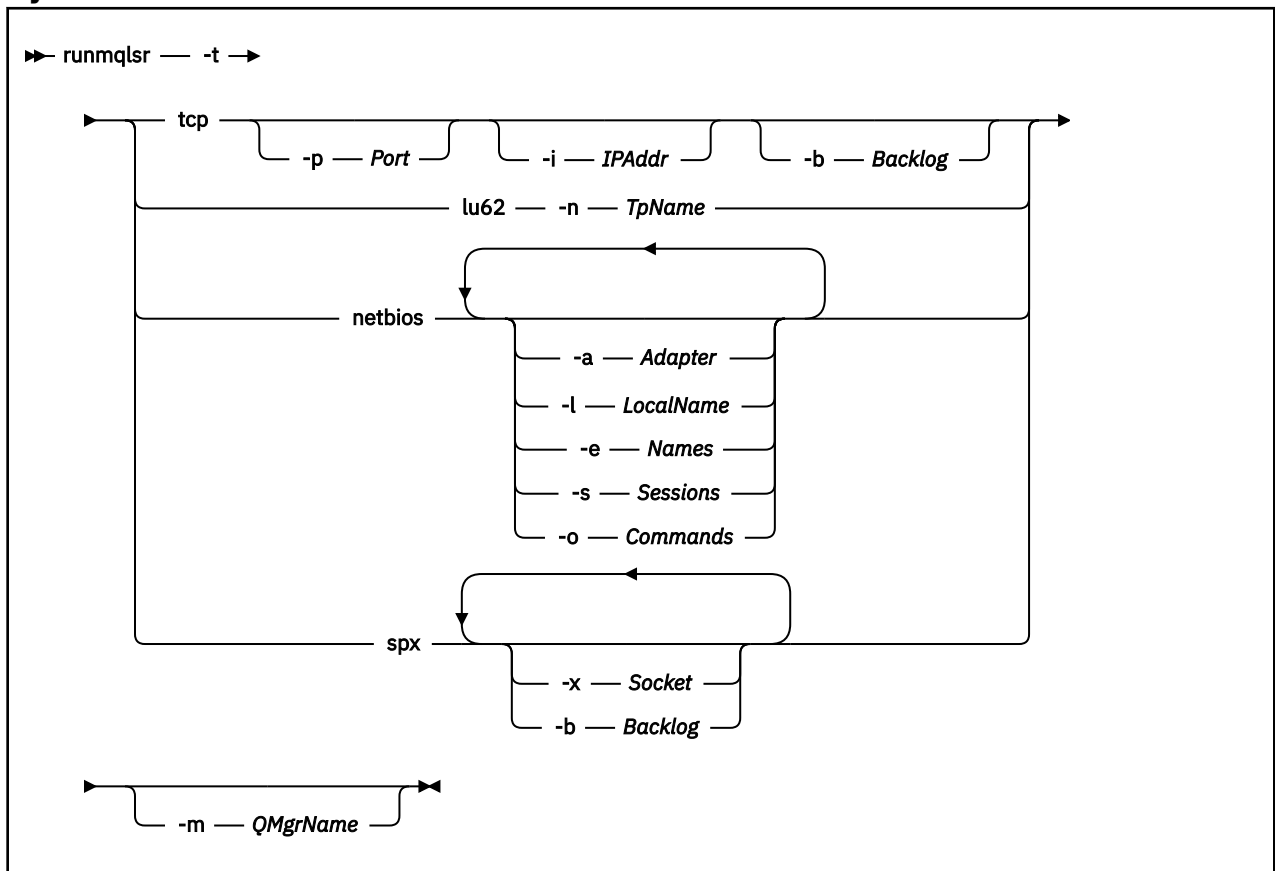
Run a listener process to listen for remote requests on various communication protocols.

## Purpose

Use the **runmqlsr** command to start a listener process.

This command is run synchronously and waits until the listener process has finished before returning to the caller.




## Syntax



## Required parameters

**-t**

The transmission protocol to be used:

Value	Description
tcp	Transmission Control Protocol / Internet Protocol (TCP/IP)
lu62	 SNA LU 6.2 ( Windows only)
netbios	 NetBIOS ( Windows only)
spx	 SPX ( Windows only)

## Optional parameters

**-p Port**

The port number for TCP/IP. This flag is valid for TCP only. If you omit the port number, it is taken from the queue manager configuration information, or from defaults in the program. The default value is 1414. It must not exceed 65535.

**-i IPAddr**

The IP address for the listener, specified in one of the following formats:

- IPv4 dotted decimal
- IPv6 hexadecimal notation
- Alphanumeric format

This flag is valid for TCP/IP only.

On systems that are both IPv4 and IPv6 capable you can split the traffic by running two separate listeners. One listening on all IPv4 addresses and one listening on all IPv6 addresses. If you omit this parameter, the listener listens on all configured IPv4 and IPv6 addresses.

**-n TpName**

The LU 6.2 transaction program name. This flag is valid only for the LU 6.2 transmission protocol. If you omit the name, it is taken from the queue manager configuration information.

**-a Adapter**

The adapter number on which NetBIOS listens. By default the listener uses adapter 0.

**-l LocalName**

The NetBIOS local name that the listener uses. The default is specified in the queue manager configuration information.

**-e Names**

The number of names that the listener can use. The default value is specified in the queue manager configuration information.

**-s Sessions**

The number of sessions that the listener can use. The default value is specified in the queue manager configuration information.

**-o Commands**

The number of commands that the listener can use. The default value is specified in the queue manager configuration information.

**-x Socket**

The SPX socket on which SPX listens. The default value is hexadecimal 5E86.

**-m QMgrName**

The name of the queue manager. By default the command operates on the default queue manager.

**-b Backlog**

The number of concurrent connection requests that the listener supports. See [TCP](#), [LU62](#), [NETBIOS](#), and [SPX](#) for a list of default values and further information.

**Return codes**

Table 96. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
4	Command completed after being ended by the <b>endmqlsr</b> command
10	Command completed with unexpected results
20	An error occurred during processing: the AMQMSRVN process did not start.

**Examples**

The following command runs a listener on the default queue manager using the NetBIOS protocol. The listener can use a maximum of five names, five commands, and five sessions. These resources must be within the limits set in the queue manager configuration information.

```
runmqlsr -t netbios -e 5 -s 5 -o 5
```

**Related reference**

[“Listener commands” on page 12](#)

A table of listener commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

### **runmqras (collect IBM MQ troubleshooting information)**

Use the **runmqras** command to gather IBM MQ troubleshooting information (MustGather data) together into a single archive, for example to submit to IBM Support.

**Purpose**

The **runmqras** command is used to gather troubleshooting information from a machine into a single archive. You can use this command to gather information about an application or IBM MQ failure, possibly for submitting to IBM when you report a problem.

The **runmqras** command requires a Java 7, or later, Java runtime environment (JRE) in order to run. If the IBM MQ JRE component (on Linux) or feature (on Windows) is not installed, then **runmqras** searches the system path for an alternative JRE and attempts to use that.

If no alternative could be found, error message AMQ8599 is output. In this case:

1. Install the IBM MQ JRE component, or install an alternative Java 7 JRE
2. Add the JRE to the system path
3. Rerun the command

By default, **runmqras** gathers information such as:

- IBM MQ FDC files
- Error logs (from all queue managers as well as the machine-wide IBM MQ error logs)
- Product versioning, status information, and output from various other operating system commands.

Note, for example, the **runmqras** command does not gather user information that is contained in messages on queues.


Running without requesting more sections is intended as a starting point for general problem diagnosis, however, you can request more *sections* through the command line.


These additional *sections* gather more detailed information, depending on the type of problem being diagnosed. If non-default sections are needed by IBM support personnel, they will tell you.

The **runmqras** command can be run under any user ID, but the command gathers only information that the user ID can gather manually. In general, when debugging IBM MQ problems, run the command under:

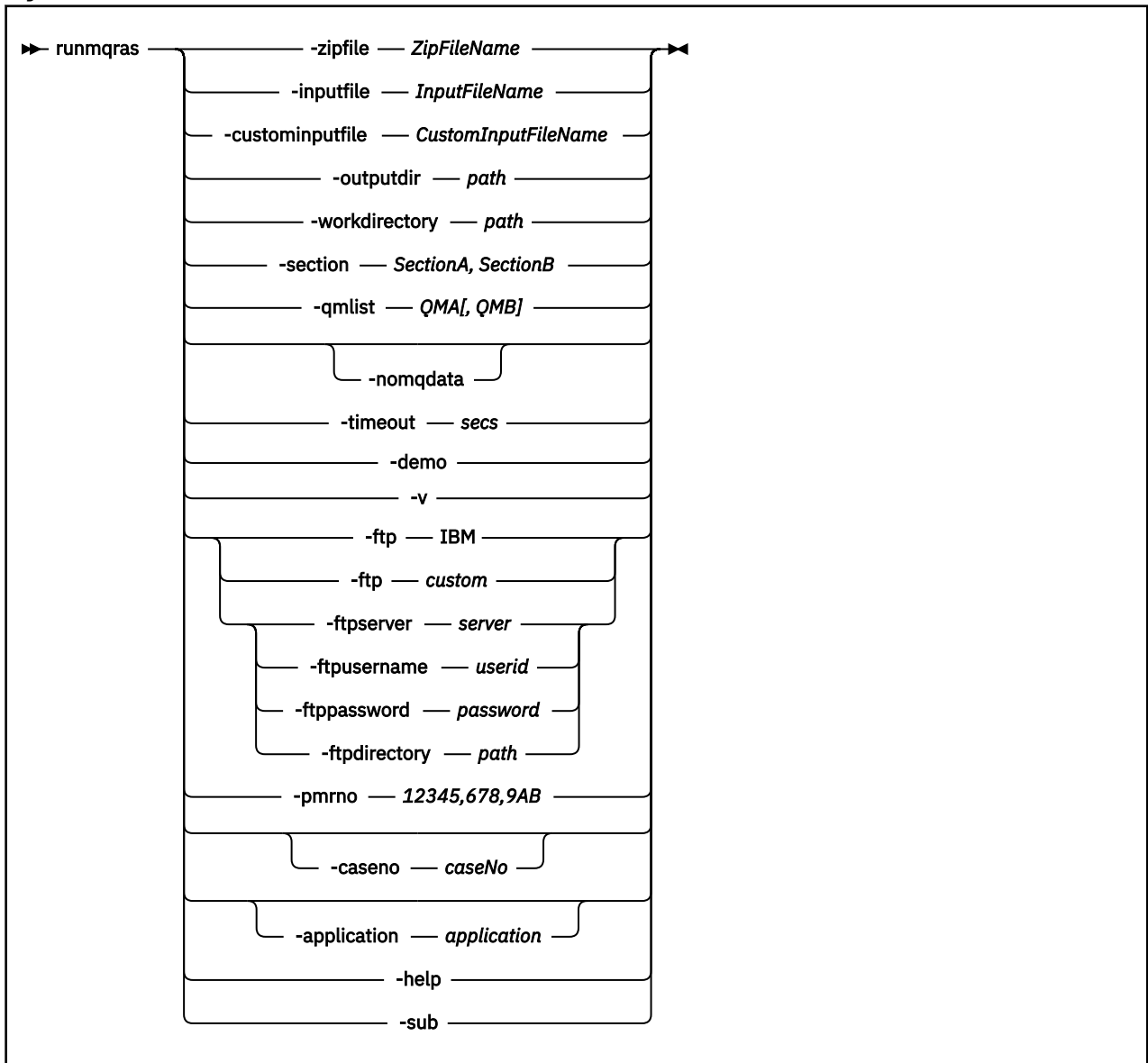
-  The mqm user ID
-  A user ID in the mqm group.

to allow the command to gather queue manager files and command outputs.

 The **runmqras** command, by default, retrieves the environment variable information. This applies to Linux and AIX.

 The **runmqras** command retrieves a listing of the queue manager's data directory by default. This applies to Multiplatforms. The userdata directory under the data directory is excluded.

## Syntax



### Keywords and parameters

All parameters are required unless the description states they are optional.

In every case, *QMgrName* is the name of the queue manager to which the command applies.

#### **-application *application***

Collects information about valid applications.

#### **-caseno *caseNo***

A valid Salesforce case number.

Use this option to ensure that the output is prefixed with your case number, so that when the information is sent to IBM, the information is automatically associated with that case number.

**Note:** If you want to specify a PMR number, use the **-pmrno** parameter, not the **-caseno** parameter.

It is not permitted to supply both the **-caseno** and **-pmrno** parameters together.

#### **-custominputfile *CustomInputFileName***

Fully qualified name of an additional XML input file

**-demo**

Run in demonstration mode where no commands are processed, and no files gathered.

By running in demonstration mode, you can see exactly which commands would have been processed, and what files would have been gathered. The output `.zip` file contains a `console.log` file that documents exactly what would have been processed and gathered, should the command be run normally.

**-ftp *ibm|custom***

Allows the collected archive to be sent through basic FTP to a remote destination.

At the end of processing, the resultant archive can be sent through basic FTP to a site of your choosing.

**Important:** From IBM MQ 9.3.0, the **-ftp IBM** option is no longer available. If you select this option, the following message is generated:

The FTP IBM option will no longer work as the IBM FTP servers have been disabled

**-ftpdirectory *path***

The directory on the FTP server to place the resulting `.zip` file into, used when an FTP custom option is used.

**-ftppassword *password***

The password to log in to the FTP server with, when an FTP custom option is used.

**-ftpserver *server***

An FTP server name to connect to, when an FTP custom option is used.

**-ftpusername *userid***

The user ID to log in to the FTP server with, when an FTP custom option is used.

**-help**

Provide simple help.

**-inputfile *InputFileName***

Fully qualified name of the XML input file

**-noqmdata**

From IBM MQ 9.3.0, setting **-noqmdata** captures installation-level diagnostics only, skipping any queue manager-specific diagnostics.

The **-qmlist** parameter and the **-noqmdata** parameter cannot be used together. If both parameters are specified, the following error is returned:

Argument error: At most one of -noqmdata or -qmlist may be supplied

**-outputdir *path***

The directory in which the resulting output file is placed.

By default, the output directory is the same as the work directory.

**-pmrno *12345,678,9AB***

A valid IBM PMR number (problem record number) against which to associate the documentation.

Use this option to ensure that the output is prefixed with your PMR number, so that when the information is sent to IBM, the information is automatically associated with that problem record.

**Note:** If you want to specify a Salesforce case number, use the **-caseno** parameter, not the **-pmrno** parameter.

It is not permitted to supply both the **-caseno** and **-pmrno** parameters together.

**-qmlist *QMA[, QMB]***

A list of queue manager names on which the **runmqras** command is to be run.

This parameter does not apply to a client product because there are no queue managers from which to request direct output.

By supplying a comma-separated list, you can restrict the iteration across queue managers to a specific list of queue managers. By default, iteration of commands is across all queue managers.

## **-section SectionA,SectionB**

The optional sections about which to gather more specific information. You must use a comma as the separator character between sections, with no spaces. For example:

```
runmqras -qmlist ESBSTGAPPQMVH2 -section defs,trace,cluster -caseno TEST123
```

By default, a generic section of documentation is collected, whereas more specific information can be gathered for a specified problem type; for example, a section name of *trace* gathers all of the contents of the trace directory.

The default collections can be avoided by supplying a section name of *nodefault*.

IBM support generally supplies you with the sections to use. Example available sections are:

### **all**

Gathers all possible information, including all trace files, and diagnostics for many different types of problems. You must use this option only in certain circumstances and this option is not intended for general use.

### **cluster**

Gathers cluster configuration and queue information.


### **dap**

Gathers transaction and persistence information.

### **default**

IBM MQ logs, FDC files, basic configuration, and status.

**Note:** Always gathered unless you use the section name **nodefault**. Some information about the current environment (saved in `env.stdout` on Linux, AIX and IBM i, and in `set.stdout` on Windows) and current user limits (saved in `mqconfig.stdout` on AIX and Linux) might be altered by the **runmqras** command. If necessary, run the **env**, **set**, or **mqconfig** commands manually in your environment to check the actual values.

 On the IBM MQ Appliance, any files other than queue manager trace files present in the `mqtrace: filesystem` are now captured in the *default* section.

**Note:** You should continue to specify the *trace* section if you need to obtain any queue manager trace files present in the `mqtrace: filesystem`.

### **defs**

Gathers the queue manager definitions and status information.

### **kernel**

Gathers queue manager kernel information.

### **leak**

Gathers IBM MQ process resource usage information.

This section applies to Linux, and AIX.

### **logger**

Gathers recovery logging information.

### **mft**

Captures the data obtained by the **fteRas** command.

**Note:** **-section mft** only collects information for the default coordination queue manager topology.

### **mqweb**

Gathers trace and configuration data for the mqweb server.

### **nativeha**

Gathers diagnostic information from a native HA queue manager instance. It is useful to gather information from all instances to view recent history of activity and communication between the instances.

**nodefault**

Prevents the default collections from occurring, but other explicitly requested sections are still collected.

**QMGR**

Gathers all queue manager files: queues, logs, and configuration files.

**topic**

Gathers topic tree information.

**trace**

Gathers all the trace file information plus the default information.

**Note:** Does not enable tracing.

For more information, see [Choosing sections to gather](#), in the IBM technote on using the IBM MQ **runmqras** command to collect data.

**-sub**

Shows the keywords that will be substituted in the xml.

**-timeout secs**

The default timeout to give an individual command before the command stops waiting for completion. By default, a timeout of 10 seconds is used. A value of zero means wait indefinitely.




**-v**

Extends the amount of information that is logged in the console .log file, contained in the output .zip file.

**-workdirectory path**

The directory that is used for storing the output from commands that are run during the processing of the tool. If supplied, this directory must either not exist, in which case it is created, or must be empty.

If you do not supply the path, a directory whose name starts with **runmqras** and is suffixed by the date and time is used:

-   On AIX and Linux, the directory is under /tmp.
-  On Windows, the directory is under %temp%.

**-zipfile ZipFileName**

Supply the file name of the resulting archive.

**runmqras** appends the hostname to the name of the archive file. For example, if you run the following command:

```
runmqras -zipFile diagnostics.zip
```

the resulting archive file is called `diagnostics-hostname.zip`.

By default, the name of the archive file is `runmqras-hostname.zip` where *hostname* is the hostname that **runmqras** appends to the file name.

**Examples**

This command gathers the default documentation from the IBM MQ installation, and all queue managers on a machine:

```
runmqras
```

This command gathers the default documentation from the IBM MQ installation on a machine into an output file with a name that starts with the appropriate case number:

```
runmqras -caseno TS123456789
```



This command gathers the default documentation from a machine, plus all trace files, the queue manager definitions, and status for all queue managers on the machine:

```
runmqras -section trace,defs
```

For more examples of how to use **runmqras**, see [Collecting troubleshooting information automatically with runmqras](#).

## Return codes

A non zero return code indicates failure.

## Related tasks

[Collecting troubleshooting information automatically with runmqras](#)

## Related information

[Sending troubleshooting information to IBM](#)

## Multi **runmqsc (run MQSC commands)**

Reference information about the **runmqsc** command prompt, which you can use to issue MQSC commands to a queue manager.

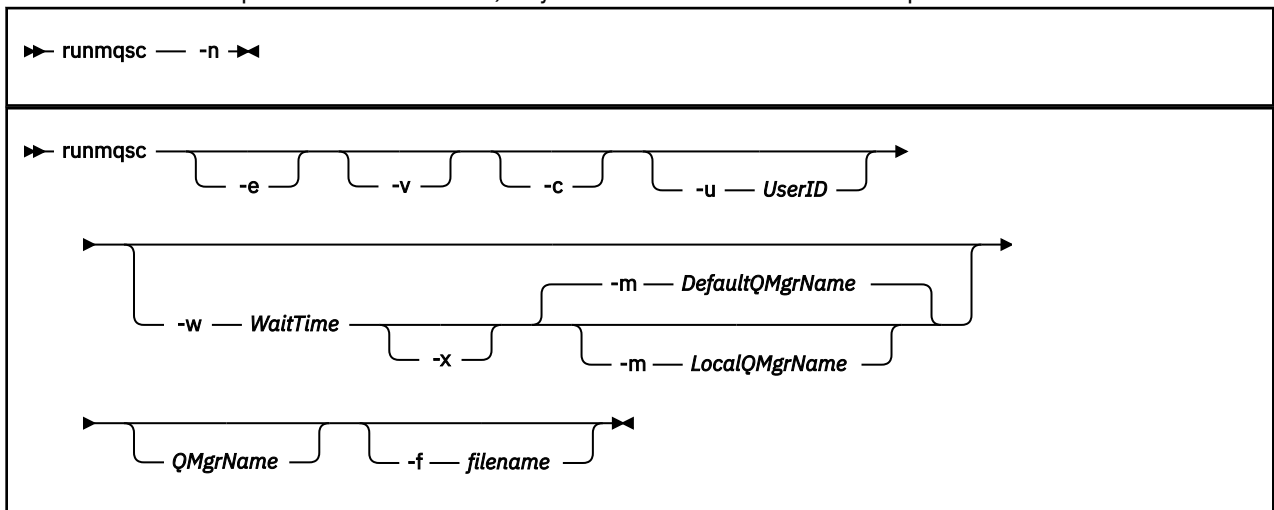
## Purpose

On AIX, Linux, and Windows, the **runmqsc** command is used to open a command prompt from which you can issue MQSC commands. MQSC commands enable you to perform administration tasks. For example, you can define, alter, or delete a local queue object.

How to open, use and close the **runmqsc** command prompt is described in [Running MQSC commands interactively under runmqsc](#). The MQSC commands and their syntax are described in [“MQSC commands reference”](#) on page 251.

## Syntax

You can use the **-n** parameter on its own, or you can use a number of other parameters in combination:



## Description

You can start the **runmqsc** command in three ways:

### Verify command

Verify MQSC commands but do not run them. An output report is generated indicating the success or failure of each command. This mode is available on a local queue manager only.

### Run command directly

Send MQSC commands directly to a local queue manager.

### Run command indirectly

Run MQSC commands on a remote queue manager. These commands are put on the command queue of a remote queue manager and run in the order in which they were queued. Reports from the commands are returned to the local queue manager.

The **runmqsc** command takes its input from `stdin`. When the commands are processed, the results and a summary are put into a report that is sent to `stdout`.

By taking `stdin` from the keyboard, you can enter MQSC commands interactively.

Alternatively, you can redirect `stdin` from a text file. By redirecting the input from a file, you can run a sequence of frequently used commands contained in the file. You can also redirect the output report to a file.

**Note:** If you run **runmqsc** in client mode by redirecting `stdin` from a text file, IBM MQ expects the first line of the input file to be a password.

## Optional parameters

### -c

Modifies the **runmqsc** command to connect to a queue manager by using a client connection. The client channel definitions used to connect to the queue manager are located using the following environment variables in this order of precedence: `MQSERVER`, `MQCHLLIB`, and `MQCHLTAB`.

This option requires the client to be installed. If it is not installed an error message reporting the missing client libraries is issued.

### -e

Prevents source text for the MQSC commands from being copied into a report. This parameter is useful when you enter commands interactively.

### -m *LocalQMgrName*

The local queue manager that you want to use to submit commands to the remote queue manager. If you omit this parameter the local default queue manager is used to submit commands to the remote queue manager. The **-w** parameter must also be specified.

### -n

Modifies the **runmqsc** command to not connect to a queue manager. If this parameter is specified, all other command parameters must be omitted, otherwise an error message is issued.

This option requires the client libraries to be installed. If they are not installed an error message is issued.

MQSC commands entered in this mode are limited to managing the local channel definition file, which is located through the `MQCHLLIB` and `MQCHLTAB` environment variables, or the default values if not defined.

**Note:** If you add new entries into the local channel definition file, or alter existing entries, these changes are not reflected inside the queue manager. The queue manager does not read the contents of the local channel definition file. The CCDT file is a write-only file from the perspective of the queue manager. The queue manager does not read the contents of the CCDT file.

Only the following MQSC commands are recognized:

**ALTER, DEFINE, DELETE, DISPLAY AUTHINFO** (Only of type CRLLDAP or OCSP)  
**ALTER, DEFINE, DELETE, DISPLAY CHANNEL** (Only of type CLNTCONN)

For the AUTHINFO management commands, the names of existing AUTHINFO definitions are mapped and addressed using the names CRLLDAP $n$  or OCSP  $n$  (according to type), where  $n$  is the numeric

order in which they appear in the channel definition file. New AUTHINFO definitions are appended to the client channel table in order. For example, the the following commands are issued:

```
DEFINE AUTHINFO(XYZ) AUTHTYPE(CRLLDAP) CONNAME('xyz')
DEFINE AUTHINFO(ABC) AUTHTYPE(CRLLDAP) CONNAME('abc')
```

This results in the 'xyz' LDAP server being checked for a CRL first, if that CRL server is unavailable then the 'abc' server is checked.

Using the **DISPLAY AUTHINFO(\*) CONNAME** command shows this:

```
AMQ8566: Display authentication information details.
AUTHINFO(CRLLDAP1)
AUTHTYPE(CRLLDAP)          CONNAME(xyz)
AMQ8566: Display authentication information details.
AUTHINFO(CRLLDAP2)
AUTHTYPE(CRLLDAP)          CONNAME(abc)
```

**Note:** The client mode only supports inserting new entries at the end of the client channel table. If you want to change the order of precedence of the CRL LDAP servers, you must remove the existing objects from the list and reinsert them in the correct order at the end.

#### **-u UserID**

If you use the **-u** parameter to supply a user ID, you are prompted for a matching password.

If you have configured the CONNAUTH AUTHINFO record with CHCKLOCL (REQUIRED) or CHCKLOCL (REQDADM), you must use the **-u** parameter otherwise you will not be able to administer your queue manager with **runmqsc**.

If you specify this parameter and redirect stdin, a prompt will not be displayed and the first line of redirected input should contain the password.

#### **-v**

Verifies the specified commands without performing the actions. This mode is only available locally. The **-w** and **-x** parameters are ignored if they are specified at the same time as **-v**.

**Important:** The **-v** flag checks the syntax of the command only. Setting the flag does not check if any objects mentioned in the command actually exist.

For example, if the queue Q1 does not exist in the queue manager, the following command is syntactically correct and does not generate any syntax errors: `runmqsc -v Qmgr display q1(Q1)`.


However, if you omit the **-v** flag, you receive error message AMQ8147.

#### **-w WaitTime**

Run the MQSC commands on another queue manager. You must have the required channel and transmission queues set up for this. For more information, see [Configuring queue managers for remote administration](#).

This parameter is ignored if the **-v** parameter is specified.

#### **WaitTime**

The time, in seconds, that **runmqsc** waits for replies. Any replies received after this are discarded, but the MQSC commands still run.  The wait time is set as the expiry time of the PCF command message, and the remaining time is set on the PCF reply messages by the command server. Specify a time in the range 1 through 999999.

Each command is sent as an Escape PCF to the command queue (SYSTEM.ADMIN.COMMAND.QUEUE) of the target queue manager.

The replies are received on queue SYSTEM.MQSC.REPLY.QUEUE and the outcome is added to the report. This can be defined as either a local queue or a model queue.

**-x**

The target queue manager is running under z/OS. This parameter applies only in indirect mode. The **-w** parameter must also be specified. In indirect mode, the MQSC commands are written in a form suitable for the IBM MQ for z/OS command queue.

**QMgrName**

The name of the target queue manager on which to run the MQSC commands. If not specified, the default queue manager is used.

**-f filename**

Read input to be processed from the supplied filename rather than standard input.

**Return codes**

Table 97. Return code identifiers and descriptions

Return code	Description
00	MQSC command file processed successfully
10	MQSC command file processed with errors; report contains reasons for failing commands
20	Error; MQSC command file not run

**Related tasks**

Running MQSC commands interactively under [runmqsc](#)

Running MQSC commands from text files


**runmqsfb (run IBM MQ Bridge to Salesforce)**

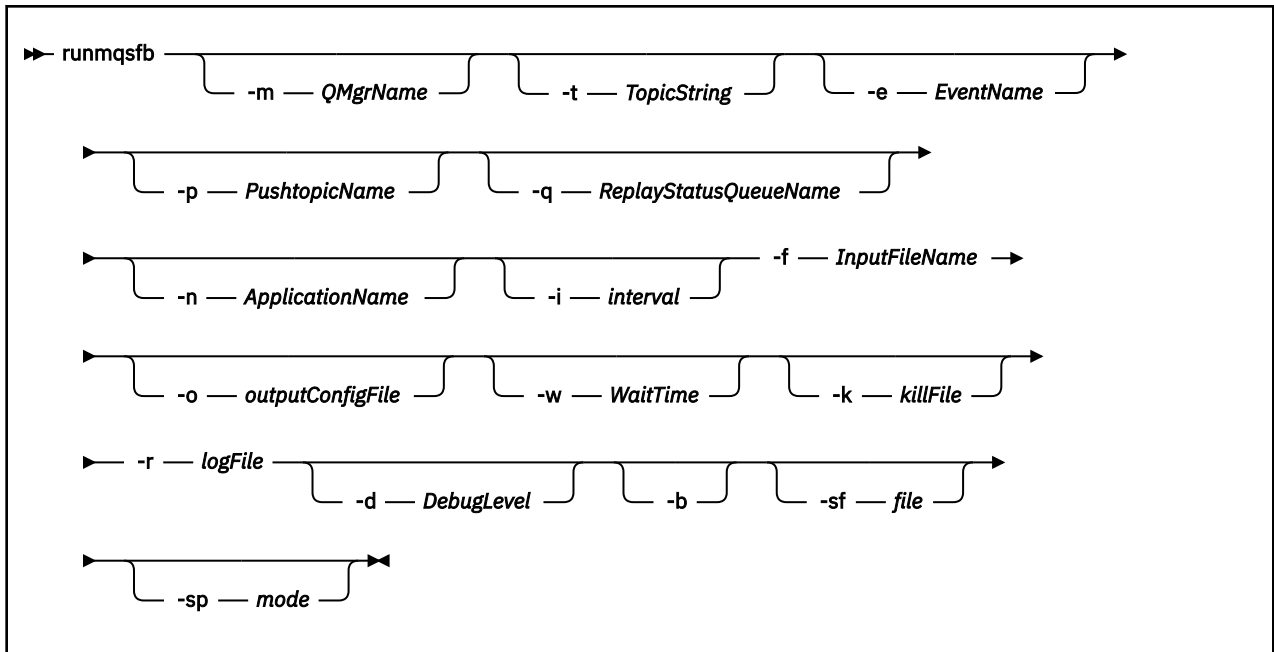
Configure and run the IBM MQ Bridge to Salesforce.

**Note:** The IBM MQ Bridge to Salesforce is deprecated across all releases from November 22 2022 (see [US Announcement letter 222-341](#)).

- [“Syntax” on page 180](#)
- [“Usage notes” on page 181](#)
- [“Command line parameters” on page 182](#)
- [Configuration parameters](#)
- [Examples](#)

**Syntax**

The diagram shows the syntax for the **runmqsfb** command usage as described in note [“1” on page 181](#).



## Usage notes

1. You can run the **runmqsf** command to start the IBM MQ Bridge to Salesforce and connect to Salesforce and IBM MQ. When the connections are made, the bridge receives Salesforce generated events and publishes them to an IBM MQ network, or create event messages for Salesforce platform events.

```
runmqsf -f configFile -r logFile -m QMgrName -t TopicString -e EventName -p PushtopicName
-d debugLevel -i interval -w WaitTime -k killFile
```

When you use the command for runtime processing, the required parameters are **-f**, with the name of the previously created configuration file, and **-r** with the name of the log file. When the other command parameters are also given on the command line, they override the values in the configuration file. The option allows a core default configuration to be created and provides a simple way to handle minor variations such as the queue manager name.

2. You can also use the **runmqsf** command to generate a configuration file that is used to define the parameters that are needed to connect to Salesforce and IBM MQ.

When you are creating the configuration file, the **-f** and **-b** parameters are optional, the input configuration file is included in the samples directory for the IBM MQ Bridge to Salesforce, `/opt/mqm/mqsF/samp`.

```
runmqsf [-b][-f inputConfigFile] -o outputConfigFile
```

When you run the command in this way, you are prompted to enter values for each of the configuration parameters. To keep an existing value press `Enter`. To remove an existing value press `Space`, then `Enter`. For more information, see [“Configuration parameters”](#) on page 183.

3. From IBM MQ 9.2.0, the usage of the enhanced protection parameters is as follows:

### Configuration mode

Newly entered passwords are written to the output configuration file protected with the new key. Note that preexisting passwords are not changed in format.

A warning is issued when the default key is being used, that is, you have not provided a keyfile.

Provision of passwords in the batch configuration mode, using environment variables, continues unchanged; that is, the value of the environment variables is given in plain text.

## Runtime mode

When a password is decrypted, warning messages are issued if the password is found to be in the old format, and the name of the parameter causing the warning is given in the message to encourage you to migrate. However, the bridge continues processing commands.

**Note:** The warning message is not issued if you have specified the **-sp 0** parameter on the command line, as you explicitly wanted to use old formats.

A warning is also issued when the default key is being used, that is, you have not provided a keyfile.

Errors occur if a password cannot be decrypted, for example, if you have specified the wrong keyfile.

## Command line parameters

### **-m QMgrName or ConnFactoryName**

Queue manager or Connection Factory name.

### **-r logFile**

Required. Location and name of the log file for trace information. You can specify the log file path and name in the configuration file or on the command line.

### **-t TopicString**

IBM MQ topic root.

### **-e EventName**

Salesforce platform event name (can repeat). You can specify multiple **-e** entries on the command line, one for each event type that the bridge listens for. You must provide the base part of the event name. The bridge automatically adds `/event` or `/topic` prefixes when it connects to Salesforce. Multiple **-e** parameters can be comma-separated.

### **-p PushtopicName**

Salesforce push topic name (can repeat). You can specify multiple **-p** entries on the command line, one for each topic type that the bridge listens for. You must provide the base part of the topic name. The bridge automatically adds `/event` or `/topic` prefixes when it connects to Salesforce. Multiple **-p** parameters can be comma-separated.

### **-i interval**

Monitor interval. Enter 0 to disable monitoring.

### **-f inputConfigFile**

Configuration file. The **-f** parameter is required when you are running the **runmqsf** command to start the IBM MQ Bridge to Salesforce, as described in usage note [“1” on page 181](#). You can optionally use the **-f** parameter to reuse some of the values from an existing *inputConfigFile*, as described in usage note [“2” on page 181](#), and also enter some of the new values. If you do not specify the **-f** parameter when you are creating the configuration file, all the values for the parameters you are prompted for are empty.

### **-n ApplicationInstanceName**

If you have multiple instances of the bridge on the same queue manager, this option gives you a way to distinguish each instance in the monitoring. This identifier is added to the \$SYS topic as part of the application name, so that monitors like **amqsr** can get separate metadata trees.

If this option is not empty, the metadata root topic adds `_` together with the selected value to the application name. For example, setting it to `2` results in publications based on

```
$SYS/Application/runmqsfb_2/INFO/QMGR/<qmgr>/Monitor/METADATA
```

**Note:** There is no coordination between running bridge instances, so it is possible to have two instances with the same identifier. The only confusion this causes is in the monitoring statistics.

### **-q ReplyStatusQueueName**

The default value is `SYSTEM.SALESFORCE.SYNCQ`.

If you want to have multiple bridges accessing the same queue manager, and having inbound messages from Salesforce, then you must have separate synchronization queue settings for each.

If a bridge instance is not subscribed to any Salesforce topics, the synchronization queue is not used at all.

**Note:** The synchronization queue is accessed exclusively; that is, the bridge will not start if another instance already has the nominated queue open.

**-o outputConfigFile**

New configuration file. When you run the command with the **-o** parameter, **runmqsfb** command loads existing configuration values from the **-f** file and prompts for new values for each configuration parameter.

**-k killFile**

A file to cause the bridge to exit. When you run the command with the **-k** parameter and specify a file, if the file exists, it causes the bridge program to exit. Using this file is an alternative way to stop the program when you don't want to use **Ctrl+C** or **kill** command. The file is deleted by the bridge on startup in case it exists. If the deletion fails, the bridge abends but monitors for the recreation of the file.

**-d debugLevel**

Debug level, 1, or 2.

**1**

Terse debug information is displayed.

**2**

Verbose debug information is displayed.

**-w WaitTime**

Wait before fully starting.

**-b**

Use environment variables to drive the configuration, instead of interactive prompts. This allows the configuration to be set programatically.

The environment variables have the format "runmqsfb\_<attribute>" where <attribute> is the JSON field in the generated configuration file. For example:

```
export runmqsfb_QueueManager=QM1
```

The environment variables are merged with the configuration specified in the *inputConfigurationFile* (**-f** option) to create the *outputConfigurationFile*.

One way of using this method, is for you to interactively create a configuration file that contains common attributes that are to be used by all instances of the IBM MQ Bridge to Salesforce, and then apply environment variables programatically for just the few instance-unique parameters.

**-sf file**

File containing password protection key.

**-sp mode**

Password protection mode. The values can be:

**2**

Use the latest password protection mode. This is the default value from IBM MQ 9.3.0.

**1**

Use the IBM MQ 9.2 password protection mode for compatibility with versions earlier than IBM MQ 9.3.0. This is the default value in versions earlier than IBM MQ 9.3.0.

**0**

Use the deprecated password protection mode that is compatible with versions earlier than IBM MQ 9.2.

## Configuration parameters

When you run the **runmqsfb** command to create the configuration file, the parameters are stepped through in four groups. Passwords are obfuscated and are not displayed as you type. The generated

configuration file is in JSON format. You must use the **runmqsfb** command to create the configuration file. You cannot edit the passwords and security certificate information directly in the JSON file.

### Connection to queue manager

Parameters relating to the IBM MQ queue manager.

#### IBM MQ Queue manager or JNDI CF

Required.

#### IBM MQ Base Topic

Required. All events are published by using the topic root as the prefix to the Salesforce event name.

#### IBM MQ Channel

Blank **channel** implies local bindings.

#### IBM MQ Conname

Uses standard connection name format of "host(port), host(port)" to enable multiple destinations such as for multi-instance queue managers. Blank **conname** implies local bindings.

#### IBM MQ Publication Error Queue

Required for creating platform event messages. IBM MQ error queue for processing bad input messages. The default queue *SYSTEM.SALESFORCE.ERRORQ* is created when you run the **mqsfbSyncQ.mqsc** script command that also creates the required synchronization queue on the queue manager.

#### IBM MQ CCDT URL

If a TLS connection is required to the queue manager, you must use a JNDI or CCDT definition.

#### JNDI implementation class name

The class name of your JNDI provider. The "queue manager name" parameter refers to the connection factory name when you are using JNDI.

#### JNDI provider URL

The endpoint of your JNDI service.

#### IBM MQ UserId

#### IBM MQ Password

### Connection to Salesforce

Parameters relating to Salesforce.

#### Salesforce Userid (required)

Required. Log in email for your Salesforce account.

#### Salesforce password (required)

Required. Password for your Salesforce account.

#### Salesforce security token (required)

Required. Security token that you can generate from the **Security controls** section of the **Administer** menu of your Salesforce **Force.com Home** page.

#### Login Endpoint

Salesforce login endpoint URL, <https://login.salesforce.com>.

#### Consumer key

The consumer key that you generate when you add the IBM MQ Bridge to Salesforce as a connected application in your Salesforce account. For more information, see [Step 5 in Configuring the IBM MQ Bridge to Salesforce](#)

#### Consumer secret

The consumer secret that is generated along with the consumer key.

OAuth consumer key and secret values are optional but must be considered for production systems.

### Certificate stores for TLS connections

Parameters relating to certificate stores for TLS connections.



**Personal keystore for TLS certificates**

Required. The keystore that you create in your Salesforce account. For more information, see [Step 3 in Configuring the IBM MQ Bridge to Salesforce](#).

**Keystore password**

Required. The password that you create when you are exporting the keystore from your Salesforce account.

**Trusted store for signer certificates**

Required. If you do not add the trusted store, the personal keystore for TLS certificates is used.

**Trusted store password**

Required. If the personal keystore for TLS certificates is used, this is the password for the keystore for TLS certificates.

**Use TLS for MQ connection**

If you are using TLS for your IBM MQ connectivity, you can use the same keystore that you used to connect to Salesforce.

For the Salesforce connection, a truststore must be available and contain at least the signer certificates to validate the Salesforce system. Only TLS 1.1 and TLS 1.2 protocols are supported for connection to Salesforce. A user certificate is not required. If you don't provide a truststore parameter, the keystore is used for both roles. The stores can be the same as the one configured for the IBM MQ connection in the CCDT or JNDI.

**Behavior of bridge program**

Parameters relating to the behavior of the IBM MQ Bridge to Salesforce.

**Push topic names**

You can provide one push topic name at a time and then move on to the next parameter by pressing `enter`.

**Platform event names**

You can provide one platform event name at a time and then move on to the next parameter by pressing `enter`.

**Monitoring frequency**

IBM MQ Monitoring frequency.

**At least once delivery**

Quality of service. At-least-once or at-most-once delivery.

**Subscribe to IBM MQ publications for platform events**

Required. The default option is `N`. You must enter `Y` to enable the bridge feature for creating event messages for Salesforce platform events.

**Publish control data with the payload**

On republish, send full message not only subject.

**Delay before starting to process events**

Delay before the bridge starts to process events.

**Runtime logfile for copy of stdout/stderr**

Path to and name of the log file for the tracing information.

**Bridge unique identifier**

The default value is no Bridge unique identifier specified.

If you have multiple instances of the bridge on the same queue manager, this option gives you a way to distinguish each instance in the monitoring. This identifier is added to the `$SYS` topic as part of the application name, so that monitors like `amqsrua` can get separate metadata trees.

If this option is not empty, the metadata root topic adds `"_"` together with the selected value to the application name. For example, setting it to `"2"` results in publications based on

```
$SYS/Application/runmqsfb_2/INFO/QMGR/<qmgr>/Monitor/METADATA
```

**Note:** There is no coordination between running bridge instances, so it is possible to have two instances with the same identifier. The only confusion this causes is in the monitoring statistics.

### **Treat unknown Salesforce topic as warning**

The default option is *N*.

You must set this option to *Y* to continue with a warning, instead of exiting if a push topic or an event is not known to Salesforce during the subscribe.

This option can be useful when creation of the topics in Salesforce is done independently, and might not be available immediately. This allows the bridge to run for the topics that are known.

The bridge still needs restarting, or forced through a reconnection sequence, for example, restarting the queue manager, before it tries to subscribe to the topic again.

### **Continue to retry after maximum reconnection attempts**

The default option is *N*.

You must set this option to *Y* to not exit after the final retry timer. Instead, keep retrying at the last interval forever.

### **At-least-once delivery for IBM MQ publications?**

The value can be *Y* or *N*, and the default value is not set. The value is inherited from another quality of service attribute.

This attribute determines whether a durable or non-durable subscription is made for IBM MQ publications. It separates the existing quality of service, that is:

- At-most-once for a non durable subscription, or
- At-least-once for a durable subscription

into different attributes for each direction of flow.

This is useful for situations where you want to have at-most-once inbound messages from Salesforce, not bothering with the **ReplayId**, but still want to send saved outbound messages from IBM MQ (at-least-once) after an outage.

If not set, the existing quality of service value is used. Note that this happens if you have migrated from an older version without updating the configuration file, that is, run the configuration process.

### **MQ Replay Status Queue**

The default value is `SYSTEM.SALESFORCE.SYNCQ`.

If you want to have multiple bridges accessing the same queue manager, and having inbound messages from Salesforce, then you must have separate synchronization queue settings for each.

If a bridge instance is not subscribed to any Salesforce topics, the synchronization queue is not used at all.

**Note:** The synchronization queue is accessed exclusively; that is, the bridge will not start if another instance already has the nominated queue open.

### **Number of logfiles**

The default value is 3.

Allow rotating log files for the output recording. If the value is greater than one, the configured log file name is used as a base with ".0", ".1", and so on, appended or inserted before the filetype.

If you use the default value, do not add an index.

Note that normal **stdout** and **stderr** handling is unaffected.

### **Maximum size of each logfile**

The default value is 2097152 bytes (2 MB) .

If you configure more than one log file, this is when the log switch is made.

If you configure only one log file, this parameter is ignored.

**Push topic names** and **Platform event names** can be entered individually or as a comma-separated list, in the same way as the command line **-p** and **-e** parameters are entered. The **Startup wait interval** provides an option to delay the initial processing of events. For example, if the bridge and the IBM MQ applications that use it are all run as services, the order in which they start cannot be sequenced. Therefore, events might be republished before applications are prepared to receive them.

When you delay the bridge startup, you give the applications the time to start and subscribe to events and push topics.

The configuration is only read on startup of the bridge process. Changes to the configuration require a restart, such as through the IBM MQ Service definitions.

## Examples

The **-f** parameter is optional when you use the **runmqsf** to create the configuration file as described in the usage note “2” on page 181.

```
runmqsf -f inputConfigFile -o outputConfigFile
```

In this example, the *outputConfigFile* is created:

```
runmqsf -o outputConfigFile
```

The **-f** parameter is required when you use the **runmqsf** command to run the IBM MQ Bridge to Salesforce, as described in the usage note “1” on page 181.

```
runmqsf -f inputConfigFile -r logFile
```

## Related tasks

[Configuring IBM MQ for use with Salesforce push topics and platform events](#)

[Tracing the IBM MQ Bridge to Salesforce](#)

[Monitoring the IBM MQ Bridge to Salesforce](#)

## **Multi** runmqtm (start client trigger monitor)

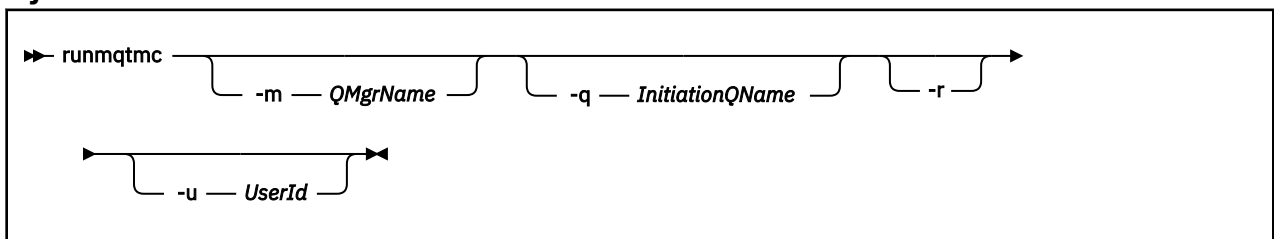
Start the trigger monitor on a client.

## Purpose

Use the **runmqtm** command to start a trigger monitor for a client. For further information about using trigger monitors, see [Trigger monitors](#).

When a trigger monitor starts, it continuously monitors the specified initiation queue. The trigger monitor does not stop until the queue manager ends, see “[endmqm \(end queue manager\)](#)” on page 121. While the client trigger monitor is running it keeps the dead letter queue open.

## Syntax



## Optional parameters

### **-m QMgrName**

The name of the queue manager on which the client trigger monitor operates, by default the default queue manager.

### **-q InitiationQName**

The name of the initiation queue to be processed, by default SYSTEM.DEFAULT.INITIATION.QUEUE.

### **-r**

Specifies that the client trigger monitor automatically reconnects.

### **-u *UserId***

The ID of the user authorized to get the triggered message.

Note that using this option does not affect the authority of the triggered program, that might have its own authentication options.

**Note:** As the `runmqtrmc` command makes a standard client connection, you can send a user ID and password, and have the password encrypted, using the `mqccred` security exit.

## **Return codes**

For IBM MQ 9.1.3 and earlier, and for LTS before IBM MQ 9.2.0, the value of 0 is not used and the value is reserved. The trigger monitor is designed to run continuously and therefore not to end.

Table 98. Return code identifiers and descriptions

<b>Return code</b>	<b>Description</b>
0	From IBM MQ 9.1.4, client trigger monitor interrupted because the queue manager is ending, or the channel is stopped.
10	Trigger monitor interrupted by an error.
20	Error; client trigger monitor not run.

## **Examples**

For examples of using this command, see [The Triggering sample programs](#).

### **Multi `runmqtrm (start trigger monitor)`**

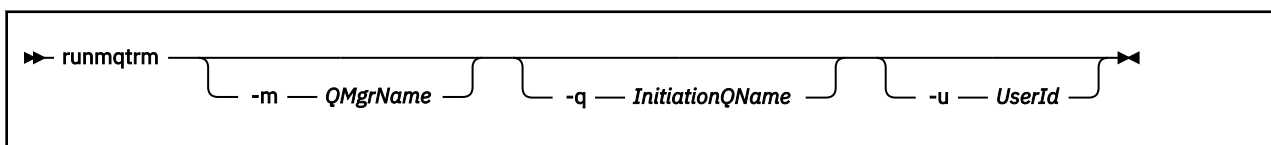
Start the trigger monitor on a server.

## **Purpose**

Use the `runmqtrm` command to start a trigger monitor. For further information about using trigger monitors, see [Trigger monitors](#).

When a trigger monitor starts, it continuously monitors the specified initiation queue. The trigger monitor does not stop until the queue manager ends, see [“endmqm \(end queue manager\)”](#) on page 121. While the trigger monitor is running it keeps the dead letter queue open.

## **Syntax**



## **Optional parameters**

### **-m *QMGrName***

The name of the queue manager on which the trigger monitor operates, by default the default queue manager.

### **-q *InitiationQName***

Specifies the name of the initiation queue to be processed, by default SYSTEM.DEFAULT.INITIATION.QUEUE.

### **-u *UserId***

The ID of the user authorized to read the initiation queue, and get the triggered message.

Note that using this option does not affect the authority of the triggered program, that might have its own authentication options.

## Return codes

For IBM MQ 9.1.3 and earlier, and for LTS before IBM MQ 9.2.0, the value of 0 is not used and the value is reserved. The trigger monitor is designed to run continuously and therefore not to end.

Table 99. Return code identifiers and descriptions

Return code	Description
0	From IBM MQ 9.2.0, trigger monitor interrupted because the queue manager is ending.
10	Trigger monitor interrupted by an error.
20	Error; trigger monitor not run.

## Multi V 9.3.0 **runp11cred (protect PKCS #11 cryptographic hardware passwords)**

The **runp11cred** command protects passwords that are contained in PKCS #11 cryptographic hardware configuration strings. The cryptographic hardware configuration strings are supplied to IBM MQ clients by using either the **MQSSLCRYP** environment variable, or the **SSLCryptoHardware** attribute in the **SSL** stanza of the client configuration file.

### Purpose

The **runp11cred** command prompts for the password to be encrypted to be entered. The initial key that is used to encrypt the password can be specified in a file. The path to the file that contains the initial key is specified by using one of the following options, in order of priority:

1. The **-sf** parameter to the **runp11cred** command.
2. The **MQS\_SSLCRYP\_KEYFILE** environment variable.

If the initial key file is not specified by using either of these options, a default initial key is used to encrypt the password.



**Attention:** Do not use the default initial key as it does not protect passwords securely.

After the password is encrypted, **runp11cred** displays the encrypted password string.

Store the encrypted password in the cryptographic hardware configuration string, which is specified either in the appropriate property in the `mqclient.ini` file, or the **MQSSLCRYP** environment variable.

### Syntax

```
runp11cred -sf keyfile -sp protection_mode
```

### Optional Parameters

#### **-sf keyfile**

The path to the file that contains the initial key that is used to encrypt the password. If specified, the file must contain at least one character, and only one line.

If this parameter is not specified, a default initial key is used.

### **-sp protection\_mode**

The password protection mode to be used by the command. One of the following values can be specified:

**1**

Use the IBM MQ 9.2.0 password protection algorithm.

**2**

Use the latest password protection mode. This mode is the most secure credentials protection method.

This value is the default.

### **Examples**

```
>runp11cred
```

```
5724-H72 (C) Copyright IBM Corp. 1994, 2024.
Enter password:
*****
Credentials are encrypted using the default encryption key. For more secure
protection of stored credentials, use a custom, strong encryption key.
<P11>!2!N5eSuyDco5urE1GXhvpX7Hdk4bo840A08b0ZqyZv9P8=!Wt1g2x2S1YmCvhFtkUM5Ag==
```

```
>runp11cred -sf InitialKey.file
```

```
5724-H72 (C) Copyright IBM Corp. 1994, 2024.
Enter password:
*****
<P11>!2!8ctSQHBKH0m7cBhbqz11Fx0iVGrlka9340DvIR/Dx7g=!Ssv1sLVVZrt/30DvwcoK1w==
```

### **Return codes**

**0**

Command completed successfully.

**1**

Command completed unsuccessfully.

## **Linux V9.3.4 AIX runqmc cred (protect authentication token keystore password)**

Use the **runqmc cred** command to encrypt the password for the queue manager keystore that contains the trusted authentication token issuer's public key certificates or symmetric keys.

### **Purpose**

The **runqmc cred** command is used to encrypt the queue manager authentication token keystore. The authentication token keystore contains the public key certificates or symmetric keys for trusted authentication token issuers. The path to the keystore and the file that contains the encrypted password are specified in the **AuthToken** stanza in the `qm.ini` file. The queue manager uses the information in the **AuthToken** stanza to verify that the token that an application provides for authentication purposes is issued by a trusted issuer.

The command is provided because the use and storing of unencrypted passwords is not secure.

An encryption key, which is known as the initial key, is used to encrypt the password. You can provide a file that contains the initial key when you run the **runqmc cred** command. Create the initial key file before you run the command. If you do not provide the initial key, a default initial key is used.



**CAUTION:** Do not use the default initial key to encrypt passwords as it does not protect passwords securely.

**Important:** If you supply an initial key file that contains the encryption key, the same initial key must be specified in the queue manager **INITKEY** attribute so that the queue manager can decrypt the password.

If the queue manager **INITKEY** attribute is already set, use the same initial key when you run the **runqmcrcd** command. For more information about the queue manager **INITKEY** attribute, see “[UNIX, Linux, Windows, IBM i][MQ 9.3.0 Jun 2022][MQ 9.3.0 Jun 2022]INITKEY” on page 363.

## Syntax

```
runqmcrcd -sf keyfile
```

## Optional Parameters

### **-sf keyfile**

Path to a file that contains the initial key that is used to encrypt the password. Create this file that contains the initial key before you run the **runqmcrcd** command. The same initial key must be specified in the queue manager **INITKEY** attribute. The file must contain a single line of at least one character.

If this parameter is not specified, a default initial key is used.

## Examples

The following example encrypts the authentication token keystore passwords using the initial key that you provide.

Use the **-sf** argument to provide the initial key file path. You are prompted to enter the password you want to encrypt.

```
runqmcrcd -sf /home/initial.key
```

The encrypted password is output on the last line. Copy the encrypted password into a file and include the path in the **KeyStorePwdFile** attribute of the **AuthToken** stanza in the `qm.ini` file.

```
5724-H72 (C) Copyright IBM Corp. 1994, 2024.  
Enter password:  
*****  
<QM>!2!UnH/9hRXEGA0cenLVSGCW9a0s5A2vHDkTiA7vRv8ogc=!yh1sHFw7MIh48SvaYeTwRQ==
```

The following example encrypts the authentication token keystore password using a default initial key.

```
runqmcrcd
```

Copy the encrypted password that is output on the last line into a file. Include the path to the file in the **KeyStorePwdFile** attribute of the **AuthToken** stanza in the `qm.ini` file.

```
5724-H72 (C) Copyright IBM Corp. 1994, 2024.  
Credentials are encrypted using the default encryption key. For more secure  
protection of stored credentials, use a custom, strong encryption key.  
Enter password:  
*****  
<QM>!2!b5rb01sMzFzc1ClZeQMryruWFM3HSm8DKyEaZK7qzWY=!TrWdU57DCDXM0Qah99I/Lg==
```

## Return codes

**0**

Command completed successfully.

**1**

Command completed unsuccessfully.

## Related tasks

[Protecting passwords in IBM MQ component configuration files](#)

[Token stanza of the `qm.ini` file](#)

## Multi runswchl (switch cluster channel)

runswchl (switch cluster channel) on AIX, Linux, and Windows.

### Purpose

The command switches or queries the cluster transmission queues associated with cluster-sender channels.

### Usage notes

You must log on as an Administrator to run this command.

The command switches all the stopped or inactive cluster-sender channels that match the `-c` parameter, require switching, and can be switched. The command reports back on the channels that are switched, the channels that do not require switching, and the channels it cannot switch because they are not stopped or inactive.

If you set the `-q` parameter, the command does not perform the switch, but it provides the list of channels that would be switched.

### Syntax

```
runswchl -m ? QmgrName -c ? *
          -c ? GenericChannelName
          -c ? ChannelName
          -q
          -n
```

### Required parameters

#### **-m** *QmgrName*

The queue manager to run the command against. The queue manager must be started.

#### **-c** \*

All the cluster-sender channels

#### **-c** *GenericChannelName*

All matching cluster-sender channels

#### **-c** *ChannelName*

Single cluster-sender channel.

### Optional parameters

#### **-q**

Display the state of one or more channels. If you omit this parameter, the command switches any stopped or inactive channels that require switching.

#### **-n**

When switching transmission queues, do not transfer messages from the old queue to the new transmission queue.

**Note:** Take care with the `-n` option: messages on the old transmission queue are not transferred unless you associate the transmission queue with another cluster-sender channel.

### Return codes

#### **0**

The command completed successfully

#### **10**

The command completed with warnings.



## 20

The command completed with errors.

### Examples

To display the configuration state of cluster-sender channel TO.QM2:

```
RUNSWCHL -m QM1 -c TO.QM2 -q
```

To switch the transmission queue for cluster-sender channel TO.QM3 without moving the messages on it:

```
RUNSWCHL -m QM1 -c TO.QM3 -n
```

To switch the transmission queue for cluster-sender channel TO.QM3 and move the messages on it:

```
RUNSWCHL -m QM1 -c TO.QM3
```

To display the configuration state of all cluster-sender channels on QM1:

```
RUNSWCHL -m QM1 -c * -q
```

To display the configuration state of all cluster-sender channels with a generic name of TO.\*:

```
RUNSWCHL -m QM1 -c TO.* -q
```

### Related tasks



[Clustering: Switching cluster transmission queues](#)



#### **setmqaut (grant or revoke authority)**

Change the authorizations to a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

For more information about authorization service components, see [Configuring installable services](#), [Service components](#), and [Authorization service interface](#).

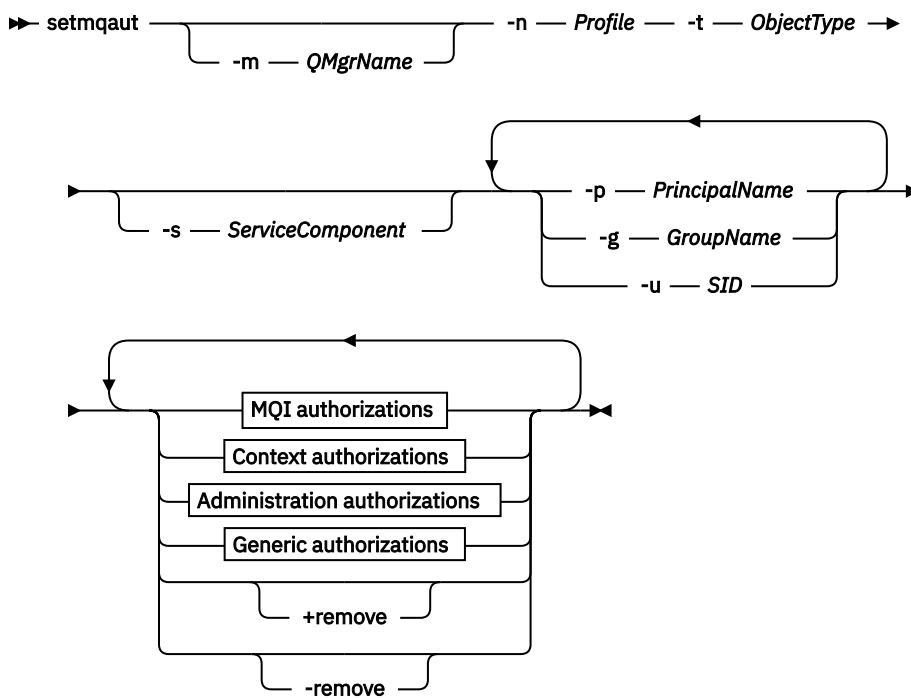
For more information about how authorizations work, see [How authorizations work](#).

  From IBM MQ 8.0, on UNIX and Linux systems, the object authority manager (OAM) can use user-based authorization as well as group-based authorization. For more information about user-based authorizations, see [OAM user-based permissions on AIX and Linux systems](#).

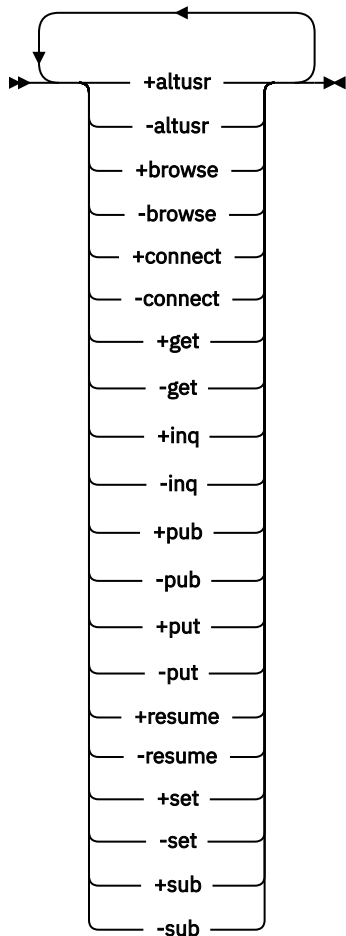
  If you specify the **-p** (principal) option, IBM MQ authorization uses groups, instead. This means that if you enter `setmqaut -p username . . .`, the primary group of the specified user is the one that is associated with the authorization being updated.

**Note:** The previous statement does not apply if you have configured your Object Authority Manager (OAM) to allow users.

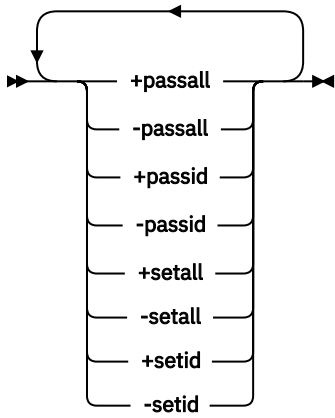
## Syntax



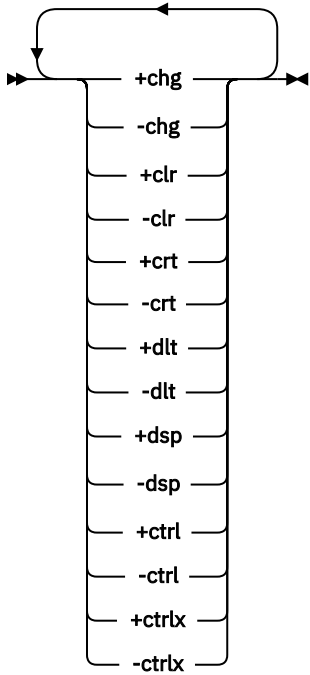
### MQI authorizations



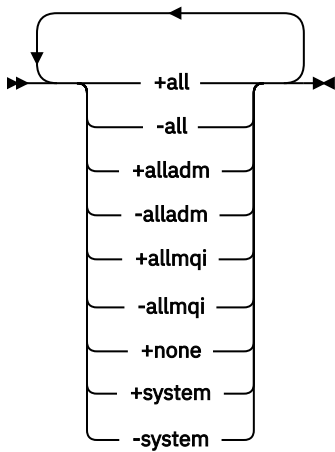
### Context authorizations



**Administration authorizations**



**Generic authorizations**



## Description

Use **setmqaut** both to grant an authorization, that is, give a principal or user group permission to perform an operation, and to revoke an authorization, that is, remove the permission to perform an operation. You can specify a number of parameters:

- Queue manager name
- Principals and user groups
- Object type
- Profile name
- Service component

The authorizations that can be given are categorized as follows:

- Authorizations for issuing MQI calls
- Authorizations for MQI context
- Authorizations for issuing commands for administration tasks
- Generic authorizations

Each authorization to be changed is specified in an authorization list as part of the command. Each item in the list is a string prefixed by a plus sign (+) or a minus sign (-). For example, if you include +put in the authorization list, you grant authority to issue MQPUT calls against a queue. Alternatively, if you include -put in the authorization list, you revoke the authority to issue MQPUT calls.

On AIX, Linux, and Windows, you can use the **SecurityPolicy** attribute to control the queue manager authorization:

- **Windows** On Windows systems, the **SecurityPolicy** attribute applies only if the service specified is the default authorization service, that is, the OAM. The **SecurityPolicy** attribute allows you to specify the security policy for each queue manager.
- **Linux** **AIX** On UNIX and Linux systems, for IBM MQ 8.0 and later, the value of the **SecurityPolicy** attribute specifies whether the queue manager uses user-based or group-based authorization. If you do not include this attribute, the default, which uses group-based authorization, is used.

For more information about the **SecurityPolicy** attribute, see [Configuring installable services](#), [Configuring authorization service stanzas on Windows](#), and [Configuring authorization service stanzas on UNIX and Linux](#).

For more information about the effect of the `user` and `group` settings of the **SecurityPolicy** attribute, see [OAM user-based permissions on UNIX and Linux systems](#).

You can specify any number of principals, user groups, and authorizations in a single **setmqaut** command, but you must specify at least one principal or user group.

If a principal is a member of more than one user group, the principal effectively has the combined authorities of all those user groups.

**Windows** On Windows systems, the principal also has all the authorities that are granted to it explicitly using the **setmqaut** command.

**Linux** **AIX** On AIX and Linux, if the **SecurityPolicy** attribute is set to `user`, the principal has all the authorities that are granted to it explicitly using the **setmqaut** command. However, if the **SecurityPolicy** attribute is set to `group` or `default`, or if the **SecurityPolicy** attribute is not set, all authorities are held by user groups internally, not by principals. Granting authorities to groups has the same implications as it did before IBM MQ 8.0:

- If you use the **setmqaut** command to grant an authority to a principal, the authority is granted to the primary user group of the principal. This means that the authority is effectively granted to all members of that user group.

- If you use the **setmqaut** command to revoke an authority from a principal, the authority is revoked from the primary user group of the principal. This means that the authority is effectively revoked from all members of that user group.

To alter authorizations for a cluster sender channel that has been automatically generated by a repository, see [Channel definition commands](#).

## Required parameters

### -t *ObjectType*

The type of object for which to change authorizations.

Possible values are as follows:

*Table 100. ObjectType values.*

Value	Description
<b>authinfo</b>	An authentication information object
<b>channel</b> or <b>chl</b>	A channel
<b>clntconn</b> or <b>clcn</b>	A client connection channel
<b>comminfo</b>	A communication information object
<b>listener</b> or <b>lstr</b>	A listener
<b>namelist</b> or <b>nl</b>	A namelist
<b>process</b> or <b>prcs</b>	A process
<b>queue</b> or <b>q</b>	A queue
<b>qmgr</b>	A queue manager
<b>rqmname</b> or <b>rqmn</b>	A remote queue manager name
<b>service</b> or <b>srvc</b>	A service
<b>topic</b> or <b>top</b>	A topic

### -n *Profile*

The name of the profile for which to change authorizations. The authorizations apply to all IBM MQ objects with names that match the profile name specified. The profile name can be generic, using wildcard characters to specify a range of names as explained in [Using OAM generic profiles on AIX, Linux, and Windows systems](#).

This parameter is required, unless you are changing the authorizations of a queue manager, in which case you must not include it. To change the authorizations of a queue manager use the queue manager name, for example

```
setmqaut -m QMGR -t qmgr -p user1 +connect
```

where *QMGR* is the name of the queue manager and *user1* is the principal for which you are adding or removing permissions.

Each class of object has authority records for each group or principal. These records have the profile name @CLASS and track the crt (create) authority common to all objects of that class. If the crt authority for any object of that class is changed then this record is updated. For example:

```
profile: @class
object type: queue
entity: test
```

```
entity type: principal
authority: crt
```

This shows that members of the group test have crt authority to the class queue.



**Attention:** You cannot delete the @CLASS entries (the system is working as designed)

## Optional parameters

### -m *QMgrName*

The name of the queue manager of the object for which to change authorizations. The name can contain up to 48 characters.

This parameter is optional if you are changing the authorizations of your default queue manager.

### -p *PrincipalName*

The name of the principal for which to change authorizations.

**Windows** For IBM MQ for Windows only, the name of the principal can optionally include a domain name, specified in the following format:

```
userid@domain
```

For more information about including domain names on the name of a principal, see [Principals and groups on UNIX, Linux and Windows](#).

You must have at least one principal or group.

### -g *GroupName*

The name of the user group for which to change authorizations. You can specify more than one group name, but each name must be prefixed by the -g flag.

**Windows** For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following formats:

```
GroupName@domain
domain\GroupName
```

The IBM MQ Object Authority Manager validates the users and groups at the domain level, only if you set the **GroupMode1** attribute to *GlobalGroups* in the [Securing](#) stanza of the queue manager.

### -u *SID*

The SID for which authorities are to be removed. You can specify more than one SID, but each name must be prefixed by the -u flag.

This option must be used with either +remove or -remove.

This parameter is only valid on IBM MQ for Windows.

### -s *ServiceComponent*

The name of the authorization service to which the authorizations apply (if your system supports installable authorization services). This parameter is optional; if you omit it, the authorization update is made to the first installable component for the service.

### +remove or -remove

Remove all the authorities from IBM MQ objects that match the specified profile.

### Authorizations

The authorizations to be granted or revoked. Each item in the list is prefixed by a plus sign (+) or a minus sign (-). The plus sign indicates that authority is to be granted. The minus sign indicates that authority is to be revoked.

For example, to grant authority to issue MQPUT calls, specify +put in the list. To revoke the authority to issue MQPUT calls, specify -put.

Table 101 on page 199 shows the authorities that can be given to the different object types.

Authority	Queue	Process	Queue manager	Remote queue manager name	Namespace	Topic	Auth info	Clientconn	Channel	Listener	Service	Commit info
all <sup>1</sup>	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
alladm <sup>2</sup>	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
allmqi <sup>3</sup>	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
none	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
altusr	No	No	Yes	No	No	No	No	No	No	No	No	No
browse	Yes	No	No	No	No	No	No	No	No	No	No	No
chg	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
clr	Yes	No	No	No	No	Yes	No	No	No	No	No	No
connect	No	No	Yes	No	No	No	No	No	No	No	No	No
crt	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ctrl <sup>4</sup>	No	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	No
ctrlx	No	No	No	No	No	No	No	No	Yes	No	No	No
dlt	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
dsp	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
get	Yes	No	No	No	No	No	No	No	No	No	No	No
pub	No	No	No	No	No	Yes	No	No	No	No	No	No
put	Yes	No	No	Yes	No	No	No	No	No	No	No	No
inq	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	No
passall	Yes	No	No	No	No	No	No	No	No	No	No	No
passid	Yes	No	No	No	No	No	No	No	No	No	No	No
resume	No	No	No	No	No	Yes	No	No	No	No	No	No
set	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No
setal <sup>5</sup>	Yes	No	Yes	No	No	No	No	No	No	No	No	No
setid <sup>5</sup>	Yes	No	Yes	No	No	Yes	No	No	No	No	No	No

Table 101. Specifying authorities for different object types (continued)

Authority	Queue	Process	Queue manager	Remote queue manager name	Name list	Topic	Auth info	Conn	Channel	Listener	Service	Comm info
sub	No	No	No	No	No	Yes	No	No	No	No	No	No
system	No	No	Yes	No	No	No	No	No	No	No	No	No

**Notes:**

1. The authority `all` is equivalent to the union of the authorities `alladm`, `allmqi`, and `system` appropriate to the object type.
2. The authority `alladm` is equivalent to the union of the individual authorities `chg`, `clr`, `dlt`, `dsp`, `ctrl`, and `ctrlx` appropriate to the object type. `crt` authority is not included in the subset `alladm`.
3. The authority `allmqi` is equivalent to the union of the individual authorities `altusr`, `browse`, `connect`, `get`, `inq`, `pub`, `put`, `resume`, `set`, and `sub` appropriate to the object type.
4. The authority `ctrl` on the `qmgr` object is included when the you specify `alladm` on the **setmqaut** command.
5. To use `setid` or `setall` authority, authorizations must be granted on both the appropriate queue object and also on the queue manager object. `setid` and `setall` are included in `allmqi`.

**Description of specific authorities**

You should not grant a user an authority (for example, `set` authority on a queue manager, or `system` authority) that allows the user to access IBM MQ privileged options, unless the required authority is specifically documented, and required to run any IBM MQ command, or IBM MQ API call.

For example, a user requires `system` authority to run the **setmqaut** command.

**chg**

A user needs `chg` authority to make any authorization changes on the queue manager. The authorization changes include:

- Changing the authorizations to a profile, object, or class of objects
- Creating and modifying channel authentication records, and so on

A user also needs `chg` authority to change or set the attributes of an IBM MQ object, using `PCF` or `MQSC` commands.

**ctrl**

Within `CHLAUTH` rules it is possible to insist that users connecting are not privileged.

For the channel to check whether a user is privileged, the real user id running the channel process must have `+ctrl` authority on the `qmgr` object.

For example, when the `SVRCONN` channel is running as a thread in an `amqrmppa` process and the real uid for this process is a `userid` named `mqadmin` (the `userid` that started the queue manager), then `mqadmin` must have `+ctrl` authority on the `qmgr` object.

**crt**

If you grant an entity `+crt` authority to the queue manager, then that entity also gains `+crt` authority for each object class.



However, when you remove `+crt` authority against the queue manager object that only removes the authority on the queue manager object class; `crt` authority for other objects classes are not removed.

Note that `crt` authority on the queue manager object has no functional use, and is available for backwards-compatibility purposes only.

#### **dlt**

Note that the `dlt` authority against the queue manager object has no functional use, and is available for backwards-compatibility purposes only.

#### **set**

A user needs `set` authority against the queue to change or set the attributes of a queue using the [MQSET](#) API call.

`set` authority on the queue manager is not required for any administrative purpose, or for any application connecting to the queue manager.

However, a user needs `set` authority against the queue manager to set privileged connection options.

Note that `set` authority on the process object has no functional use, and is available for backwards-compatibility purposes only.

**Important:** Privileged connection options are internal to the queue manager and are not available in IBM MQ API calls used by IBM MQ applications.

#### **system**

The `setmqaut` command makes a privileged IBM MQ connection to the queue manager.

Any user who runs IBM MQ commands that makes a privileged IBM MQ connection needs `system` authority on the queue manager.

## **Return codes**

*Table 102. Return code identifiers and descriptions*

<b>Return code</b>	<b>Explanation</b>
0	Successful operation
26	Queue manager running as a standby instance.
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
58	Inconsistent use of installations detected
69	Storage not available
71	Unexpected error
72	Queue manager name error
133	Unknown object name
145	Unexpected object name
146	Object name missing
147	Object type missing
148	Invalid object type
149	Entity name missing

Table 102. Return code identifiers and descriptions (continued)

Return code	Explanation
150	Authorization specification missing
151	Invalid authorization specification

### Examples

1. This example shows a command that specifies that the object on which authorizations are being given is the queue orange.queue on queue manager saturn.queue.manager.

```
setmqaut -m saturn.queue.manager -n orange.queue -t queue  
-g tango +inq +alladm
```

The authorizations are given to a user group called tango, and the associated authorization list specifies that the user group can:

- Issue MQINQ calls
  - Perform all administration operations on that object
2. In this example, the authorization list specifies that a user group called foxy:
    - Cannot issue any MQI calls to the specified queue
    - Can perform all administration operations on the specified queue

```
setmqaut -m saturn.queue.manager -n orange.queue -t queue  
-g foxy -allmqi +alladm
```

3. This example gives user1 full access to all queues with names beginning a.b. on queue manager qmgr1. The profile applies to any object with a name that matches the profile.

```
setmqaut -m qmgr1 -n a.b.* -t q -p user1 +all
```

4. This example deletes the specified profile.

```
setmqaut -m qmgr1 -n a.b.* -t q -p user1 -remove
```

5. This example creates a profile with no authority.

```
setmqaut -m qmgr1 -n a.b.* -t q -p user1 +none
```

### Related reference

[“dmpmqaut \(dump MQ authorizations\)” on page 55](#)

Dump a list of current authorizations for a range of IBM MQ object types and profiles.

[“DISPLAY AUTHREC \(display authority records\) on Multiplatforms” on page 646](#)

Use the MQSC command DISPLAY AUTHREC to display the authority records associated with a profile name.

[“SET AUTHREC \(set authority records\) on Multiplatforms” on page 920](#)

Use the MQSC command SET AUTHREC to set authority records associated with a profile name.

[OAM user-based permissions on AIX and Linux](#)

<i>Table 103. Authorizations for MQI calls.</i>	
<b>Value</b>	<b>Description</b>
altusr	Use another user's authority for the queue manager. Also required for channel operations where the asserting userid is different from the one associated with the connection handle. (For example, an assigned dedicated profile on the receiver MCA end, or when processing a RESET CHL SEQNUM() request from remote systems.)
browse	Retrieve a message from a queue using an MQGET call with the BROWSE option.
connect	Connect the application to the specified queue manager using an MQCONN call.
get	Retrieve a message from a queue using an MQGET call.
inq	Make an inquiry on a specific queue using an MQINQ call.
pub	Publish a message on a topic using the MQPUT call.
put	Put a message on a specific queue using an MQPUT call.
resume	Resume a subscription using the MQSUB call.
set	Set attributes on a queue from the MQI using an MQSET call.
sub	Create, alter or resume a subscription to a topic using the MQSUB call.

**Note:** If you open a queue for multiple options, you must be authorized for each option.

<i>Table 104. Authorizations for context.</i>	
<b>Value</b>	<b>Description</b>
passall	Pass all context on the specified queue. All the context fields are copied from the original request.
passid	Pass identity context on the specified queue. The identity context is the same as that of the request.
setall	Set all context on the specified queue. This is used by special system utilities.

Table 104. Authorizations for context. (continued)

Value	Description
setid	<p>Set identity context on the specified queue. This is used by special system utilities.</p> <p>In order to modify any of the message context options, you must have the appropriate authorizations to issue the call. For example, in order to use MQOO_SET_IDENTITY_CONTEXT or MQPMO_SET_IDENTITY_CONTEXT, you must have +setid permission.</p>

**Note:** To use setid or setall authority, authorizations must be granted on both the appropriate queue object and also on the queue manager object.

### Authorizations for commands

Table 105. Authorizations for commands.

Value	Description
chg	Change the attributes of the specified object.
clr	Clear the specified queue or a topic.
crt	Create objects of the specified type.
dlt	<p>Delete the specified object.</p> <p>Note, that the dlt authority has no effect on a queue manager object.</p>
dsp	Display the attributes of the specified object.
ctrl	<p>For listeners and services, start and stop the specified channel, listener, or service.</p> <p>For channels, start, stop, and ping the specified channel.</p> <p>For topics, define, alter, or delete subscriptions.</p>
ctrlx	Reset or resolve the specified channel.

### Authorizations for generic operations

Table 106. Authorizations for generic operations.

Value	Description
all	Use all operations applicable to the object. all authority is equivalent to the union of the authorities alladm, allmqi, and system appropriate to the object type.
alladm	Use all administration operations applicable to the object.
allmqi	Use all MQI calls applicable to the object.

Table 106. Authorizations for generic operations. (continued)

Value	Description
none	No authority. Use this authorization to create profiles without authority. When an authority is given to an object or group that was previously showing "none", then the authorization changes to the authority just applied. However, when the "none" authorization is added to an object or group with an existing alternative authority, the authority does not change.
system	Use queue manager for internal system operations.

## Windows **setmqcrl (set CRL LDAP server definitions)**

Administer certificate revocation list (CRL) LDAP definitions in an Active Directory ( Windows only).

### Purpose

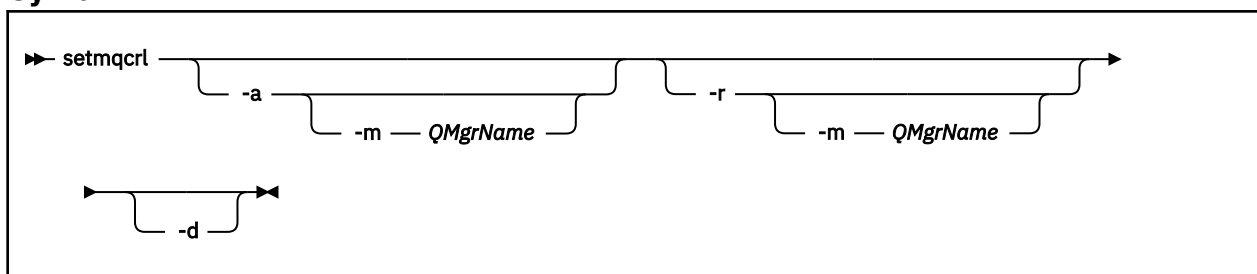
**Note:** The **setmqcrl** command applies to IBM MQ for Windows only.

Use the **setmqcrl** command to configure and administer support for publishing CRL (certificate revocation list) LDAP definitions in an Active Directory.

A domain administrator must use this command, or **setmqscpsetmqcrl**, initially to prepare the Active Directory for IBM MQ usage and to grant IBM MQ users and administrators the relevant authorities to access and update the IBM MQ Active Directory objects. You can also use the **setmqcrl** command to display all the currently configured CRL server definitions available on the Active Directory, that is, those definitions referred to by the queue manager's CRL namelist.

The only types of CRL servers supported are LDAP servers.

### Syntax



### Optional parameters

You must specify one of -a (add), -r (remove) or -d (display).

#### -a

Adds the IBM MQ MQI client connections Active Directory container, if it does not already exist. You must be a user with the appropriate privileges to create subcontainers in the *System* container of your domain. The IBM MQ folder is called CN=IBM-MQClientConnections. Do not delete this folder in any other way than by using the **setmqscp** command.

#### -d

Displays the IBM MQ CRL server definitions.

#### -r

Removes the IBM MQ CRL server definitions.

### **-m [ \* | qmgr ]**

Modifies the specified parameter (**-a** or **-r**) so that only the specified queue manager is affected. You must include this option with the **-a** parameter.

### **\* | qmgr**

\* specifies that all queue managers are affected. This enables you to migrate a specific IBM MQ CRL server definitions file from one queue manager alone.

## **Examples**

The following command creates the IBM-MQClientConnections folder and allocates the required permissions to IBM MQ administrators for the folder, and to child objects created subsequently. (In this, it is functionally equivalent to `setmqscp -a`.)

```
setmqcrl -a
```

The following command migrates existing CRL server definitions from a local queue manager, Paint.queue.manager, to the Active Directory.

**Note:** The command first deletes any other CRL definitions from the Active Directory.

```
setmqcrl -a -m Paint.queue.manager
```

### **Multi**

## **setmqenv (set IBM MQ environment)**

Use the **setmqenv** command to set up the IBM MQ environment on AIX, Linux, and Windows.

### **Purpose**

You can use the **setmqenv** command to automatically set up the environment for use with an installation of IBM MQ. Alternatively, you can use the **crtmqenv** command to create a list of environment variables and values to manually set each environment variable for your system; see [“crtmqenv \(create IBM MQ environment\)”](#) on page 34 for more information.

**Note:** Any changes you make to the environment are not persistent. If you log out, and log in again, your changes are lost.

You can specify which installation the environment is set up for by specifying a queue manager name, an installation name, or an installation path. You can also set up the environment for the installation that issues the **setmqenv** command by issuing the command with the **-s** parameter.

The **setmqenv** command sets the following environment variables, appropriate to your system:

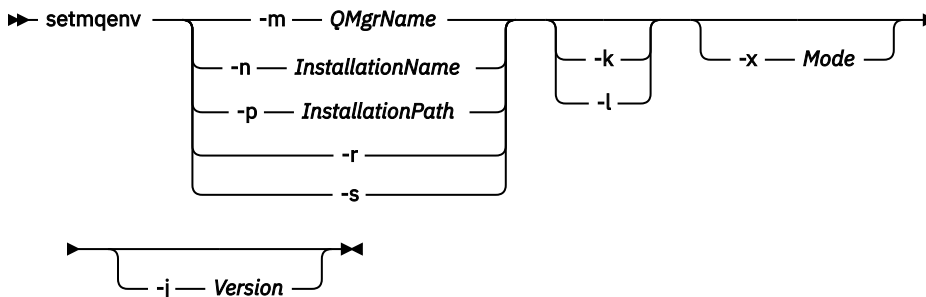
- CLASSPATH
- INCLUDE
- LIB
- MANPATH
- MQ\_DATA\_PATH
- MQ\_ENV\_MODE
- MQ\_FILE\_PATH
- MQ\_INSTALLATION\_NAME
- MQ\_INSTALLATION\_PATH
- MQ\_JAVA\_INSTALL\_PATH
- MQ\_JAVA\_DATA\_PATH
- MQ\_JAVA\_LIB\_PATH
- MQ\_JAVA\_JVM\_FLAG

- MQ\_JRE\_PATH
- PATH

## Usage notes

- The **setmqenv** command removes all directories for all IBM MQ installations from the environment variables before adding new references to the installation for which you are setting up the environment for. Therefore, if you want to set any additional environment variables that reference IBM MQ, set the variables after issuing the **setmqenv** command. For example, if you want to add *MQ\_INSTALLATION\_PATH/java/lib* to *LD\_LIBRARY\_PATH*, you must do so after running the **setmqenv** command.
- In some shells, command-line parameters cannot be used with **setmqenv** and any **setmqenv** command issued is assumed to be a **setmqenv -s** command. The command produces an informational message that the command has been run as if a **setmqenv -s** command had been issued. Therefore, in these shells you must ensure that you issue the command from the installation for which you want to set the environment for. In these shells, you must set the *LD\_LIBRARY\_PATH* variable manually. Use the **crtmqenv** command with the **-l** or **-k** parameter to list the *LD\_LIBRARY\_PATH* variable and value. Then use this value to set the *LD\_LIBRARY\_PATH*.

## Syntax



## Optional Parameters

### **-m QMgrName**

Set the environment for the installation associated with the queue manager *QMgrName*.

### **-n InstallationName**

Set the environment for the installation named *InstallationName*.

### **-p InstallationPath**

Set the environment for the installation in the path *InstallationPath*.

### **-r**

Remove all installations from the environment.

### **-s**

Set the environment for the installation that issued the **setmqenv** command.

### Linux AIX **-k**

Applies to AIX and Linux only. If the **-k** flag is specified:

- **AIX** On AIX, the *LIBPATH* environment variable is set.
- **Linux** On Linux, the *LD\_LIBRARY\_PATH* environment variable is set.

Include the *LD\_LIBRARY\_PATH* or *LIBPATH* environment variable in the environment, adding the path to the IBM MQ libraries at the start of the current *LD\_LIBRARY\_PATH* or *LIBPATH* variable.

### Linux AIX **-l**

Applies to AIX and Linux only. If the **-l** flag is specified:

- **AIX** On AIX, the `LIBPATH` environment variable is set.
- **Linux** On Linux, the `LD_LIBRARY_PATH` environment variable is set.

Include the `LD_LIBRARY_PATH` or `LIBPATH` environment variable in the environment, adding the path to the IBM MQ libraries at the end of the current `LD_LIBRARY_PATH` or `LIBPATH` variable.

#### -x Mode

`Mode` can take the value 32 or 64.

Create a 32-bit or 64-bit environment. If this parameter is not specified, the environment matches that of the queue manager or installation specified in the command.

Any attempt to display a 64-bit environment with a 32-bit installation fails.

#### **V 9.3.0** **V 9.3.0** -j Version

`Version` can take the value 2.0, or 3.0.

- **JMS 2.0** If you specify `-j 2.0` the `CLASSPATH` environment variable is changed to include the JAR files necessary to run JMS 2.0 applications. This is the default if `-j` is not specified.
- **JM 3.0** If you specify `-j 3.0` the `CLASSPATH` environment variable is changed to include the JAR files necessary to run Jakarta Messaging 3.0 applications.

IBM MQ 9.3.0 introduces support for [Jakarta Messaging 3.0](#). JMS 2.0 is still fully supported.

## Return codes

Table 107. Return code identifiers and descriptions

Return code	Description
0	Command completed normally.
10	Command completed with unexpected results.
20	An error occurred during processing.

## Examples

**Linux** **AIX** The following examples assume that a copy of IBM MQ is installed in the `/opt/mqm` directory on AIX and Linux systems.

### Notes:

- The period character ( `.` ) used at the beginning of each command makes the `setmqenv` script run in the current shell. Therefore, the environment changes made by the `setmqenv` script are applied to the current shell. Without the period character ( `.` ), the environment variables are changed in another shell, and the changes are not applied to the shell from which the command is issued.
- Some shells, for example the Ubuntu 18.04 default shell, do not pass parameters when you use the period character at the beginning of a command. If you are using such a shell, the following warning is issued and any parameters to the command are ignored:

```
$ . /opt/mqm/bin/setmqenv -sAMQ8588W: No parameter was detected.
```

The environment has been set for the installation from which the `setmqenv` command was issued.

To pass parameters into `setmqenv` for such shells, you must use the following environment variable: **MQ\_ENV\_OPTIONS=<your options>**.

For example, to remove IBM MQ from the environment, issue the command:

```
$ MQ_ENV_OPTIONS=-x . /opt/mqm/bin/setmqenv
```



- The following command sets up the environment for an installation installed in the /opt/mqm directory:

```
. /opt/mqm/bin/setmqenv -s
```

- The following command sets up the environment for an installation installed in the /opt/mqm2 directory, and includes the path to the installation at the end of the current value of the `LD_LIBRARY_PATH` variable:

```
. /opt/mqm/bin/setmqenv -p /opt/mqm2 -l
```

- The following command sets up the environment for queue manager QM1 in a 32-bit environment:

```
. /opt/mqm/bin/setmqenv -m QM1 -x 32
```

**Windows** The following example assumes that a copy of IBM MQ is installed in C:\Program Files\IBM\MQ on a Windows system. This command sets up the environment for an installation called Installation1:

```
"C:\Program Files\IBM\MQ\bin\setmqenv.cmd" -n Installation1
```

### Related concepts

[Multiple installations](#)

### Related tasks

[Choosing a primary installation](#)

### Related reference

“[crtmqenv \(create IBM MQ environment\)](#)” on page 34

Create a list of environment variables for an installation of IBM MQ, on AIX, Linux, and Windows.

## **Multi** **setmqinst (set IBM MQ installation)**

Set IBM MQ installations, on AIX, Linux, and Windows and IBM i.

### Purpose

You can use the **setmqinst** command to change the installation description of an installation, to set or unset an installation as the primary installation, or to set the entitlement of the installation. To change the primary installation, you must unset the current primary installation before you can set a new primary installation. This command updates information contained in the `mqinst.ini` file.

You can use the **dspmqinst** command to display the installations.

After unsetting the primary installation, the **setmqinst** command will not be available unless you specify the full path or have an appropriate installation directory on your PATH (or equivalent). The default path in a system standard location will have been deleted.

**Linux** **AIX** On AIX and Linux, you should not assume that the current directory is in the path. If you are in `/opt/mqm/bin` and want to run, for example, `/opt/mqm/bin/dspmqver`, you need to enter `"/opt/mqm/bin/dspmqver"` or `./dspmqver`.

File `mqinst.ini` contains information about all IBM MQ installations on a system. For more information about `mqinst.ini`, see [Installation configuration file, mqinst.ini](#).



**Attention:** Only the user `root` can run this command.

**ALW** On AIX and Linux, you must run this command as root. On Windows, you must run this command as a member of the Administrators group. The command does not have to be run from the installation you are modifying.

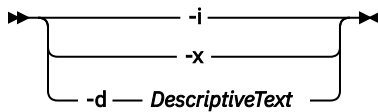
**Note:** By default, user mqm does not have sufficient authority to use this command.

**IBM i V 9.3.0** From IBM MQ 9.3.0, this command is supported on IBM i to set and unset HA Replica and Non-Production IBM License Metric Tool (ILMT) tags for an installation.

## Syntax



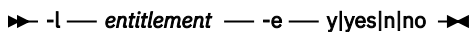
### Action



### Installation



### Entitlement



## Parameters

### -d *DescriptiveText*

Text that describes the installation.

The text can be up to 64 single-byte characters, or 32 double-byte characters. The default value is all blanks. You must use double quotation marks around the text if it contains spaces.

### -i

Set this installation as the primary installation.

### -x

Unset this installation as the primary installation.

### -n *InstallationName*

The name of the installation to modify.

### -p *InstallationPath*

The path of the installation to modify, for example, opt/mqm. You must use double quotation marks around the path if it contains spaces.

### -l *entitlement* -e y|yes|n|no

Set or unset the entitlement of the installation.

The entitlement can be set only if an applicable component is installed and entitlement is not already set to IBM MQ Advanced for Developers. *entitlement* is one of the following values:

#### **hareplica**

Sets the entitlement to High Availability Replica.

The hareplica entitlement cannot be set with the IBM MQ Advanced (Non-production) entitlement.

#### **advanced**

Sets the entitlement to IBM MQ Advanced.

#### **nonprod**

**V 9.3.4** For Continuous Delivery from IBM MQ 9.3.4, sets the entitlement to IBM MQ (Non-Production) or IBM MQ Advanced (Non-Production), depending on whether the installation is IBM MQ Advanced or not.

**LTS** For Long Term Support (and Continuous Delivery before IBM MQ 9.3.4), sets the entitlement to IBM MQ Advanced (Non-Production).

The nonprod entitlement cannot be set with the High Availability Replica entitlement. The server component must be installed to set this entitlement.

**IBM i V 9.3.0** From IBM MQ 9.3.0, you can set the options of hareplica and nonprod for *entitlement* on IBM i.

The entitlement is picked up automatically by the IBM License Metric Tool (ILMT) after it is set. See [IBM MQ license information](#).

Use **-e y** or **-e yes** to set the entitlement.

Use **-e n** or **-e no** to unset the entitlement.

## Return codes

Table 108. Return code identifiers and descriptions

Return code	Description
0	Entry set without error
36	Invalid arguments supplied
37	Descriptive text was in error
44	Entry does not exist
59	Invalid installation specified
71	Unexpected error
89	ini file error
96	Could not lock ini file
98	Insufficient authority to access ini file
131	Resource problem

## Examples

1. This command sets the installation with the name of myInstallation as the primary installation:

```
setmqinst -i -n myInstallation
```

2. This command sets the installation with an installation path of /opt/myInstallation as the primary installation:

```
setmqinst -i -p /opt/myInstallation
```

3. This command unsets the installation named myInstallation as the primary installation:

```
setmqinst -x -n myInstallation
```

4. This command unsets the installation with an installation path of /opt/myInstallation as the primary installation:

```
setmqinst -x -p /opt/myInstallation
```

5. This command sets the descriptive text for the installation named myInstallation:

```
setmqinst -d "My installation" -n myInstallation
```

The descriptive text is enclosed in quotation marks as it contains spaces.

6. This command specifies that the installation at /opt/myInstallation has High Availability Replica entitlement:

```
setmqinst -l hareplica -e yes -p /opt/myInstallation
```

7. This command specifies that the installation myInstallation does not have High Availability Replica entitlement:

```
setmqinst -l hareplica -e no -n myInstallation
```

8. This command specifies that the installation myInstallation has non-production entitlement:

```
setmqinst -l nonprod -e y -n myInstallation
```

9. This command specifies that the installation at /opt/myInstallation does not have non-production entitlement:

```
setmqinst -l nonprod -e n -p /opt/myInstallation
```

### Related tasks

[Choosing a primary installation](#)

[Changing the primary installation](#)

## **setmqm (set queue manager)**

Set the associated installation of a queue manager.

### Purpose

Use the **setmqm** command to set the associated IBM MQ installation of a queue manager. The queue manager can then be administered using only the commands of the associated installation. For example, when a queue manager is started with **strmqm**, it must be the **strmqm** command of the installation that was specified by the **setmqm** command.

For more information about using this command, including information about when to use it, see [Associating a queue manager with an installation](#).

This command is only applicable to AIX, Linux, and Windows.

### Usage notes

- You must use the **setmqm** command from the installation you want to associate the queue manager with.
- The installation name specified by the **setmqm** command must match the installation from which the **setmqm** command is issued.
- You must stop the queue manager before executing the **setmqm** command. The command fails if the queue manager is still running.
- Once you have set the associated installation of a queue manager using the **setmqm** command, migration of the queue manager's data occurs when you start the queue manager using the **strmqm** command.

- Once you have started the queue manager on an installation, you cannot then use **setmqm** to set the associated installation to an earlier version of IBM MQ, as it is not possible to migrate back to earlier versions of IBM MQ.
- You can find out which installation is associated with a queue manager by using the **dspmqs** command. See [“dspmqs \(display queue managers\)”](#) on page 73 for more information.

## Syntax

```
➤ setmqm — -m — QMgrName — -n — InstallationName ➤
```

## Required Parameters

### -m *QMgrName*

The name of the queue manager to set the associated installation for.

### -n *InstallationName*

The name of the installation that the queue manager is to be associated with. The installation name is not case-sensitive.

## Return codes

Table 109. Return code identifiers and descriptions

Return code	Description
0	Queue manager set to an installation without error
5	Queue manager running
36	Invalid arguments supplied
59	Invalid installation specified
60	Command not executed from the installation named by the -n parameter
61	Invalid installation name for this queue manager
69	Resource problem
71	Unexpected error
72	Queue manager name error
119	User not authorized

## Examples

1. This command associates a queue manager QMGR1, with an installation with the installation name of myInstallation.

```
MQ_INSTALLATION_PATH/bin/setmqm -m QMGR1 -n myInstallation
```

## Multi **setmqprd (enroll production license)**

Enroll an IBM MQ production license.

A license is normally enrolled as part of the installation process.

**Note:** You must have the appropriate privileges to run this command on your system. AIX and Linux require root access, and Windows with UAC (User Account Control) requires Administrator access to run this command.

## Syntax

```
► setmqprd — LicenseFile ◄
```

## Required parameters

### LicenseFile

Specifies the fully-qualified name of the production license certificate file.

The full license file is `amqpcert.lic`:

- **Linux** ► **AIX** On AIX and Linux, it is in the `/MediaRoot/licenses` directory on the installation media.
- **Windows** On Windows it is in the `\MediaRoot\licenses` directory on the installation media. It is installed into the `bin` directory on the IBM MQ installation path.
- **IBM i** On IBM i, issue the command

```
CALL PGM(QMQM/SETMQPRD) PARM('LICENSE_PATH/amqpcert.lic')
```

where `LICENSE_PATH` is the path to the `amqpcert.lic` file that you obtained.

## Trial license conversion

A trial license installation is identical to a production license installation, except for the "count-down" message that is displayed when you start a queue manager on an installation with a trial license. Parts of IBM MQ that are not installed on the server, such as the IBM MQ MQI client, continue to work after the expiry of the trial license. You do not need to run **setmqprd** to enroll them with a production license.

When a trial license expires, you can still uninstall IBM MQ. You can also reinstall IBM MQ with a full production license.

Run **setmqprd** to enroll a production license after installing and using a installation with a trial license.

### Related tasks

- **AIX** [Converting a trial license on AIX](#)
- **IBM i** [Converting a trial license on IBM i](#)
- **Linux** [Converting a trial license on Linux](#)
- **Windows** [Converting a trial license on Windows](#)

### ► **Windows** **setmqscp (set service connection points)**

Publish client connection channel definitions in an Active Directory ( Windows only).

## Purpose

**Note:** The **setmqscp** command applies to IBM MQ for Windows only.

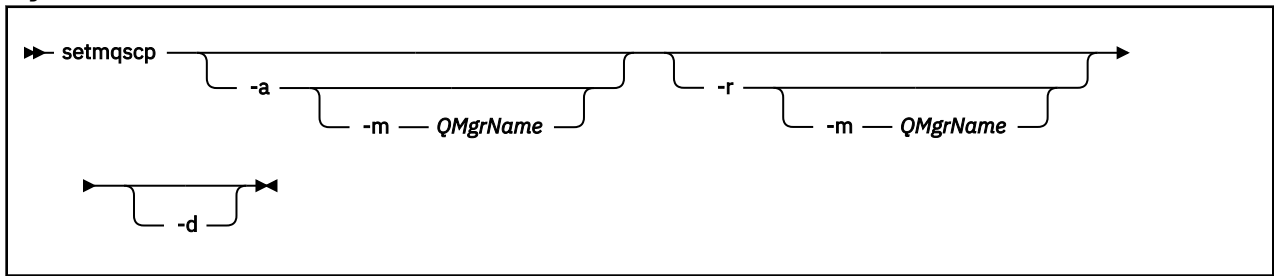
Use the **setmqscp** command to configure and administer support for publishing client connection channel definitions in an Active Directory.

Initially, this command is used by a domain administrator to:

- Prepare the Active Directory for IBM MQ use
- Grant IBM MQ users and administrators the relevant authorities to access and update the IBM MQ Active Directory objects

You can also use the **setmqscp** command to display all the currently configured client connection channel definitions available on the Active Directory.

## Syntax



## Optional parameters

You must specify one of -a (add), -r (remove) or -d (display).

### -a

Adds the IBM MQ MQI client connections Active Directory container, if it does not already exist. You must be a user with the appropriate privileges to create subcontainers in the *System* container of your domain. The IBM MQ folder is called CN=IBM-MQClientConnections. Do not delete this folder in any other way than by using the `setmqscp -r` command.

### -d

Displays the service connection points.

### -r

Removes the service connection points. If you omit **-m**, and no client connection definitions exist in the IBM-MQClientConnections folder, the folder itself is removed from the Active Directory.

### -m [ \* | qmgr ]

Modifies the specified parameter (-a or -r) so that only the specified queue manager is affected.

#### \* | qmgr

\* specifies that all queue managers are affected. This enables you to migrate a specific client connection table file from one queue manager alone, if required.

## Examples

The following command creates the IBM-MQClientConnections folder and allocates the required permissions to IBM MQ administrators for the folder, and to child objects created subsequently:

```
setmqscp -a
```

The following command migrates existing client connection definitions from a local queue manager, Paint.queue.manager, to the Active Directory:

```
setmqscp -a -m Paint.queue.manager
```

The following command migrates all client connection definitions on the local server to the Active Directory:

```
setmqscp -a -m *
```

## setmqspl (set security policy)

Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy. **z/OS** On z/OS you use the command with the CSQOUTIL utility.

### Before you begin

The queue manager on which you want to operate must be running.

**Multi** You must grant the necessary +connect, +inq and +chg authorities, using the [setmqaut](#) command, to connect to the queue manager and create a security policy.

**z/OS** For more information on the authorities needed to run this command on z/OS see [“Specific security information”](#) on page 2810 in the CSQOUTIL topic.

For more information about configuring security see [Setting up security](#).

### Syntax

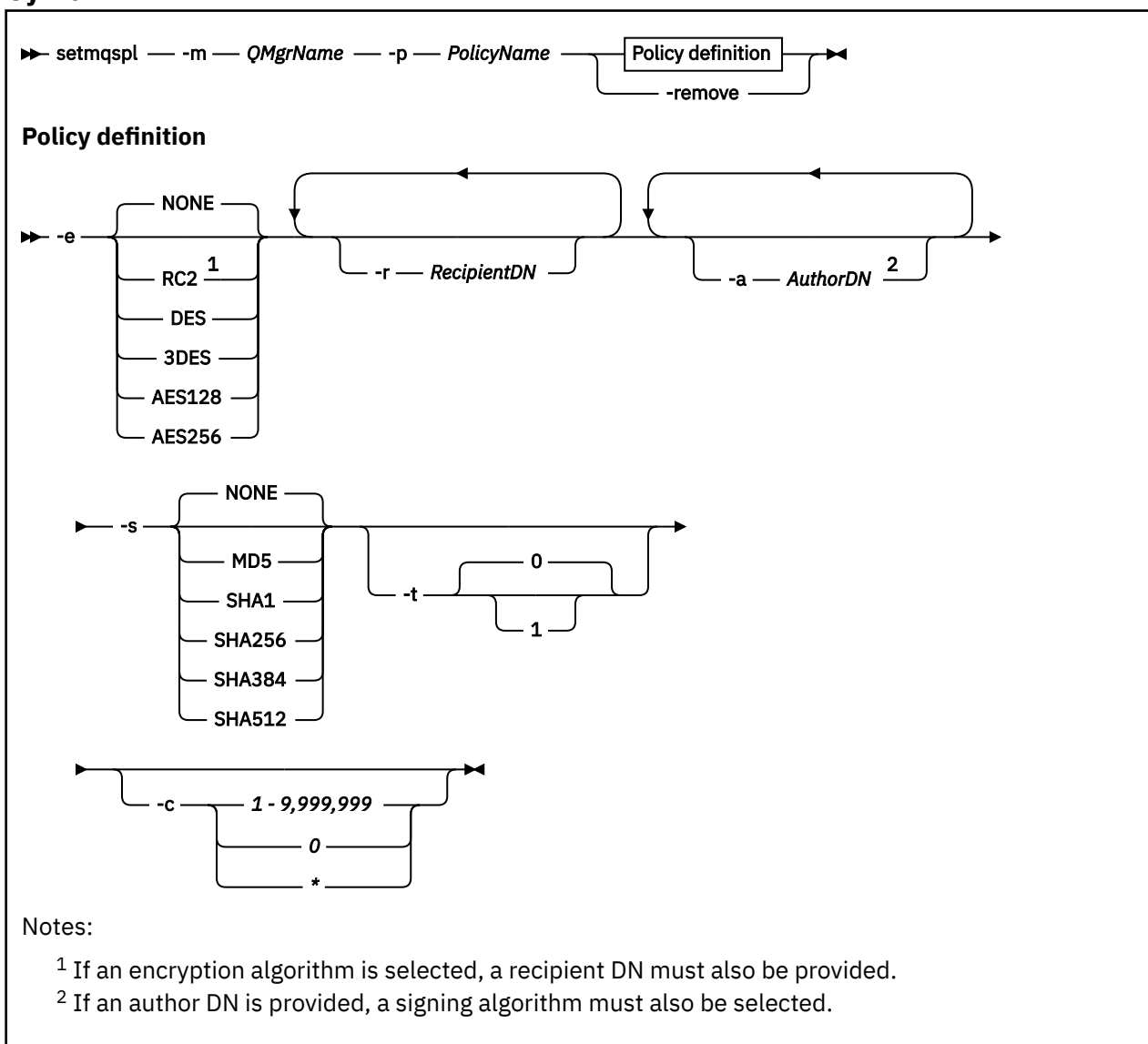




Table 110. `setmqsp` command flags






Command flag	Explanation
-m	<p>Queue manager name.</p> <p>This flag is mandatory for all actions on security policies.</p>
-p	<p>Policy name.</p> <p>Set the policy name to the name of the queue you want the policy to apply to.</p>
-e	<p>Digital encryption algorithm.</p> <p>Advanced Message Security supports the following encryption algorithms:</p> <ul style="list-style-type: none"> <li>•  <a href="#">RC2</a></li> <li>•  <a href="#">DES</a></li> <li>•  <a href="#">3DES</a></li> <li>• AES128</li> <li>• AES256</li> </ul> <p>The default value is NONE.</p> <p><b>Important:</b></p> <ul style="list-style-type: none"> <li>• The name of the encryption algorithm must be provided in uppercase</li> <li>•  On z/OS encryption algorithm  <a href="#">RC2</a> is not supported for confidentiality policies.</li> </ul>
-r	<p>The distinguished name (DN) of the message recipient (if provided, the certificate pertaining to the DN is used to encrypt a given message). Recipients can be specified, only if the encryption algorithm is different from NONE. Multiple recipients can be included for a message. Each DN must be provided with a separate <code>-r</code> flag.</p> <p><b>Important:</b></p> <ul style="list-style-type: none"> <li>• DN attribute names must be in uppercase.</li> <li>• Commas must be used as a name separators.</li> <li>• To avoid command interpreter errors, place quotation marks around the DNs.</li> </ul> <p>For example:</p> <pre data-bbox="873 1713 1468 1799">-r "CN=alice, O=ibm, C=US"</pre>

Table 110. `setmqsp` command flags (continued)



Command flag	Explanation
<p><b>-a</b></p>	<p>Signature DN that is validated during message retrieval. Only messages signed by a user with a provided DN are accepted during the retrieval. Signature DNs can be specified only if the signature algorithm is different from NONE. Multiple authorized signers can be specified, each authorized signer needs to have a separate <b>-a</b> flag.</p> <p><b>Important:</b> The attribute in the DN name must be in uppercase. Specify CN= rather than cn=.</p> <p>The attribute values in the DN are case sensitive so, for example, CN=USERID1 is different from CN=userid1.</p>
<p><b>-s</b></p>	<p>Digital signature algorithm.</p> <p>Advanced Message Security supports the following values:</p> <ul style="list-style-type: none"> <li>•  <a href="#">MD5</a></li> <li>•  <a href="#">SHA-1</a></li> <li>• SHA-2 family: <ul style="list-style-type: none"> <li>– SHA256</li> <li>– SHA384 (minimum key length acceptable - 768 bits)</li> <li>– SHA512 (minimum key length acceptable - 768 bits)</li> </ul> </li> </ul> <p>All must be in uppercase. The default value is NONE.</p> <p><b>Important:</b></p> <ul style="list-style-type: none"> <li>• For the SHA384 and SHA512 cryptographic hash functions, keys used for signing must be longer than 768 bits.</li> <li>• The name of the signature algorithm must be provided in uppercase.</li> <li>• From IBM MQ 9.0, with the Confidentiality policy, the signature algorithm must be NONE. For more information about the Confidentiality policy, see <a href="#">Qualities of protection available with AMS</a>.</li> </ul>

Table 110. `setmqsp1` command flags (continued)

Command flag	Explanation
<b>-t</b>	<p>The toleration flag indicates whether messages that do not meet the requirements of the policy can still be successfully browsed or retrieved by an application. Toleration may be useful for example when introducing a policy to a queue which already contains unprotected messages. Valid values include:</p> <ul style="list-style-type: none"> <li>• <b>0 (default)</b> Toleration flag off.</li> <li>• <b>1</b> Toleration flag on.</li> </ul> <p>Toleration is optional and facilitates staged implementation, where policies were applied to queues but those queues may already contain messages that have no policy, or still receive messages from remote systems that do not have the security policy set.</p>
<b>-c</b>	<p>The key reuse count can be provided as an integer from 1 through 9,999,999. Special values are:</p> <ul style="list-style-type: none"> <li>• <b>0</b> Keys are not reused.</li> <li>• <b>*</b> Allows applications to reuse an encryption key an unlimited number of times.</li> </ul> <p>If you omit the <b>-c</b> parameter when defining a policy, a key reuse count of 0 is assumed for backwards compatibility with previous versions of Advanced Message Security and IBM WebSphere MQ Extended Security Edition.</p> <p>Note that a non-zero key reuse count is only valid for a confidentiality policy. If you attempt to create or modify an integrity or privacy policy, with a non-zero key reuse count, you receive error message AMQ9091: Key reuse is not valid for policy and the policy operation fails.</p>
<b>-remove</b>	<p>Delete policy.</p> <p>Only the policy name flag, <b>-p</b> is valid for use in combination with this flag.</p>

## Examples

The following list shows examples of some valid **setmqsp1** commands on Multiplatforms:

```
setmqsp1 -m QMGR -p PROT -s SHA256
setmqsp1 -m QMGR -p PROT -s SHA256 -a "CN=Alice, O=IBM, C=US"
setmqsp1 -m QMGR -p PROT -s SHA256 -e AES128 -a "CN=Alice, O=IBM, C=US" -r "CN=Bob, O=IBM, C=GB"
setmqsp1 -m QMGR -p PROT -e AES128 -r "CN=Bob, O=IBM, C=GB" -c 50
```

The following list shows examples of **setmqsp1** commands that are not valid:

- No recipients specified:

```
setmqsp1 -m QMGR -p PROT -e AES128
```

- Key reuse not valid for an Integrity policy:

```
setmqsp1 -m QMGR -p PROT -s SHA256 -c 1
```

- Key reuse is not valid for a Privacy policy:

```
setmqsp1 -m QMGR -p PROT -s SHA256 -e AES128 -r "CN=Bob, O=IBM, C=GB" -c 1
```

### Related reference


“[SET POLICY \(set security policy\) on Multiplatforms](#)” on page 938

Use the MQSC command SET POLICY to set a security policy.

“[DISPLAY POLICY \(display a security policy\) on Multiplatforms](#)” on page 755

Use the MQSC command **DISPLAY POLICY** to display a security policy.


“[dspmqsp1 \(display security policy\)](#)” on page 99

Use the **dspmqsp1** command to display a list of all policies and details of a named policy.  On z/OS you use the command with the CSQOUTIL utility.

## setmqweb pid (set mqweb server product ID)

Configure the product ID (PID) that the mqweb server runs under on z/OS.

### Purpose

 You can use the **setmqweb pid** command to change the PID that the mqweb server runs under. By default, on z/OS, the mqweb server runs under the PID that is chosen when you use the **crtmqweb** command.

Before using **setmqweb pid**, ensure that the mqweb server has been stopped by using the MVS™ **STOP** command on the mqweb server started task. For more information on PIDs and how they are used on z/OS, see [Product usage recording with IBM MQ for z/OS products](#).

### Using the command on z/OS



Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the WLP\_USER\_DIR environment variable so that the variable points to your mqweb server configuration.

To set the WLP\_USER\_DIR environment variable, enter the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP\_user\_directory* is the name of the directory that is passed to **crtmqweb**. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

You must also set the JAVA\_HOME environment variable to reference a 64-bit version of Java on your system.

The user ID running the command needs write access to the following directories:

- *WLP\_user\_directory* and its subdirectories.
- /tmp or to another directory that is referenced by the *TMPDIR* variable. If you do not have access to /tmp, the command fails with message FSUMF315 Cannot define temporary file. If you

need to set the *TMPDIR* variable, issue the following command in the z/OS UNIX shell: `export TMPDIR=user_directory`

## Syntax

z/OS

►► `setmqweb` — `pid` — `-p` — `pid_name` — `-l`

## Parameters

z/OS

Before using `setmqweb pid`, ensure that the mqweb server has been stopped by using the MVS **STOP** command on the mqweb server started task.

### -p *pid\_name*

Specifies the PID that the mqweb server runs under. *pid\_name* is one of the following values:

#### **MQ**

The mqweb server runs under IBM MQ for z/OS (5655-MQ9)

#### **VUE**

The mqweb server runs under IBM MQ for z/OS Value Unit Edition (5655-VU9)

#### **ADVANCEDVUE**

The mqweb server runs under IBM MQ Advanced for z/OS VUE (5655-AV1)

### -l

Enable verbose logging. Diagnostic information is written to an mqweb server log file.

## Return codes

Table 111. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

For a full list of server command exit codes, see [Liberty:server command options](#) in the WebSphere Application Server documentation.

## Related commands

Table 112. Related commands and descriptions

Command	Description
<a href="#">“setmqweb properties (set mqweb server configuration properties)”</a> on page 222	Configure the mqweb server properties.
<a href="#">“setmqweb remote (set mqweb server remote queue manager configuration)”</a> on page 230	Configure the mqweb server remote queue manager connections.
<a href="#">strmqweb</a>	Start the mqweb server.

Table 112. Related commands and descriptions (continued)

Command	Description
<a href="#">endmqweb</a>	Stop the mqweb server.
<a href="#">dspmqweb status</a>	Display the status of the mqweb server.
<a href="#">dspmqweb properties</a>	Display the mqweb server configuration properties.
<a href="#">dspmqweb remote</a>	Display the mqweb server remote queue manager configuration.

## setmqweb properties (set mqweb server configuration properties)

Configure the mqweb server properties.

### Purpose

You can use the **setmqweb properties** command to configure the mqweb server. Changes to properties take effect dynamically, within a few seconds, unless otherwise stated.

### Using the command on z/OS



Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, enter the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where `WLP_user_directory` is the name of the directory that is passed to **crtmqweb**. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

You must also set the `JAVA_HOME` environment variable to reference a 64-bit version of Java on your system.

The user ID running the command needs write access to the following directories:

- `WLP_user_directory` and its subdirectories.
- `/tmp` or to another directory that is referenced by the `TMPDIR` variable. If you do not have access to `/tmp`, the command fails with message FSUMF315 Cannot define temporary file. If you need to set the `TMPDIR` variable, issue the following command in the z/OS UNIX shell: `export TMPDIR=user_directory`

When the **setmqweb properties** command is used to modify the mqweb server configuration, the owner of the `mqwebuser.xml` file is changed to the user ID that issued the command, and the file permissions are set to the permissions that are indicated by the user's **umask**.

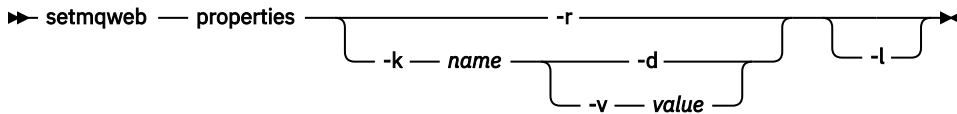
### Using the command in a stand-alone IBM MQ Web Server installation



Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ\_OVERRIDE\_DATA\_PATH** environment variable to the IBM MQ Web Server data directory.

The user ID running the command needs write access to the data directory and its subdirectories.

## Syntax



## Parameters

### -r

Reset to default values. This parameter removes all user-modified configuration properties from the `mqwebuser.xml` file.

### -k *name*

The name of the configuration property to add, update, or remove to or from the `mqwebuser.xml` file. The following values are the valid values for *name* on all platforms, including the IBM MQ Appliance.

**V 9.3.5** Some of the following properties are not available in a stand-alone IBM MQ Web Server installation.

#### **ltpaExpiration**

This configuration property is used to specify the time, in minutes, before the LTPA token expires.

The value for this property is an integer value. The default value is 120 minutes.

#### **maxTraceFiles**

This configuration property is used to specify the maximum number of mqweb server log files that are generated by the mqweb server.

The value for this property is an integer value. The default value is 2.

#### **maxTraceFileSize**

This configuration property is used to specify the maximum size, in MB, that each mqweb server log file can reach.

The value for this property is an integer value. The default value is 200.

#### **V 9.3.4 mqConsoleEnableDashboardBrowse**

Some aspects of the [MQ Console dashboard](#) contains information that is only available by browsing queues. This queue browsing is enabled by default in the IBM MQ Console. If a particular user does not have the correct authority to browse queues, many log entries can be generated, recording the failed access. To reduce the load on the logs, you can disable this queue browsing by setting this property to the string value "false".

#### **V 9.3.4 mqConsoleEnableSystemTopicMonitoring**

This configuration property is used to enable or disable system topic monitoring that is used to display system information in the IBM MQ Console (see [Metrics published on the system topics](#)). If system topic monitoring is enabled, more information is available to display in the queue manager **Overview** tab in the IBM MQ Console. See [Quick tour of the MQ Console](#).

The value for this property is a string value and is set to "true" to enable system topic monitoring or "false" to disable it.

#### **V 9.3.0 mqConsoleMaxMsgCharsToDisplay**

This configuration property is used to specify the maximum characters to retrieve from each message when you browse a queue by using the IBM MQ Console.

The value for this property is an integer. The default value is 1024.

#### **V 9.3.0 mqConsoleMaxMsgRequestSize**

This configuration property is used to specify the maximum size, in MB, a browse request can be across all messages when you browse queues by using the IBM MQ Console.

The value for this property is an integer. The default value is 1.

### ▶ V 9.3.0 **mqConsoleMaxMsgsPerRequest**

This configuration property is used to specify the total number of messages to retrieve from a queue when you browse by using the IBM MQ Console.

The value for this property is an integer. The default value is 1000.

### **mqRestCorsAllowedOrigins**

This configuration property is used to specify the origins that are allowed to access the REST API. For more information about CORS, see [Configuring CORS for the REST API](#).

The value for this property is a string value.

### **mqRestCorsMaxAgeInSeconds**

This configuration property is used to specify the time, in seconds, that a web browser can cache the results of any CORS pre-flight checks.

The value for this property is an integer value. The default value is 0.

### **mqRestCsrftValidation**

This configuration property is used to specify whether CSRF validation checks are performed. A value of `false` removes the CSRF token validation checks.

The value for this property is a Boolean value. The default value is `true`.

### **mqRestGatewayEnabled**

This configuration property is used to specify whether the administrative REST API gateway is enabled.

The value for this property is a Boolean value. The default value is `true`.

▶ V 9.3.5 This property is not valid in a stand-alone IBM MQ Web Server installation as the administrative REST API is not available in this environment.

### **mqRestGatewayQmgr**

This configuration property is used to specify the name of the queue manager to use as the gateway queue manager. This queue manager must be in the same installation as the mqweb server. A blank value indicates that no queue manager is configured as the gateway queue manager.

The value for this property is a string value. If this value can be interpreted as number or a Boolean value, it must be enclosed in double quotation marks.

▶ V 9.3.5 This property is not valid in a stand-alone IBM MQ Web Server installation as the administrative REST API is not available in this environment.

### ▶ V 9.3.2 **mqRestMessagingAdoptWebUserContext**

This configuration property is used to specify the user context that is used for authorization when you send, publish, receive, or browse messages by using the messaging REST API. That is, it specifies which user ID is used for authorization.

The value can be one of the following values:

#### **true**

The ID that is used for authorization is the user ID that is logged in to the REST API.

**MQMD.UserIdentifier** is set to the user ID that is logged in to the REST API, and the

**MQMD.AppIdentityData** is set to the user ID that is logged in to the REST API.

See [MQMD](#) for more information about the message descriptor parts of the IBM MQ message.

#### **false**

The ID that is used for authorization is the user ID that is used to start the mqweb server. The **MQMD.UserIdentifier** is left blank, and the **MQMD.AppIdentityData** is set to the user ID that is logged in to the REST API.

The value for this property is a Boolean value. The default value is `true`.

### **mqRestMessagingEnabled**

This configuration property is used to specify whether the messaging REST API is enabled.

The value for this property is a Boolean value. The default value is `true`.



### **mqRestMessagingFullPoolBehavior**

This configuration property is used to specify the behavior of the messaging REST API when all connections in the connection pool are in use.

The value can be one of the following values:

#### **block**

When all the connections in the pool are in use, wait for a connection to become available. When this option is used, the wait for a connection is indefinite.

Inactive connections are closed and removed from a queue manager pool automatically. The state of each queue manager pool is interrogated every 2 minutes, and any connections that have been inactive for the last 30 seconds are closed and removed from the associated pool.

#### **error**

When all the connections in the pool are in use, return an error.

#### **overflow**

When all the connections in the pool are in use, create a nonpooled connection to use. This connection is deleted after it is used.

The value for this property is a string value. The default value is overflow.

### **mqRestMessagingMaxPoolSize**

This configuration property is used to specify the maximum connection pool size for each queue manager connection pool.

The value for this property is an integer value. The default value is 20.

### **mqRestMftCommandQmgr**

This configuration property is used to specify the name of the command queue manager to which create transfer and create, delete, or update resource monitor requests are submitted by the REST API for MFT.

The value for this property is a string value. If this value can be interpreted as number or a Boolean value, it must be enclosed in double quotation marks.

Changes to the value of this property take effect when the mqweb server is next started.

**V 9.3.5** This property is not valid in a stand-alone IBM MQ Web Server installation as the REST API for MFT is not available in this environment.

### **mqRestMftCoordinationQmgr**

This configuration property is used to specify the name of the coordination queue manager from which transfer details are retrieved by the REST API for MFT.

The value for this property is a string value. If this value can be interpreted as number or a Boolean value, it must be enclosed in double quotation marks.

Changes to the value of this property take effect when the mqweb server is next started.

**V 9.3.5** This property is not valid in a stand-alone IBM MQ Web Server installation as the REST API for MFT is not available in this environment.

### **mqRestMftEnabled**

This configuration property is used to specify whether the REST API for MFT is enabled.

The value for this property is a Boolean value. The default value is false.

Changes to the value of this property take effect when the mqweb server is next started.

**V 9.3.5** This property is not valid in a stand-alone IBM MQ Web Server installation as the REST API for MFT is not available in this environment.

### **mqRestMftReconnectTimeoutInMinutes**

This configuration property is used to specify the length of time, in minutes, after which the REST API for MFT stops trying to connect to the coordination queue manager.

The value for this property is an integer value. The default value is 30.

Changes to the value of this property take effect when the mqweb server is next started.

**V 9.3.5** This property is not valid in a stand-alone IBM MQ Web Server installation as the REST API for MFT is not available in this environment.

### **mqRestRequestTimeout**

This configuration property is used to specify the time, in seconds, before a REST request times out.

The value for this property is an integer value. The default value is 30.

### **traceSpec**

This configuration property is used to specify the level of trace that is generated by the mqweb server. For a list of possible values, see [Configuring logging for the IBM MQ Console and REST API](#).

The value for this property is a string value. The default value is `*=info`.



The following values are the additional valid values for *name* on z/OS, UNIX, Linux, and Windows.

**V 9.3.5** Some of the following properties are not available in a stand-alone IBM MQ Web Server installation.

### **httpHost**

This configuration property is used to specify the HTTP hostname as an IP address, domain name server (DNS) hostname with domain name suffix, or the DNS hostname of the server where IBM MQ is installed.

You can use an asterisk in double quotation marks to specify all available network interfaces.

You can use the value `localhost` to allow only local connections.

The value for this property is a string value. The default value is `localhost`.

### **httpPort**

This configuration property is used to specify the HTTP port number that is used for HTTP connections.

You can use a value of `-1` to disable the port.

The value for this property is an integer value. The default value is `-1`.

### **httpsPort**

This configuration property is used to specify the HTTPS port number that is used for HTTPS connections.

You can use a value of `-1` to disable the port.

The value for this property is an integer value. The default value is `9443`.

### **ltpaCookieName**

This configuration property is used to specify the name of the LTPA token cookie name.

By default, the value of this property is `LtpaToken2_${env.MQWEB_LTPA_SUFFIX}` on AIX, Linux, and Windows, or `LtpaToken2_${httpsPort}` on z/OS. The variable after the `LtpaToken2_` prefix is used by the mqweb server to generate a unique name for the cookie. You cannot set this variable, but you can change the `ltpaCookieName` to a value of your choosing.

The value for this property is a string value.

### **maxMsgTraceFiles**

This configuration property is used to specify the maximum number of messaging trace files that are generated by the mqweb server for the IBM MQ Console.

The value for this property is an integer value. The default value is `5`.

### **maxMsgTraceFileSize**

This configuration property is used to specify the maximum size, in MB, that each messaging trace file can reach.

This property applies only to the IBM MQ Console.

The value for this property is an integer value. The default value is `20`.

### **mqConsoleAutostart**

This configuration property is used to specify whether the IBM MQ Console automatically starts when the mqweb server starts.

The value for this property is a Boolean value. The default value is true.

### **mqConsoleFrameAncestors**

This configuration property is used to specify the list of origins of web pages that can embed the IBM MQ Console in an IFrame. For more information about this property, see [embedding the IBM MQ Console in an IFrame](#).

The value for this property is a string.

#### **V 9.3.0 mqConsoleRemoteSupportEnabled**

This configuration property is used to specify whether the IBM MQ Console allows remote queue manager connections. When this property is set to true, remote queue manager connections are allowed.

The value for this property is a Boolean value. The default value is true.

**V 9.3.5** This property is not valid in a stand-alone IBM MQ Web Server installation. The IBM MQ Console can be used only with remote queue managers in a stand-alone IBM MQ Web Server installation.

#### **V 9.3.0 mqConsoleRemoteAllowLocal**

This configuration property is used to specify whether remote and local queue managers are visible in the IBM MQ Console when remote queue manager connections are allowed. When this property is set to true, both local and remote queue managers are displayed.

The value for this property is a Boolean. The default value is true.

**V 9.3.5** This property is not valid in a stand-alone IBM MQ Web Server installation. The IBM MQ Console can be used only with remote queue managers in a stand-alone IBM MQ Web Server installation.

#### **V 9.3.0 mqConsoleRemotePollTime**

This configuration property is used to specify the time, in seconds, before the remote queue manager connections list is refreshed. On refresh, unsuccessful connections are retried.

The value for this property is an integer. The default value is 300.

#### **V 9.3.0 mqConsoleRemoteUIAdmin**

This configuration property is used to specify whether remote queue managers can be added to the IBM MQ Console by using the Console, or if remote queue managers can be added only by using the **setmqweb remote** command. When this property is set to true, remote queue managers can be added by using the IBM MQ Console.

The value for this property is a Boolean. The default value is false.

### **mqRestAutostart**

This configuration property is used to specify whether the REST API automatically starts when the mqweb server starts.

The value for this property is a Boolean value. The default value is true.

#### **V 9.3.3 mqRestMessagingConnectionMode**

This configuration property is used to specify whether the messaging REST API can send messages to queue managers that are not in the same installation as the mqweb server.

The value can be one of the following values:

##### **local**

The messaging REST API can send messages only to queue managers that are in the same installation as the mqweb server.

## remote

The messaging REST API can send messages to any queue manager that is configured for use by the messaging REST API. If the queue manager is in the same installation as the mqweb server, no configuration is required. For all other queue managers, a remote queue manager definition must exist. For more information about creating a remote queue manager definition to use with the messaging REST API, see [Setting up a remote queue manager to use with the messaging REST API](#).

The value is a string value. The default value is local.

The mqweb server must be restarted after this value is set.

**V9.3.5** This property is not valid in a stand-alone IBM MQ Web Server installation. The messaging REST API can be used only with remote queue managers in a stand-alone IBM MQ Web Server installation.

## **V9.3.0** remoteKeyfile

This configuration property is used to specify the location of the key file that contains the initial encryption key that is used to decrypt the passwords that are stored in the remote queue manager connection information.

The initial key is a file that must contain a single line of at least one character. However, you should use a key that is at least 16 characters. For example, your initial key file might contain the following encryption key:

```
Th1sIs@n3Ncrypt|onK$y
```

Ensure that your key file is adequately protected by using the operating system permissions, and that the encryption key is unique to the key file.

If you do not provide a key file, a default key is used.

You can also provide the path to the key file by using the **MQS\_WEBUI\_REMOTE\_KEYFILE** environment variable.

The key file that is provided here must match the same key file that is used to encrypt the password using the **-sf** parameter.

The mqweb server must be restarted after this value is set.

## secureLtpa

This configuration property is used to specify whether the LTPA token is secured for all requests. An unsecured LTPA token is required in order to send HTTP requests from a browser.

The value for this property is a Boolean value. The default value is true.

## ALW

The following values are the additional valid values for *name* on AIX, Linux, and Windows:

## managementMode

This configuration property is used to specify whether queue managers and listeners are able to be created, deleted, started, and stopped by the IBM MQ Console.

The value for this property is a string value and can be one of the following values:

### standard

Queue managers and listeners can be created and administered in the IBM MQ Console.

### externallyprovisioned

Queue managers and listeners cannot be created in the IBM MQ Console. Only queue managers and listeners that are created outside of the IBM MQ Console can be administered.

The default value is standard.

## -d

Deletes the specified configuration property from the mqwebuser.xml file.

### **-v value**

The value of the configuration property to add to, or update in, the `mqwebuser.xml` file. Any existing configuration properties of the same *name* are overwritten. Duplicate configuration properties are removed.

The value is case-sensitive. To specify an asterisk, multiple tokens, or an empty value, enclose the value in double quotation marks.

The *value* that is specified is not validated. If incorrect values are specified a subsequent attempt to start the mqweb server might fail.

**Note:** The value that is provided for a configuration property is converted into a Java Object, and some heuristic parsing is applied:

#### **Numbers**

If the value is numeric, it is parsed as a Java Number object, such as Integer or Double. A prefix of `0` indicates an octal value, `0x` a hexadecimal one, and so on. For example, `0101` becomes an Integer with the decimal value 65.

#### **Booleans**

If the value matches `true` or `false`, it is parsed as a Boolean object.

#### **Quoted values**

If the value is enclosed in double quotation marks, it is parsed as a String object. If a single character is enclosed in single quotation marks, it is parsed as a Character object.

#### **Other values**

If none of the previous rules apply, then the value is parsed without change as a String object.

These rules are important when you provide string values. If such a value can be interpreted as a number or Boolean then you must ensure that it is specified to the `setmqweb` command in double quotation marks. For example, if you give a queue manager a numeric name or call it `TRUE`, you must enclose the name in double quotation marks.

You must escape double quotation marks on the command line. For example, you might specify

```
setmqweb properties -k mqRestGatewayQmgr - v "\"0101\""
```

to set a gateway queue manager name that resembles a number.

### **-l**

Enable verbose logging. Diagnostic information is written to an mqweb server log file.

## **Return codes**

*Table 113. Return code identifiers and descriptions*

<b>Return code</b>	<b>Description</b>
0	Command successful.
>0	Command not successful.

For a full list of server command exit codes, see [Liberty:server command options](#) in the WebSphere Application Server documentation.

## **Related commands**

*Table 114. Related commands and descriptions*

<b>Command</b>	<b>Description</b>
<a href="#">“setmqweb pid (set mqweb server product ID)”</a> on page 220	Configure the product ID that the mqweb server runs under on z/OS

Table 114. Related commands and descriptions (continued)

Command	Description
<a href="#">“setmqweb remote (set mqweb server remote queue manager configuration)”</a> on page 230	Configure the mqweb server remote queue manager connections.
<a href="#">strmqweb</a>	Start the mqweb server.
<a href="#">endmqweb</a>	Stop the mqweb server.
<a href="#">dspmqweb status</a>	Display the status of the mqweb server.
<a href="#">dspmqweb properties</a>	Display the mqweb server configuration properties.
<a href="#">dspmqweb remote</a>	Display the mqweb server remote queue manager configuration.

### **setmqweb remote (set mqweb server remote queue manager configuration)**

Configure remote queue manager connection information for the mqweb server. The remote queue manager connection information is used for the IBM MQ Console, and the messaging REST API to connect to remote queue managers.

#### Purpose

You can use the **setmqweb remote** command to set up remote queue manager connections to use with the IBM MQ Console and the messaging REST API.

#### Using the command on z/OS



Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the WLP\_USER\_DIR environment variable so that the variable points to your mqweb server configuration.

To set the WLP\_USER\_DIR environment variable, enter the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP\_user\_directory* is the name of the directory that is passed to **crtmqweb**. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

You must also set the JAVA\_HOME environment variable to reference a 64-bit version of Java on your system.

The user ID running the command needs write access to the following directories:

- *WLP\_user\_directory* and its subdirectories.
- /tmp or to another directory that is referenced by the *TMPDIR* variable. If you do not have access to /tmp, the command fails with message FSUMF315 Cannot define temporary file. If you need to set the *TMPDIR* variable, issue the following command in the z/OS UNIX shell: `export TMPDIR=user_directory`

#### Using the command in a stand-alone IBM MQ Web Server installation



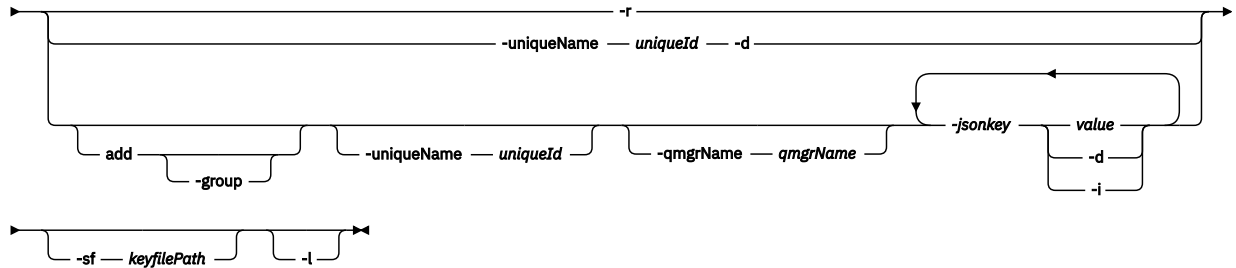
Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ\_OVERRIDE\_DATA\_PATH** environment variable to the IBM MQ Web Server data directory.

The user ID running the command needs write access to the data directory and its subdirectories.

## Syntax

ULW z/OS V 9.3.0

► setmqweb — remote ►



## Parameters

ULW z/OS V 9.3.0

### add

Add an entry to the remote queue manager connection information instead of editing an existing entry.

### V 9.3.3 -group

Whether this remote queue manager connection is part of a queue manager group.

This option can be used only when adding a new remote queue manager connection, and is valid only for the messaging REST API.

### -uniqueName *uniqueID*

A unique name for the remote queue manager connection. This is the name that appears in the IBM MQ Console.

Use a name that makes it clear that the queue manager is remote. For example, if you are specifying a unique name for remote connections to queue manager QM2, you could specify the remote name "remote-QM2".

The unique name must be specified in the following cases:

- To add a new remote queue manager connection when a queue manager with the same name might already exist in the remote queue manager connection information. In this case you must also specify the **-qmgrName** parameter to specify the queue manager that you are creating a unique name for.
- To delete a remote queue manager connection that has a unique name.
- To modify a remote queue manager connection that has a unique name.

### -qmgrName *qmgrName*

The name of the queue manager to add or update.

This parameter must be specified to add a new remote queue manager connection. The name appears as *remote\_qmgrName* in the IBM MQ Console.

### -sf *keyfilePath*

The location of the key file that contains the initial encryption key that is used to encrypt the passwords that are stored in the remote queue manager connection information.

The initial key is a file that must contain a single line of at least one character. However, you should use a key that is at least 16 characters. For example, your initial key file might contain the following encryption key:

```
Th1sIs@n3Ncrypt|onK$y
```

Ensure that your key file is adequately protected by using the operating system permissions, and that the encryption key is unique to the key file.

If you do not provide a key file, a default key is used.

You can also provide the path to the key file by using the **MQS\_REMOTE\_KEYFILE** environment variable.

### **-jsonkey value|-d|-i**

#### **jsonkey**

The name of the property to add, update, or remove. To add or update a value, specify the value after the *jsonkey* property. To delete a value, specify the **-d** flag after the *jsonkey* property.

You can add, update, or remove two types of properties. The first type are global properties that you can set with the **setmqweb remote** command without specifying a queue manager name or unique name. The second type are properties that are specific to a single remote queue manager connection. These properties can be set with the **setmqweb remote** command only if you also specify a queue manager name, a unique name, or both.

The following values are valid *jsonkey* values that you must specify without a queue manager name or unique name in the **setmqweb remote** command:

#### **globalTrustStorePath**

The path to the truststore JKS file. This truststore is used for all remote connections unless it is overridden by specific remote queue manager connection information in the **trustStorePath** entry.

The value for this *jsonkey* is a string value.

#### **globalTrustStorePassword**

The password for the global truststore.

The value for this *jsonkey* is a string value, and it is encrypted in the remote queue manager connection information.

#### **globalKeyStorePath**

The path to the keystore JKS file. This keystore is used for all remote connections unless it is overridden by a specific remote queue manager connection information in the **keyStorePath** entry.

#### **globalKeyStorePassword**

The password for the global keystore.

The value for this *jsonkey* is a string value, and it is encrypted in the remote queue manager connection information.

The following values are valid *jsonkey* values that require you to specify a queue manager name or unique name in the **setmqweb remote** command:

#### **ccdtURL**

The path to the CCDT file that is associated with the remote queue manager.

The value for this *jsonkey* is a string value.

#### **username**

The username that is used for the remote queue manager connection.

The value for this *jsonkey* is a string value.

#### **password**

The password that is associated with the username that is used for the remote queue manager connection.



The value for this jsonkey is a string value, and it is encrypted in the remote queue manager connection information.

**enableMutualTLS**

Whether this remote queue manager connection adds a keystore to enable mutual TLS.

The value for this jsonkey is a Boolean value.

**keyStorePath**

The path to the keystore JKS file.

The value for this jsonkey is a string value, and it overrides the global keystore value.

**keyStorePassword**

The password for the keystore file.

The value for this jsonkey is a string value, and it is encrypted in the remote queue manager connection information.

**trustStorePath**

The path to the truststore JKS file.

The value for this jsonkey is a string value, and it overrides the global truststore value.

**trustStorePassword**

The password for the truststore file.

The value for this jsonkey is a string value, and it is encrypted in the remote queue manager connection information.

**V9.3.3 visibility**

Whether this remote queue manager connection can be used by the messaging REST API, the IBM MQ Console, or both.

The value for this jsonkey can be one of the following values:

**messaging**

The queue manager connection can be used only by the messaging REST API.

**console**

The queue manager connection can be used only by the IBM MQ Console.

**messaging,console**

The queue manager connection can be used by either the messaging REST API or the IBM MQ Console.

The value for this jsonkey is a string value. The default value is **messaging,console**.

**value**

The value of the JSON key entry to add or update.

The values are case-sensitive and must be enclosed in double quotation marks.

**-d**

Delete the specified property from the remote connection information.

**-i**

Enable interactive mode for the specified JSON key entry. You are then prompted for the JSON key value as the command runs.

**-d**

Delete the connection information for the queue manager with the specified unique name.

**-r**

Reset and remove all remote connection information.

**-l**

Enable verbose logging. Diagnostic information is written to an mqweb server log file.

## Return codes

Table 115. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

For a full list of server command exit codes, see [Liberty:server command options](#) in the WebSphere Application Server documentation.

## Examples

### V9.3.0

The following example sets the path to the global keystore for remote queue manager connections:

```
setmqweb remote -globalTrustStorePath "c:\supersecure\keys.jks"
```

The following example creates a new entry for a queue manager, QM2, in the remote queue manager connection information. The example sets the CCDT URL, a username and password to use with the connection, and a keystore path:

```
setmqweb remote add -qmgrName "QM2" -ccdtURL "c:\myccdt\cdt.json" -username "user" -password "password" -keyStorePath "c:\supersecure\keys.jks"
```

The following example creates a new entry for a different queue manager that is also named QM2, and specifies a unique name to differentiate between the two QM2 queue managers. The example sets the CCDT URL, a username, and a password. The example uses the **-i** option to interactively enter the password that is associated with the username when the command runs:

```
setmqweb remote add -uniqueName qm2remote -qmgrName "QM2" -ccdtURL "c:\myccdt\cdt.json" -username "mqadmin" -password -i
```

## Related commands

Table 116. Related commands and descriptions

Command	Description
<a href="#">setmqweb properties</a> ( <a href="#">set mqweb server configuration properties</a> )” on page <a href="#">222</a>	Configure the mqweb server properties.
<a href="#">setmqweb pid</a> ( <a href="#">set mqweb server product ID</a> )” on page <a href="#">220</a>	Configure the product ID that the mqweb server runs under on z/OS
<a href="#">strmqweb</a>	Start the mqweb server.
<a href="#">endmqweb</a>	Stop the mqweb server.
<a href="#">dspmqweb status</a>	Display the status of the mqweb server.
<a href="#">dspmqweb properties</a>	Display the mqweb server configuration properties.
<a href="#">dspmqweb remote</a>	Display the mqweb server remote queue manager configuration.

## setmqxcred (add XA credentials)

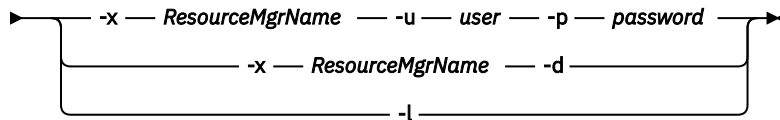
Use the **setmqxcred** command to add or modify credentials in the IBM MQ XA credentials store.

### Purpose

The **setmqxcred** command adds new credentials to the IBM MQ XA credentials store, or modifies or deletes existing credentials.

### Syntax

►► setmqxcred — -m — ? — QmgrName ►►



### Required parameters

#### -m QmgrName

The queue manager for which authentication details are stored.

### Optional parameters

#### -x ResourceMgrName

Specifies the resource manager name as defined in the `qm.ini` file.

#### -u user

Specifies the user name to use to connect to the database.

#### -p password

Specifies the password for the user.

#### -d

Deletes the credentials for the named resource manager.

#### -l

Lists the credentials in the queue manager store.

### Examples

To add credentials for the queue manager QM1 for the resource mqdb2:

```
# setmqxcred -m QM1 -x mydb2 -u user1 -p Password1
Successfully added credentials for XA Resource Manager mydb2
```

To delete the credentials for the queue manager QM1 for the resource mqdb2:

```
# setmqxcred -m QM1 -x mydb2 -d
Successfully removed credentials for XA Resource Manager mydb2
```

To list details about the credentials stored in the credentials store.

```
# setmqxcred -m QM1 -l
ResourceName(mydb2) UserName(user1)
ResourceName(myora) UserName(user2)
```

## Multi **strmqcsv (start command server)**

Start the command server for a queue manager.

### Purpose

Use the **strmqcsv** command to start the command server for the specified queue manager. This enables IBM MQ to process commands sent to the command queue.

You must use the **strmqcsv** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmqr -o installation` command.

If the queue manager attribute, SCMDSERV, is specified as QMGR then changing the state of the command server using **strmqcsv** does not effect how the queue manager acts upon the SCMDSERV attribute at the next restart.

### Syntax



### Required parameters

None

### Optional parameters

#### -a

Blocks the following PCF commands from modifying or displaying authority information:

- Inquire authority records (MQCMD\_INQUIRE\_AUTH\_RECS)
- Inquire entity authority (MQCMD\_INQUIRE\_ENTITY\_AUTH)
- Set authority record (MQCMD\_SET\_AUTH\_REC).
- Delete authority record (MQCMD\_DELETE\_AUTH\_REC).

#### QMgrName

The name of the queue manager on which to start the command server. If omitted, the default queue manager is used.

### Return codes

Table 117. Return code identifiers and descriptions

Return code	Description
0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

### Examples

The following command starts a command server for queue manager earth:

```
strmqcsv earth
```

## Related commands

Table 118. Related command names and descriptions

Command	Description
<a href="#">endmqcsv</a>	End a command server
<a href="#">dspmqcsv</a>	Display the status of a command server

### Related reference

[“Command server commands” on page 7](#)

A table of command server commands, showing equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

## **strmqsvc (start IBM MQ service)**

Start the IBM MQ service on Windows.

### Purpose

The command starts the IBM MQ service on Windows. Run the command on Windows only.

If you are running IBM MQ on Windows systems with User Account Control (UAC) enabled, you must invoke **strmqsvc** with elevated privileges.

Run the command to start the service, if it has not been started automatically, or if the service has ended.

Restart the service for IBM MQ processes to pick up a new environment, including new security definitions.

### Syntax

```
strmqsvc
```

### Parameters

The **strmqsvc** command has no parameters.

You must set the path to the installation that contains the service. Either make the installation primary, run the **setmqenv** command, or run the command from the directory containing the **strmqsvc** binary file.

### Related reference

[“endmqsvc \(end IBM MQ service\)” on page 127](#)

End the IBM MQ service on Windows.

## **strmqm (start queue manager)**

Start a queue manager or ready it for standby operation.

### Purpose

Use the **strmqm** command to start a queue manager.

You must use the **strmqm** command from the installation that is associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmqr -o installation` command.

If a queue manager has no associated installation and there is no installation of IBM MQ on the system, the **strmqm** command associates the queue manager with the installation that issued the **strmqm** command.

If the queue manager startup takes more than a few seconds IBM MQ shows intermittent messages detailing the startup progress.

## Usage notes

From IBM MQ 9.1, IBM MQ supports the use of back-up queue managers. That is, a queue manager where log extents are asynchronously copied to a backup machine, and where replay of the log records is periodically driven by use of the command **strmqm -r**. When the backup queue manager needs to be activated, use the command **strmqm -a** and then start the queue manager normally.



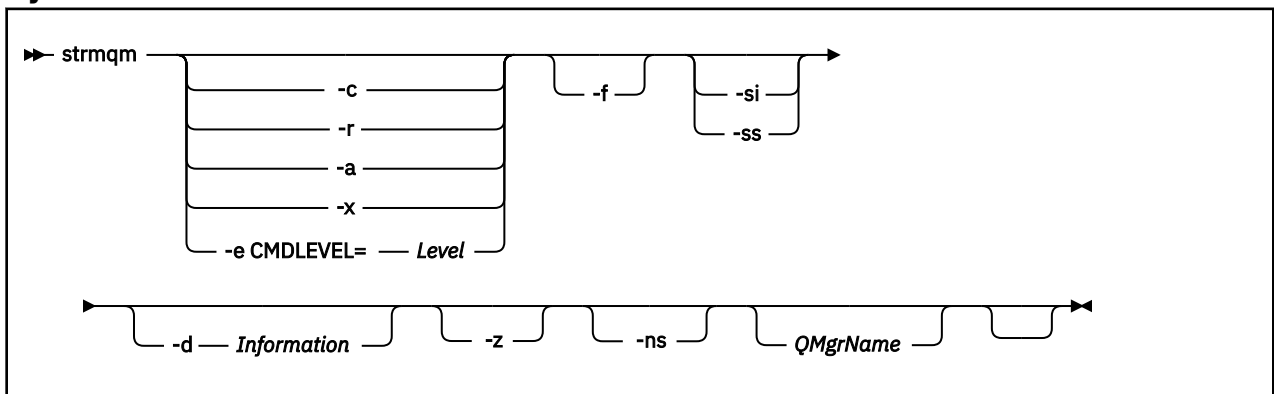
**Attention:** You cannot use **LogManagement=Automatic**, along with a backup queue manager, as extents might be reused before they are backed up. Furthermore, if you run the command **strmqm -r** together with **LogManagement=Automatic**, the command fails.

**UNIX** From IBM MQ 9.1, the security of `data path/log/qm` on UNIX systems is changed from 2775 to 2770.

The **strmqm** command checks the syntax of the CHANNELS and SSL stanzas in the `qm.ini` file early on, before starting the queue manager fully. If the `qm.ini` file contains any errors, this check makes it much easier to see what is wrong, and correct it quickly. If an error is found, **strmqm** outputs an AMQ9224 error message, describing the full details of the position of the error in the `qm.ini` file. It also ends immediately without starting the queue manager.

**Linux** From IBM MQ 9.2.0, you can use the environment variable `MQLICENSE` to accept or view the license.

## Syntax



## Optional parameters

**-a**

Activate the specified backup queue manager. The backup queue manager is not started.

When activated, a backup queue manager can be started by using the control command `strmqm QMgrName`. The requirement to activate a backup queue manager prevents accidental startup.

When activated, a backup queue manager can no longer be updated.

For more information about using backup queue managers, see [Backing up and restoring IBM MQ queue manager data](#).

**-c**

Starts the queue manager, redefines the default and system objects, then stops the queue manager. Any existing system and default objects that belong to the queue manager are replaced if you specify this flag, and any non-default system object values are reset (for example, the value of MCAUSER is set to blank).

Use the **strmqm** command to create the default and system objects for a queue manager.

**Note:** If you run **strmqm -c** on a queue manager that is being used as a Managed File Transfer coordination queue manager you must re-run the MQSC script that defines the coordination queue manager objects. This script is in a file called *queue\_manager\_name.mqsc*, which is in the Managed File Transfer configuration directory.

The **-c** option is not available for Native HA queue managers.

**-d Information**

Specifies whether information messages are displayed. Possible values for *Information* are as follows:

Value	Description
all	All information messages are displayed. This value is the default value.
minimal	The minimal number of information messages are displayed.
none	No information messages are displayed. This parameter is equivalent to <b>-z</b> .

The **-z** parameter takes precedence over this parameter.

**-e CMDLEVEL = Level**

Enables a command level for this queue manager, and then stops the queue manager.

The queue manager is now able to use all functions that are provided by the specified command level. You can start the queue manager only with an installation that supports the new command level.

This option is only valid if the current command level that is used by the queue manager is lower than the maximum command level supported by the installation. Specify a command level that is greater than the current command level of the queue manager and less than or equal to the maximum command level supported by the installation.

Use exactly the command level as a value for *Level* that is associated with the function you want to enable.

This flag cannot be specified with **-a**, **-c**, **-r** or **-x**.

**-f**

Use this option if you know that a queue manager is not starting because its data directories are missing or corrupted.

The **strmqm -f qmname** command attempts to re-create the queue manager data directory and reset file permissions. If it is successful, the queue manager starts, unless the queue manager configuration information is missing. If the queue manager fails to start because the configuration information is missing, re-create the configuration information, and restart the queue manager.

The default behavior of **strmqm**, with no **-f** option, is not to recover missing or corrupted data directories automatically, but to report an error, such as AMQ6235 or AMQ7001, and not start the queue manager.

You can think of the **-f** option as performing the recover actions that used to be performed automatically by **strmqm**.

You must not use **strmqm -f** to re-create the queue manager data directories if you can restore the directories by correcting the configuration.

Possible solutions to problems with **strmqm** are to make the networked file storage location accessible to the queue manager, or to ensure the gid and uid of the mqm group and user ID on the server hosting the queue manager match the gid and uid of the mqm group and user ID on the server that is hosting the queue manager data directory.

If you are performing media recovery for a queue manager, then you must use the **-f** option to re-create the queue manager data directory.

#### **-ns**

Prevents any of the following processes from starting automatically when the queue manager starts:

- The channel initiator
- The command server
- Listeners
- Services

This parameter also runs the queue manager as if the CONNAUTH attribute is blank, regardless of its current value. Client applications cannot connect because there are no listeners. Authorization of applications and control commands will occur based on the local OS user under which you run them. If the queue manager had previously used LDAP users/groups for its authorization records, then:

1. These records will be ignored while the queue manager is running in **-ns** mode.
2. You should not make changes to authorization records or create new objects while in this mode, because the authorization records that are created or amended in this mode will then contain user names derived from the operating system, not the LDAP repository.

Administrative changes must be made by using **runmqsc** because the command server is not running.


To re-enable the normal authorization service processing, that is, return the effective CONNAUTH value to its normal setting, you must end and start the queue manager without the **-ns** parameter.

#### **-r**

Updates the backup queue manager. The backup queue manager is not started.

IBM MQ updates the objects of the backup queue manager by reading the queue manager log and replaying updates to the object files.

For more information about using backup queue managers, see [Backing up and restoring IBM MQ queue manager data](#).

**Note:**  The **-r** option is not available for Native HA queue managers.

#### **Windows -si**

Interactive (manual) queue manager startup type. This option is available on IBM MQ for Windows only.

The queue manager runs under the logged on (interactive) user. Queue managers that are configured with interactive startup end when the user who started them logs off.

If you set this parameter, it overrides any startup type set previously by the **crtmqm** command, the **amqmdain** command, or the IBM MQ Explorer.

If you do not specify a startup type of either **-si** or **-ss**, the queue manager startup type that is specified on the **crtmqm** command is used.

#### **Windows -ss**

Service (manual) queue manager startup type. This option is available on IBM MQ for Windows only.

The queue manager runs as a service. Queue managers that are configured with service startup continue to run even after the interactive user is logged off.

If you set this parameter, it overrides any startup type set previously by the **crtmqm** command, the **amqmdain** command, or the IBM MQ Explorer.



**-x**

Start an instance of a multi-instance queue manager on the local server, permitting it to be highly available. If an instance of the queue manager is not already running elsewhere, the queue manager starts and the instance becomes active. The active instance is ready to accept local and remote connections to the queue manager on the local server.

If a multi-instance queue manager instance is already active on a different server the new instance becomes a standby, permitting it to takeover from the active queue manager instance. While it is in standby, it cannot accept local or remote connections.

You must not start a second instance of a queue manager on the same server.

The default behavior, omitting the -x optional parameter, is to start the instance as a single instance queue manager, forbidding standby instances from being started.

**-z**

Suppresses error messages.

This flag is used within IBM MQ to suppress unwanted information messages. Because using this flag can result in loss of information, do not use it when you are entering commands on a command line.

This parameter takes precedence over the -d parameter.

**QMgrName**

The name of a local queue manager. If omitted, the default queue manager is used.

**Return codes**

*Table 119. Return code identifiers and descriptions*

<b>Return code</b>	<b>Description</b>
0	Queue manager started.
1	The location chosen for the queue manager data directory is invalid
3	Queue manager being created.
5	Queue manager running.
16	Queue manager does not exist.
23	Log not available.
24	A process that was using the previous instance of the queue manager has not yet disconnected.
30	A standby instance of the queue manager started. The active instance is running elsewhere.
31	The queue manager already has an active instance. The queue manager permits standby instances.
39	Invalid parameter specified.
43	The queue manager already has an active instance. The queue manager does not permit standby instances.
47	The queue manager already has the maximum number of standby instances.
49	Queue manager stopping.
58	Inconsistent use of installations detected.
62	The queue manager is associated with a different installation.
69	Storage not available.
71	Unexpected error.
72	Queue manager name error.

Table 119. Return code identifiers and descriptions (continued)

Return code	Description
74	The IBM MQ service is not started.
91	The command level is outside the range of acceptable values.
92	The queue manager's command level is greater or equal to the specified value.
94	A replica instance of the queue manager has been started.
100	Log location invalid.
114	Invalid qm.ini file stanza.
119	User not authorized to start the queue manager.

### Examples

The following command starts the queue manager account:

```
strmqm account
```

### Related tasks

[Applying maintenance level updates to multi-instance queue managers on AIX](#)

[Applying maintenance level updates to multi-instance queue managers on Linux](#)

[Applying maintenance level updates to multi-instance queue managers on Windows](#)

### Related reference

[crtmqm \(create queue manager\)](#)

Create a queue manager.

[dlmqm \(delete queue manager\)](#)

Delete a queue manager.

[dspmqver \(display IBM MQ version information\)](#)

Display IBM MQ version and build information.

[endmqm \(end queue manager\)](#)

Stop a queue manager or switch to a standby queue manager or to a replica queue manager.

[“amqmdain \(services control\)” on page 23](#)

**amqmdain** is used to configure or control some Windows specific administrative tasks.

[“strmqsvc \(start IBM MQ service\)” on page 237](#)

Start the IBM MQ service on Windows.

[“endmqsvc \(end IBM MQ service\)” on page 127](#)

End the IBM MQ service on Windows.

### **strmqtrc (start trace)**

Enable trace at a specified level of detail, or report the level of tracing in effect.

### Purpose

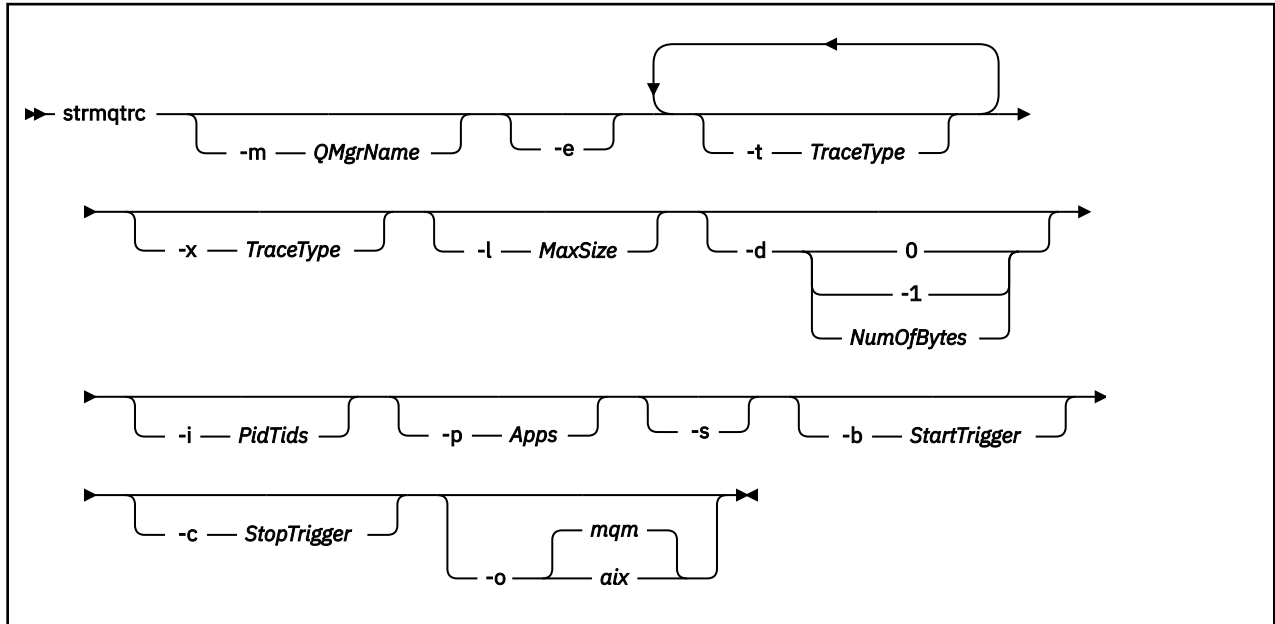
Use the **strmqtrc** command to enable tracing.

You must use the **strmqtrc** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with by using the **dspmq** command as follows:

```
dspmq -o installation
```

## Syntax

The syntax of this command is as follows:



## Description

The **strmqtrc** command enables tracing. The command has optional parameters that specify the level of tracing you want:

- One or more queue managers
- Levels of trace detail
- One or more IBM MQ processes. The processes can be either part of the IBM MQ product or customer applications that use the IBM MQ API
- Specific threads within customer applications, either by IBM MQ thread number or by operating system thread number
- Events. These can be either the entry or exit from internal IBM MQ functions or the occurrence of a first failure data capture (FDC).

Each combination of parameters on an individual invocation of the command are interpreted by IBM MQ as having a logical AND between them. You can start the **strmqtrc** command multiple times, regardless of whether tracing is already enabled. If tracing is already enabled, the trace options that are in effect are modified to those specified on the most recent invocation of the command. Multiple invocations of the command, without an intervening **enmqtrc** command, are interpreted by IBM MQ as having a logical OR between them. The maximum number of concurrent **strmqtrc** commands that can be in effect at one time is 16.

## Optional parameters

### **-m** *QMgrName*

The name of the queue manager to trace.

The following wildcards are allowed: asterisk (\*), replacing zero or more characters, and question mark (?), replacing any single character. In command environments such as the UNIX shell, where the asterisk (\*) and question mark (?) characters have special meaning, you must either escape the wildcard character or enclose it in quotation marks to prevent the command environment from operating on the wildcard character.

**-e**

Requests early tracing of all processes, making it possible to trace the creation or startup of a queue manager. If you include this parameter, any process belonging to any component of any queue manager traces its early processing. The default is not to perform early tracing.

Use the following command to trace a client:

```
strmqtrc -e
```

You cannot use the **-e** parameter with the **-m** parameter, **-i** parameter, the **-p** parameter, the **-c** parameter, or the **-b** parameter. If you try to use the **-e** parameter with the **-m** parameter, the **-i** parameter, the **-p** parameter, the **-c** parameter, or the **-b** parameter, then an error message is issued.

**-t TraceType**

The points to trace and the amount of trace detail to record. By default **all** trace points are enabled and a default-detail trace is generated.

Alternatively, you can supply one or more of the options in the following list. For each *Tracetype* value you specify, including **-t all**, specify either **-t parms** or **-t detail** to obtain the appropriate level of trace detail. If you do not specify either **-t parms** or **-t detail** for any particular trace type, only a default-detail trace is generated for that trace type.



**Attention:** When using the **-t api** option, you will see trace of the MQI calls, with all the input and output data blocks dumped in hexadecimal form.

You should be aware that IBM MQ internal programs also make MQI calls, and you will see trace files for those programs. Normally, the program names begin **amq** or **runmq**.

You should be aware that **amqrmppa** programs host many threads, some of which receive MQI calls over the network from client applications. In these threads you will see MQI calls in the **-t api** traces, but you must remember that the input arguments to those MQI calls traced in the **amqrmppa** program might not match every detail of the MQI calls made originally by the client.

Therefore, if you need to know, reliably, the input arguments to MQI calls made by the client application, you must use **-t api** tracing on the client machine directly.

If you supply multiple trace types, each must have its own **-t** parameter. You can include any number of **-t** parameters, if each has a valid trace type associated with it.

It is not an error to specify the same trace type on multiple **-t** parameters.

Value	Description
all	Output data for every trace point in the system (the default). The all parameter activates tracing at default detail level.
amqp	Output data for the AMQP service
api	Output data for trace points associated with the MQI and major queue manager components.
commentary	Output data for trace points associated with comments in the IBM MQ components.
comms	Output data for trace points associated with data flowing over communications networks.

<i>Table 120. TraceType parameter values. (continued)</i>	
<b>Value</b>	<b>Description</b>
csdata	Output data for trace points associated with internal data buffers in common services.
csflows	Output data for trace points associated with processing flow in common services.
detail	Activate tracing at high-detail level for flow processing trace points.
explorer	Output data for trace points associated with the IBM MQ Explorer.
java	Output data for trace points associated with applications using the IBM MQ classes for Java API.
lqmdata	Output data for trace points associated with internal data buffers in the local queue manager.
lqmflows	Output data for trace points associated with processing flow in the local queue manager.
mqxr	Output data for the Telemetry (MQXR) service.
otherdata	Output data for trace points associated with internal data buffers in other components.
otherflows	Output data for trace points associated with processing flow in other components.
parms	Activate tracing at default-detail level for flow processing trace points.
remotedata	Output data for trace points associated with internal data buffers in the communications component.
remoteflows	Output data for trace points associated with processing flow in the communications component.
servicedata	Output data for trace points associated with internal data buffers in the service component.
serviceflows	Output data for trace points associated with processing flow in the service component.
spldata	Output data for trace points associated with buffers and control blocks that use a security policy (AMS) operation.
splflows	Output data for trace points associated with entry and exit data for functions that use a security policy (AMS) operation.
ssl	Output data associated with using GSKit to enable TLS channel security.
versiondata	Output data for trace points associated with the version of IBM MQ running.

**-x TraceType**

The points **not** to trace. By default **all** trace points are enabled and a default-detail trace is generated. The trace points you can specify are those listed for the **-t** parameter.

You can use the **-x** parameter with *Tracetype* values to exclude those entry points you do not want to record. This is useful in reducing the amount of trace produced.

If you supply multiple trace types, each must have its own **-x** parameter. You can include any number of **-x** parameters, if each has a valid *Tracetype* associated with it.

#### **-l MaxSize**

The maximum size of a trace file ( *AMQppppp.qq.TRC*) in megabytes (MB), where *ppppp* refers to the operating system process ID of the particular IBM MQ process being traced and *qq* is a sequence number if there is already a file with that name. For example, if you specify a *MaxSize* of 1, the size of the trace is limited to 1 MB.

When a trace file reaches the specified maximum, it is renamed to *AMQppppp.qq.TRS* and a new *AMQppppp.qq.TRC* file is started. If a previous copy of an *AMQppppp.qq.TRS* file exists, it is deleted.

The highest value that *MaxSize* can be set to is 2048 MB.

#### **-d**

Trace options. The value can be:

**0**

Trace no user data.

**-1 or all**

Trace all user data.

#### **NumOfBytes**

- For a communication trace; trace the specified number of bytes of data including the transmission segment header (TSH).
- For an MQPUT or MQGET call; trace the specified number of bytes of message data held in the message buffer.
- Values in the range 1 through 15 are not allowed.

#### **-i PidTids**

Process identifier (PID) and thread identifier (TID) to which the trace generation is restricted. You cannot use the **-i** parameter with the **-e** parameter. If you try to use the **-i** parameter with the **-e** parameter, then an error message is issued.

The precise format of this parameter is *PID[.TID]*. For example:

Coding **-i 12345** traces all threads in PID 12345, whereas

Coding **-i 12345.67** traces only thread 67 in PID 12345

This parameter is not supported for .NET clients if *NMQ\_MQ\_LIB* is set to managed, so that the client uses managed IBM MQ problem diagnostics.

#### **-p Apps**

The named processes to which the trace generation is restricted. *Apps* is a comma-separated list. You must specify each name in the list exactly as the program name would be displayed in the "Program Name" FDC header. Asterisk (\*) or question mark (?) wildcards are allowed. You cannot use the **-p** parameter with the **-e** parameter. If you try to use the **-p** parameter with the **-e** parameter, then an error message is issued.

This parameter is not supported for .NET clients if *NMQ\_MQ\_LIB* is set to managed, so that the client uses managed IBM MQ problem diagnostics.

#### **-s**

Reports the tracing options that are currently in effect. You must use this parameter on its own with no other parameters.

A limited number of slots are available for storing trace commands. When all slots are in use, then no more trace commands can be accepted unless they replace an existing slot. Slot numbers are not fixed, so if the command in slot number 0 is removed, for example by an **endmqtrc** command, then all the other slots move up, with slot 1 becoming slot 0, for example. An asterisk (\*) in a field means that no value is defined, and is equivalent to the asterisk wildcard.

An example of the output from this command is as follows:

```
Listing Trace Control Array
Used slots = 2 of 15

EarlyTrace      [OFF]
TimedTrace      [OFF]
TraceUserData   [0]
MaxSize         [0]
Trace Type      [1]

Slot position 1

Untriggered
Queue Manager   [avocet]
Application     [*]
PID.TID        [*]
TraceOptions    [1f4ffff]
TraceInterval   [0]
Trace Start Time [0]
Trace Stop Time [0]
Start Trigger   [KN346050K]
Start Trigger   [KN346080]

Slot position 2

Untriggered
Queue Manager   [*]
Application     [*]
PID.TID        [*]
TraceOptions    [1fcffff]
TraceInterval   [0]
Trace Start Time [0]
Trace Stop Time [0]
Start Trigger   [KN346050K]
Start Trigger   [KN346080]
```

This parameter is not supported for .NET clients if NMQ\_MQ\_LIB is set to managed, so that the client uses managed IBM MQ problem diagnostics.

**-b Start\_Trigger**

FDC probe IDs for which tracing must be turned on. *Start\_Trigger* is a comma-separated list of FDC probe IDs. You can use asterisk (\*) and question mark (?) wildcards in the specification of probe IDs. You cannot use the **-b** parameter with the **-e** parameter. If you try to use the **-b** parameter with the **-e** parameter, then an error message is issued. This parameter must only be used under the guidance of IBM Service personnel.

<i>Table 121. Start trigger and effect</i>	
<b>Start_Trigger</b>	<b>Effect</b>
FDC=comma-separated list of FDC probe IDs.	Turns on tracing when any FDCs with the specified FDC probe IDs are generated.

This parameter is not supported for .NET clients if NMQ\_MQ\_LIB is set to managed, so that the client uses managed IBM MQ problem diagnostics.

**-c Stop\_Trigger**

FDC probe IDs for which tracing must be turned off, or interval in seconds after which tracing must be turned off. *Stop\_Trigger* is a comma-separated list of FDC probe IDs. You can use asterisk (\*) and question mark (?) wildcards in the specification of probe IDs. This parameter should be used only under the guidance of IBM Service personnel.

<i>Table 122. Stop triggers and their effects</i>	
<b>Stop_Trigger</b>	<b>Effect</b>
FDC=comma-separated list of FDC probe IDs.	Turns tracing off when any FDCs with the specified FDC probe IDs are generated.

Table 122. Stop triggers and their effects (continued)

Stop_Trigger	Effect
interval=n where n is an unsigned integer between 1 and 32,000,000.	Turns tracing off n seconds after it starts or, if it tracing is already enabled, turns tracing off n seconds after this instance of the command is issued.

This parameter is not supported for .NET clients if NMQ\_MQ\_LIB is set to managed, so that the client uses managed IBM MQ problem diagnostics.

**-o**

**mqm**

Enables IBM MQ trace as in previous releases.

This is the default value if no -o option is supplied.

**AIX aix**

Enables IBM MQ to write AIX system trace, provided AIX system trace is enabled.

As previously, you must use the AIX operating system trace command for any output to actually be produced.

This is a legacy option, and you should use this option only when directed to do so by IBM service personnel.

**Return codes**

Table 123. Return code identifiers and descriptions

**Return code Description**

AMQ7024	Non-valid arguments supplied to the command.
AMQ7077	You are not authorized to perform the requested operation.
AMQ8304	Nine concurrent traces (the maximum) already running.
58	Inconsistent use of installations detected

**Examples of enabling tracing at different levels of detail**

**Linux AIX** This command enables tracing of processing flow from common services and the local queue manager for a queue manager called QM1 on IBM MQ for AIX or Linux systems. Trace data is generated at the default level of detail.

```
stmqtrc -m QM1 -t csflows -t lqmflows -t parms
```

This command disables tracing of TLS activity on a queue manager called QM1. Other trace data is generated at the parms level of detail.

```
stmqtrc -m QM1 -x ssl -t parms
```

This command enables high-detail tracing of the processing flow for all components:

```
stmqtrc -t all -t detail
```



## Examples of enabling tracing for an FDC

This command enables tracing when FDC KN346050 or FDC KN346080 occur on any process that is using queue manager QM1:

```
strmqtrc -m QM1 -b FDC=KN346050,KN346080
```

This command enables tracing when FDC KN34650 occurs, and stops tracing when FDC KN346080 occurs. In both cases the FDC must occur on a process that is using queue manager QM1:

```
strmqtrc -m QM1 -b FDC=KN346050 -c FDC=KN346080
```

## Examples of using the -p and -m parameters for individual and multiple invocations of strmqtrc

The following examples use the **-p** and **-m** parameters to show:

- How a combination of parameters on an individual invocation of the command are interpreted by IBM MQ as having a logical AND between them.
- How multiple invocations of the command, without an intervening enmqtrc command, are interpreted by IBM MQ as having a logical OR between them:

1. This command enables tracing for all threads that result from any executing process called amqxxx.exe:

```
strmqtrc -p amqxxx.exe
```

2. After running the **strmqtrc** command as shown in step 1, you can then enter either of the following commands without an intervening **endmqtrc** command.

- If you start the following command after the command in step 1, without an intervening **endmqtrc** command, then tracing is limited to all threads that result from any executing process called amqxxx.exe *and* that are using queue manager QM2:

```
strmqtrc -p amqxxx.exe -m QM2
```

- If you start the following command after the command in step 1, without an intervening **endmqtrc** command, then tracing is limited to all processes and threads that result from executing amqxxx.exe *or* that are using queue manager QM2:

```
strmqtrc -m QM2
```

## Example of enabling dynamic tracing of LDAP client library code shipped with IBM MQ

From IBM MQ 9.1.0 Fix Pack 4 and IBM MQ 9.1.4, it is possible to switch LDAP client trace on and off without also stopping or starting the queue manager.

You can use the following command to switch on the trace:

```
strmqtrc -m QMNAME -t servicedata
```

To enable this behavior, it is also necessary to set an environment variable `AMQ_LDAP_TRACE` to a non-null value. For more information, see [Enabling dynamic tracing of LDAP client library code](#).

## Related commands

Table 124. Related command names and descriptions

Command	Description
<code>dspmqtrc</code>	Display formatted trace output
<code>endmqtrc</code>	End trace

### Related tasks

[Using trace](#)

### Related reference

Command sets comparison: [Other commands](#)

A table of other commands, showing the command description, and the equivalent PCF commands, MQSC commands, and control commands. The REST API resource and HTTP method equivalents, and IBM MQ Explorer equivalents, are included if available.

## Multi `strmqweb` (start mqweb server)

Start the mqweb server that is used to support the IBM MQ Console and REST API.

### Purpose

Use the `strmqweb` command to start the mqweb server to use the IBM MQ Console or the REST API.

### Usage notes

**V 9.3.5** **Linux** Before you issue the `strmqweb` command in a stand-alone IBM MQ Web Server installation, you must set the `MQ_OVERRIDE_DATA_PATH` environment variable to the IBM MQ Web Server data directory.

- If the mqweb server is part of an IBM MQ installation, you must start the mqweb server as a [privileged user](#).
- **V 9.3.5** **Linux** If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must start the mqweb server as a user that has read and write access to the IBM MQ Web Server data directory.

**Linux** Before you start the mqweb server, you must accept the IBM MQ license. On Linux, you can accept the license after installation. For more information, see [“mqlicense \(accept license post installation\)”](#) on page 135.

### Syntax

```
strmqweb --clean
```

### Optional parameters

#### `--clean`

Cleans all persistent cached information that is related to the specified server instance, which includes OSGi resolver metadata and persistent OSGi bundle data. If you use this option, the server will be required to recompute any cached data at the next startup, which might take more time than a restart that can reuse cached data.

**Note:** This option is not necessary for normal operation. IBM service might request that you use this option when providing an interim fix, or if there is a suspected problem with the cached data.

## Return codes

Table 125. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

For a full list of server command exit codes, see [Liberty:server command options](#) in the WebSphere Application Server documentation.

## Related commands

Table 126. Related command names and descriptions

Command	Description
<a href="#">dspmqweb</a>	Display the status of the mqweb server.
<a href="#">endmqweb</a>	Stop the mqweb server.

## MQSC commands reference

Use MQSC commands to help you manage queue manager objects, including the queue manager itself, queues, process definitions, channels, client connection channels, listeners, services, namelists, clusters, and authentication information objects.

This section describes, in alphabetical order, all the MQSC commands that can be issued by operators and administrators.

**Note:** How you run MQSC commands varies by platform. See [Administering IBM MQ using MQSC commands](#).

[“ALTER AUTHINFO \(alter authentication information object\)” on page 254](#)

[“ALTER BUFFPOOL \(alter buffer pool settings\) on z/OS” on page 266](#)

[“ALTER CFSTRUCT \(alter CF application structure\) on z/OS” on page 268](#)

[“ALTER CHANNEL \(alter channel settings\)” on page 274](#)

[“ALTER CHANNEL \(alter channel settings\) MQTT” on page 329](#)

[“ALTER COMMINFO \(alter communication information object\) on Multiplatforms” on page 332](#)

[“ALTER LISTENER \(alter an existing listener\) on Multiplatforms” on page 336](#)

[“ALTER NAMELIST \(alter a list of names\)” on page 339](#)

[“ALTER PROCESS \(alter an existing process definition\)” on page 342](#)

[“ALTER PSID \(change page set expansion method\) on z/OS” on page 346](#)

[“ALTER QMGR \(alter queue manager settings\)” on page 348](#)

[“ALTER queues \(alter queue settings\)” on page 382](#)

[“ALTER SECURITY \(alter security options\) on z/OS” on page 413](#)

[“ALTER SERVICE \(alter a service definition\) on Multiplatforms” on page 415](#)

[“ALTER SMDS \(alter shared message data sets\) on z/OS” on page 417](#)

[“ALTER STGCLASS \(alter storage class settings\) on z/OS” on page 419](#)

[“ALTER SUB \(alter subscription settings\)” on page 421](#)

[“ALTER TOPIC \(alter topic settings\)” on page 425](#)

[“ALTER TRACE \(alter trace event settings\) on z/OS” on page 434](#)

[“ARCHIVE LOG \(back up the active log\) on z/OS” on page 436](#)

[“BACKUP CFSTRUCT \(back up a CF application structure\) on z/OS” on page 438](#)

[“CLEAR QLOCAL \(clear messages from local queue\)” on page 440](#)

[“CLEAR TOPICSTR \(clear topic string\)” on page 441](#)

[“DEFINE AUTHINFO \(define an authentication information object\)” on page 443](#)

[“DEFINE BUFFPOOL \(define a buffer pool\) on z/OS” on page 457](#)

[“DEFINE CFSTRUCT \(define coupling facility application structure\) on z/OS” on page 459](#)

[“DEFINE CHANNEL \(define a new channel\)” on page 466](#)

[“DEFINE CHANNEL \(define a new channel\) for MQTT” on page 523](#)

[“DEFINE COMMINFO \(define a new communication information object\) on Multiplatforms” on page 526](#)

[“DEFINE LISTENER \(define a new listener\) on Multiplatforms” on page 530](#)

[“DEFINE LOG \(define a new active log\) on z/OS” on page 534](#)

[“DEFINE MAXSMSGS \(define maximum messages setting\) on z/OS” on page 535](#)

[“DEFINE NAMELIST \(define a list of names\)” on page 536](#)

[“DEFINE PROCESS \(create a new process definition\)” on page 540](#)

[“DEFINE PSID \(define page set and buffer pool\) on z/OS” on page 545](#)

[“DEFINE queues” on page 547](#)

[“DEFINE SERVICE \(create a new service definition\) on Multiplatforms” on page 581](#)

[“DEFINE STGCLASS \(define storage class to page set mapping\) on z/OS” on page 585](#)

[“DEFINE SUB \(create a durable subscription\)” on page 588](#)

[“DEFINE TOPIC \(define a new administrative topic\)” on page 595](#)

[“DELETE AUTHINFO \(delete authentication information\)” on page 605](#)

[“DELETE AUTHREC \(delete authority records\) on Multiplatforms” on page 607](#)

[“DELETE BUFFPOOL \(delete a buffer pool\) on z/OS” on page 609](#)

[“DELETE CFSTRUCT \(delete CF application structure\) on z/OS” on page 610](#)

[“DELETE CHANNEL \(delete a channel\)” on page 611](#)

[“DELETE CHANNEL \(delete a channel\) MQTT” on page 613](#)

[“DELETE COMMINFO \(delete communications information\) on Multiplatforms” on page 614](#)

[“DELETE LISTENER \(delete a listener\) on Multiplatforms” on page 614](#)

[“DELETE NAMELIST \(delete a name list\)” on page 615](#)

[“DELETE POLICY \(delete a security policy\) on Multiplatforms” on page 617](#)

[“DELETE PROCESS \(delete a process definition\)” on page 618](#)

[“DELETE PSID \(delete a page set\) on z/OS” on page 619](#)

[“DELETE queues” on page 620](#)

[“DELETE SERVICE \(delete a service definition\) on Multiplatforms” on page 625](#)

[“DELETE STGCLASS \(delete a storage class\) on z/OS” on page 626](#)

[“DELETE SUB \(delete a durable subscription\)” on page 628](#)

[“DELETE TOPIC \(delete an administrative topic node\)” on page 630](#)

[“DISPLAY APSTATUS \(display application status\) on Multiplatforms” on page 632](#)

[“DISPLAY ARCHIVE \(display archive system information\) on z/OS” on page 638](#)

[“DISPLAY AUTHINFO \(display authentication information\)” on page 640](#)

[“DISPLAY AUTHREC \(display authority records\) on Multiplatforms” on page 646](#)

[“DISPLAY AUTHSERV \(display authorization services information\) on AIX, Linux, and Windows” on page 649](#)

[“DISPLAY CFSTATUS \(display CF application structure status\) on z/OS” on page 650](#)

[“DISPLAY CFSTRUCT \(display CF application structure settings\) on z/OS” on page 657](#)

[“DISPLAY CHANNEL \(display channel definition\)” on page 661](#)

[“DISPLAY CHANNEL \(display channel definition\) MQTT” on page 675](#)

[“DISPLAY CHINIT \(display channel initiator information\) on z/OS” on page 678](#)

[“DISPLAY CHLAUTH \(display channel authentication record\)” on page 680](#)

[“DISPLAY CHSTATUS \(display channel status\)” on page 685](#)

[“DISPLAY CHSTATUS \(display channel status\) AMQP” on page 705](#)

[“DISPLAY CHSTATUS \(display channel status\) MQTT” on page 709](#)

[“DISPLAY CLUSQMGR \(display channel information for cluster queue managers\)” on page 713](#)

[“DISPLAY CMDSERV \(display command server status\) on z/OS” on page 722](#)

[“DISPLAY COMMINFO \(display communication information\) on Multiplatforms” on page 722](#)

[“DISPLAY CONN \(display application connection information\)” on page 725](#)

[“DISPLAY ENTAUTH \(display entity authorizations\) on Multiplatforms” on page 739](#)

[“DISPLAY GROUP \(display QSG information\) on z/OS” on page 741](#)

[“DISPLAY LISTENER \(display listener information\) on Multiplatforms” on page 742](#)

[“DISPLAY LOG \(display log information\) on z/OS” on page 745](#)

[“DISPLAY LSSTATUS \(display listener status\) on Multiplatforms” on page 747](#)

[“DISPLAY MAXSMSGS \(display maximum messages setting\) on z/OS” on page 750](#)

[“DISPLAY NAMELIST \(display a list of names\)” on page 751](#)

[“DISPLAY POLICY \(display a security policy\) on Multiplatforms” on page 755](#)

[“DISPLAY PROCESS \(display process information\)” on page 756](#)

[“DISPLAY PUBSUB \(display publish/subscribe status information\)” on page 760](#)

[“DISPLAY QMGR \(display queue manager settings\)” on page 765](#)

[“DISPLAY QMSTATUS \(display queue manager status\) on Multiplatforms” on page 780](#)

[“DISPLAY QSTATUS \(display queue status\)” on page 789](#)

[“DISPLAY QUEUE \(display queue attributes\)” on page 801](#)

[“DISPLAY SBSTATUS \(display subscription status\)” on page 817](#)

[“DISPLAY SECURITY \(display security settings\) on z/OS” on page 821](#)

[“DISPLAY SERVICE \(display service information\) on Multiplatforms” on page 823](#)

[“DISPLAY SMDS \(display shared message data sets information\) on z/OS” on page 826](#)

[“DISPLAY SMDSCONN \(display shared message data sets connection information\) on z/OS” on page 828](#)

[“DISPLAY STGCLASS \(display storage class information\) on z/OS” on page 831](#)

[“DISPLAY SUB \(display subscription information\)” on page 835](#)

[“DISPLAY SVSTATUS \(display services status\) on Multiplatforms” on page 843](#)

[“DISPLAY SYSTEM \(display system information\) on z/OS” on page 846](#)

[“DISPLAY TCLUSTER \(display cluster topic attributes\)” on page 847](#)

[“DISPLAY THREAD \(display thread information\) on z/OS” on page 852](#)

[“DISPLAY TOPIC \(display topic information\)” on page 854](#)

[“DISPLAY TPSTATUS \(display topic status\)” on page 862](#)

[“DISPLAY TRACE \(display active traces list\) on z/OS” on page 870](#)

[“DISPLAY USAGE \(display usage information\) on z/OS” on page 873](#)

[“MOVE QLOCAL \(move messages between local queues\) on z/OS” on page 874](#)

[“PING CHANNEL \(test channel response\)” on page 877](#)

[“PING QMGR \(test queue manager response\) on Multiplatforms” on page 880](#)

[“PURGE CHANNEL \(stop and purge a channel\) on AIX, Linux, and Windows” on page 880](#)

[“RECOVER BSDS \(recover bootstrap data set\) on z/OS” on page 881](#)

[“RECOVER CFSTRUCT \(recover CF application structure\) on z/OS” on page 882](#)

[“REFRESH CLUSTER \(rebuild a cluster\)” on page 884](#)

[“REFRESH QMGR \(refresh a queue manager\)” on page 887](#)

[“REFRESH SECURITY \(refresh security settings\)” on page 890](#)

[“RESET CFSTRUCT \(reset a CF application structure\) on z/OS” on page 895](#)

[“RESET CHANNEL \(reset message sequence number for a channel\)” on page 895](#)

[“RESET CLUSTER \(reset a cluster\)” on page 898](#)

[“RESET QMGR \(reset a queue manager\)” on page 900](#)

[“RESET QSTATS \(report and reset queue performance data\) on z/OS” on page 903](#)

[“RESET SMDS \(reset shared message data sets\) on z/OS” on page 905](#)

[“RESET TPIPE \(reset sequence numbers for an IMS Tpipe\) on z/OS” on page 907](#)

[“RESOLVE CHANNEL \(ask a channel to resolve in-doubt messages\)” on page 908](#)

[“RESOLVE INDOUBT \(resolve threads left in doubt\) on z/OS” on page 911](#)

[“RESUME QMGR \(resume a cluster queue manager\)” on page 913](#)  
[“RVERIFY SECURITY \(set a user reverification flag\) on z/OS” on page 914](#)  
[“SET ARCHIVE \(change archive system settings\) on z/OS” on page 915](#)  
[“SET AUTHREC \(set authority records\) on Multiplatforms” on page 920](#)  
[“SET CHLAUTH \(create or modify a channel authentication record\)” on page 926](#)  
[“SET LOG \(notify completion of log archiving\) on Multiplatforms” on page 934](#)  
[“SET LOG \(change log system settings\) on z/OS” on page 935](#)  
[“SET POLICY \(set security policy\) on Multiplatforms” on page 938](#)  
[“SET SYSTEM \(change system settings\) on z/OS” on page 941](#)  
[“START CHANNEL \(start a channel\)” on page 945](#)  
[“START CHANNEL \(start a channel\) MQTT” on page 948](#)  
[“START CHINIT \(start a channel initiator\) on z/OS” on page 948](#)  
[“START CMDSERV \(start the command server\) on z/OS” on page 950](#)  
[“START LISTENER \(start a channel listener\)” on page 950](#)  
[“START QMGR \(start queue manager\) on z/OS” on page 953](#)  
[“START SERVICE \(start a service\) on Multiplatforms” on page 955](#)  
[“START SMDSCONN \(restart a shared message data set connection\) on z/OS” on page 956](#)  
[“START TRACE \(start trace\) on z/OS” on page 957](#)  
[“STOP CHANNEL \(stop a channel\)” on page 963](#)  
[“STOP CHANNEL \(stop a channel\) MQTT” on page 967](#)  
[“STOP CHINIT \(stop channel initiator\) on z/OS” on page 968](#)  
[“STOP CMDSERV \(stop the command server\) on z/OS” on page 969](#)  
[“STOP CONN \(stop a connection\) on Multiplatforms” on page 970](#)  
[“STOP LISTENER \(stop a channel listener\)” on page 971](#)  
[“STOP QMGR \(stop queue manager\) on z/OS” on page 973](#)  
[“STOP SERVICE \(stop a service\) on Multiplatforms” on page 975](#)  
[“STOP SMDSCONN \(stop shared message data sets connection\) on z/OS” on page 976](#)  
[“STOP TRACE \(stop trace\) on z/OS” on page 977](#)  
[“SUSPEND QMGR \(suspend a cluster queue manager\)” on page 980](#)

### **Related concepts**

[“IBM MQ control commands reference” on page 21](#)

Reference information about the IBM MQ control commands.

[Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#)

### **Related tasks**

[Administering IBM MQ using MQSC commands](#)

### **Related reference**

[“CL commands for IBM i reference” on page 1551](#)

A list of CL commands for IBM i, grouped according to command type.

[“Programmable command formats \(PCFs\) reference” on page 983](#)

PCFs define command and reply messages that can be exchanged across a network between a program and any queue manager that supports PCFs. This simplifies queue manager administration and other network administration.

### **Related information**

[Clustering: Using REFRESH CLUSTER best practices](#)

## **ALTER AUTHINFO (alter authentication information object)**


Use the MQSC command **ALTER AUTHINFO** to alter an authentication information object. These objects contain the definitions required to perform certificate revocation checking using OCSP or Certificate

Revocation Lists (CRLs) on LDAP servers, and the definitions required to check authentication credentials provided by applications.

## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

Parameters not specified in the **ALTER AUTHINFO** command result in the existing values for those parameters being left unchanged.

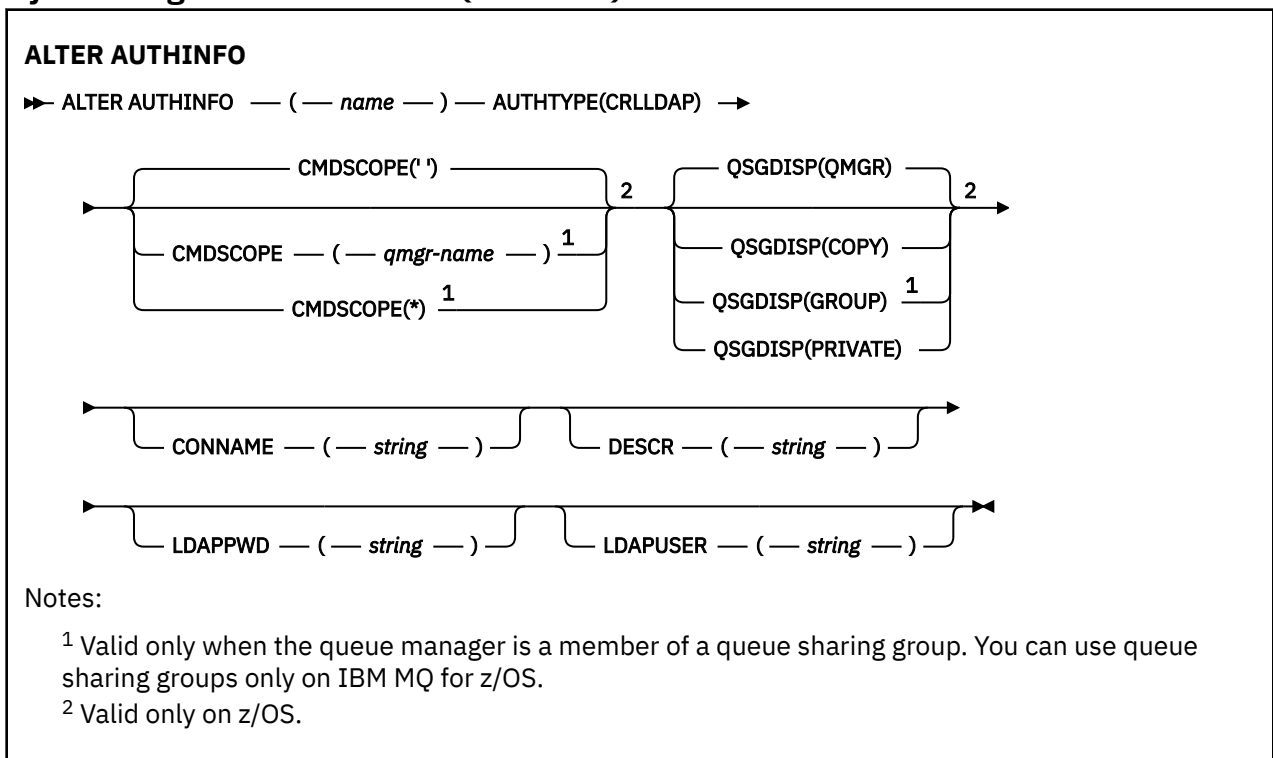
 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

There are separate syntax diagrams for each **AUTHTYPE** parameter option:

- [Syntax diagram for TYPE\(CRLLDAP\)](#)
- [Syntax diagram for TYPE\(OCSP\)](#)
- [Syntax diagram for TYPE\(IDPWOS\)](#)
- [Syntax diagram for TYPE\(IDPWLDAP\)](#)
- [“Parameter descriptions for ALTER AUTHINFO” on page 258](#)

**Synonym: ALT AUTHINFO**

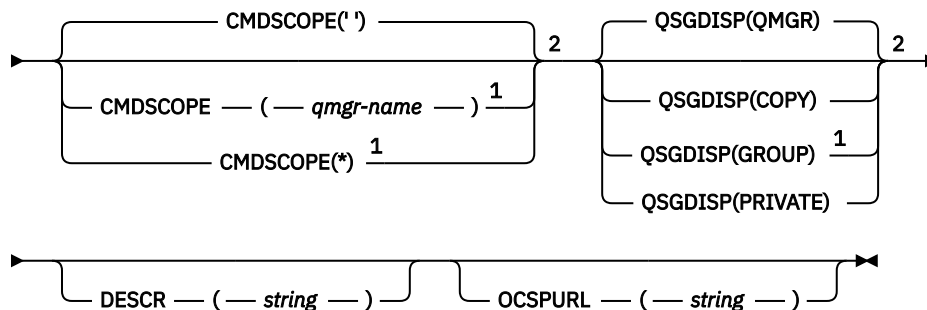
### Syntax diagram for AUTHTYPE (CRLLDAP)



## Syntax diagram for AUTHTYPE (OCSP)

### ALTER AUTHINFO

► ALTER AUTHINFO — ( — *name* — ) — AUTHTYPE(OCSP) ►



Notes:

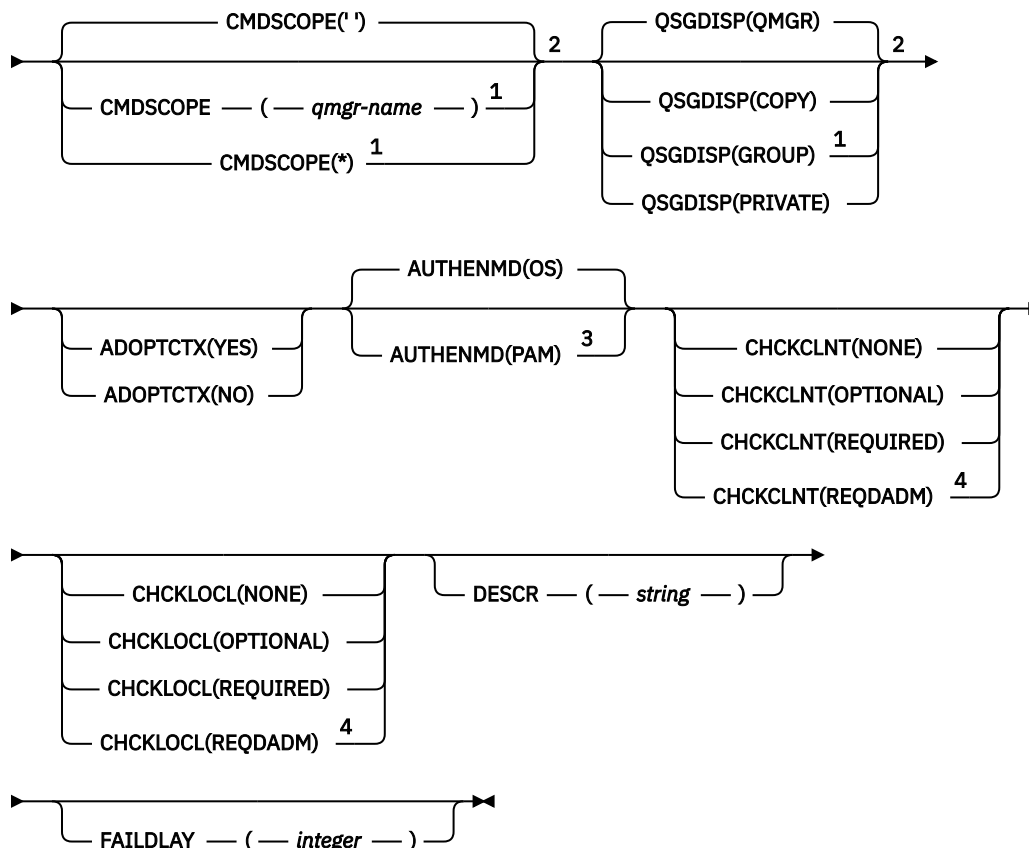
<sup>1</sup> Valid only when the queue manager is a member of a queue sharing group. You can use queue sharing groups only on IBM MQ for z/OS.

<sup>2</sup> Valid only on z/OS.

## Syntax diagram for AUTHTYPE (IDPWOS)

### ALTER AUTHINFO

► ALTER AUTHINFO — ( — *name* — ) — AUTHTYPE(IDPWOS) ►



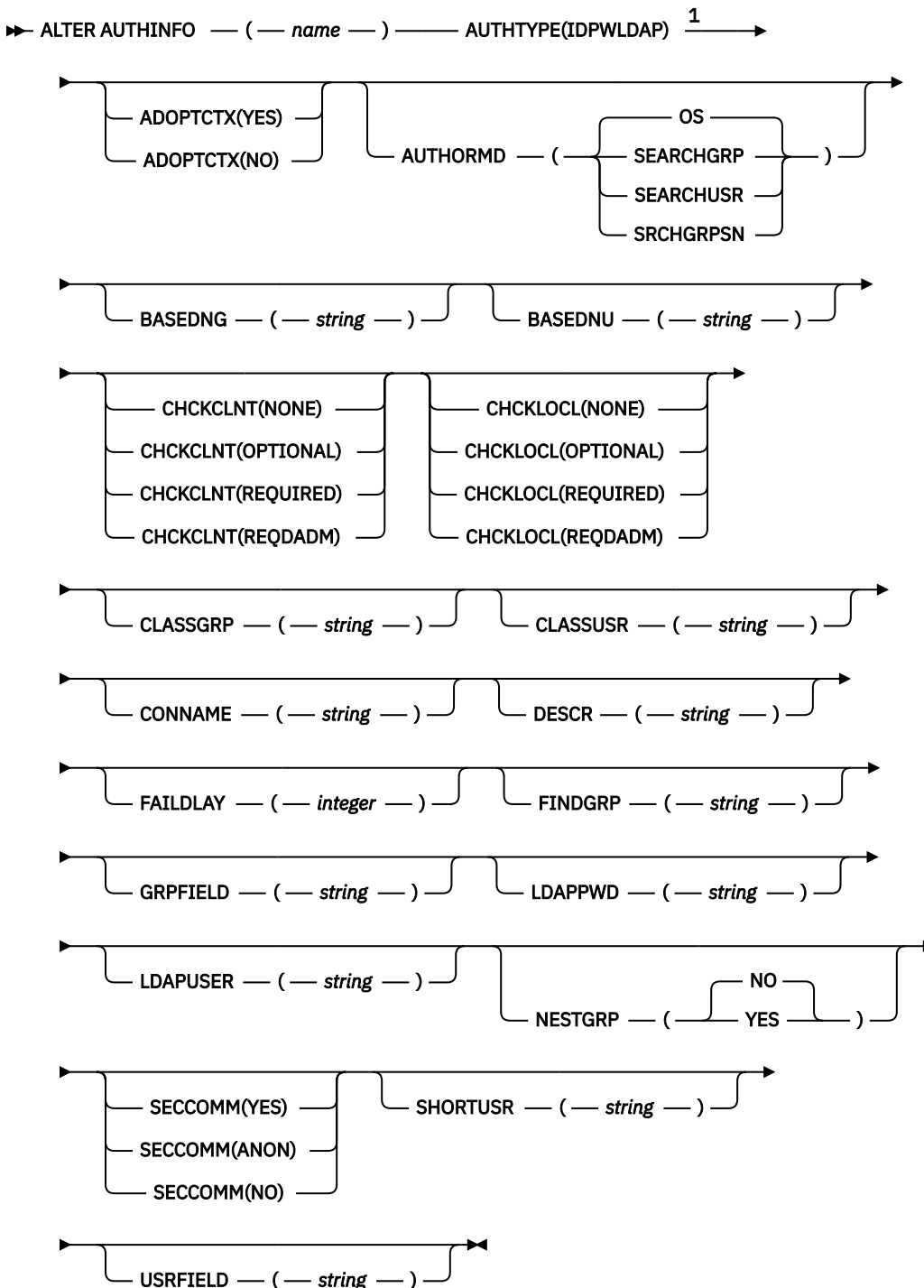
Notes:



- <sup>1</sup> Valid only when the queue manager is a member of a queue sharing group. You can use queue sharing groups only on IBM MQ for z/OS.
- <sup>2</sup> Valid only on z/OS.
- <sup>3</sup> Not valid on z/OS and PAM value can be set only on AIX and Linux.
- <sup>4</sup> Not valid on z/OS.

## Syntax diagram for AUTHTYPE (IDPWLDAP)

### ALTER AUTHINFO



Notes:

<sup>1</sup> Not valid on z/OS.

## Parameter descriptions for ALTER AUTHINFO

### *name*

Name of the authentication information object. This parameter is required.

The name must not be the same as any other authentication information object name currently defined on this queue manager (unless **REPLACE** or **ALTER** is specified). See [Rules for naming IBM MQ objects](#).


### **ADOPTCTX**

Whether to use the presented credentials as the context for this application. This means that they are used for authorization checks, shown on administrative displays, and appear in messages.

#### **YES**

The user ID in the authentication credentials presented in the MQCSP structure, which have been successfully validated, is adopted as the context to use for this application. Therefore, this user ID has the credentials checked for authorization to use IBM MQ resources.

If the application presents a user ID and password, the user ID in the MQCSP structure is adopted if the password is successfully validated.

 If the application presents an authentication token, and the token is successfully validated, the user ID in the token user claim is adopted as the context for the application. The name of the token user claim is specified by the **UserClaim** attribute in the **AuthToken** stanza of the `qm.ini` file. For more information about the **UserClaim** attribute, see [UserClaim](#).

If the user ID presented is an LDAP user ID, and authorization checks are done using operating system user IDs, the **SHORTUSR** associated with the user entry in LDAP will be adopted as the credentials for authorization checks to be done against.

**ADOPTCTX(YES)** has an effect only if **CHKCLNT** or **CHKLOCL** is set to a value that causes the credentials to be validated.

#### **NO**

Authentication is performed on the credentials presented in the MQCSP structure, but then the credentials are not adopted for further use. Authorization is performed using the user ID that the application is running under.

The **ADOPTCTX** attribute is only valid for an **AUTHTYPE** of **IDPWOS** and **IDPWLDP**.

### **AUTHENMD**


Authentication method. Whether to use the operating system or Pluggable Authentication Method (PAM) to authenticate user passwords.

#### **OS**

 Use the traditional UNIX password verification method.

#### **PAM**

Use the PAM to authenticate the user password.

 You can set the PAM value only on AIX and Linux.

Changes to this attribute are effective only after you run the [REFRESH SECURITY TYPE\(CONNAUTH\)](#) command.

The **AUTHENMD** attribute is valid only for an **AUTHTYPE** of **IDPWOS**.

### **AUTHORMD**

Authorization method.

#### **OS**

Use operating system groups to determine permissions associated with a user.

This is how IBM MQ has previously worked, and is the default value.

### **SEARCHGRP**

A group entry in the LDAP repository contains an attribute listing the Distinguished Name of all the users belonging to that group. Membership is indicated by the attribute defined in [FINDGRP](#). This value is typically *member* or *uniqueMember*.

### **SEARCHUSR**

A user entry in the LDAP repository contains an attribute listing the Distinguished Name of all the groups to which the specified user belongs. The attribute to query is defined by the [FINDGRP](#) value, typically *memberOf*.

### **SRCHGRPSN**

A group entry in the LDAP repository contains an attribute listing the short user name of all the users belonging to that group. The attribute in the user record that contains the short user name is specified by [SHORTUSR](#).

Membership is indicated by the attribute defined in [FINDGRP](#). This value is typically *memberUid*.

**Note:** This authorization method should only be used if all user short names are distinct.

Many LDAP servers use an attribute of the group object to determine group membership and you should, therefore, set this value to [SEARCHGRP](#).

Microsoft Active Directory typically stores group memberships as a user attribute. The IBM Tivoli Directory Server supports both methods.

In general, retrieving memberships through a user attribute will be faster than searching for groups that list the user as a member.

### **AUTHTYPE**

The type of authentication information.

#### **CRLLDAP**


Certificate Revocation List checking is done using LDAP servers.

#### **IDPWLDAP**

Connection authentication user ID and password checking is done using an LDAP server.


#### **IDPWOS**

Connection authentication user ID and password checking is done using the operating system.

 Authentication tokens supplied by IBM MQ MQI clients are validated if the queue manager is configured to accept authentication tokens using the **AuthToken** stanza of the `qm.ini` file. For more information about the **AuthToken** stanza, see [AuthToken stanza of the qm.ini file](#).

#### **OCSP**

Certificate revocation checking is done using OCSP.

 An authentication information object with **AUTHTYPE(OCSP)** does not apply for use on IBM i or z/OS queue managers. However, it can be specified on those platforms to be copied to the client channel definition table (CCDT) for client use.

The **AUTHTYPE** parameter is required.

You cannot define an authentication information object as **LIKE** another authentication object with a different **AUTHTYPE**. You cannot alter the **AUTHTYPE** of an authentication information object after you have created it.

### **BASEDNG**

Base DN for groups.

In order to be able to find group names, this parameter must be set with the base DN to search for groups in the LDAP server.

### **BASEDNU(base DN)**

In order to be able to find the short user name attribute, [SHORTUSR](#), this parameter must be set with the base DN to search for users within the LDAP server.

The **BASEDNU** attribute is valid only for an **AUTHTYPE** of IDPWLDAP.

## CHKCLNT

This attribute determines the authentication requirements for client applications, and is valid only for an **AUTHTYPE** of IDPWOS or IDPWLDAP. The possible values are:


### NONE

Authentication credentials supplied by client applications are not checked. If a user ID and password, or an authentication token, is supplied by a client application, the credentials are ignored. **ADOPTCTX** will have no effect and any user IDs contained within the MQCSP will not be used for authorization checks later.

### OPTIONAL

Client applications are not required to provide authentication credentials.

Any applications that do provide a user ID and password in the MQCSP structure have them authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection is only allowed to continue if the user ID and password are valid.


 If an application provides an authentication token, and the queue manager is configured to accept authentication tokens, the token is validated. The connection is only allowed to continue if the token is issued by a trusted issuer.

This option might be useful during migration, for example.

### REQUIRED

All client applications must provide authentication credentials in the MQCSP structure.

If an application provides a user ID and password, these credentials are authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection is only allowed to continue if the user ID and password are valid.

 If an application provides an authentication token, and the queue manager is configured to accept authentication tokens, the token is validated. The connection is only allowed to continue if the token is issued by a trusted issuer.


If an application does not provide any authentication credentials, the connection is rejected.

### REQDADM


All client applications using a privileged user ID must provide authentication credentials in the MQCSP structure. Any client applications using a non-privileged user ID are not required to provide authentication credentials and are treated as with the OPTIONAL setting.

A privileged user is one that has full administrative authorities for IBM MQ. See [Privileged users](#) for more information.

Any provided user ID and password are authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection is only allowed to continue if the user ID and password are valid.



 If an application provides an authentication token, and the queue manager is configured to accept authentication tokens, the token is validated. The connection is only allowed to continue if the token is issued by a trusted issuer.

**Note:** The REQDADM value for the **CHKCLNT** attribute is irrelevant if the authentication type is LDAP. This is because there is no concept of privileged user ID when using LDAP user accounts. LDAP user accounts and groups must be assigned permission explicitly.

 This setting is not allowed on z/OS systems.


### Important:

1. This attribute can be overridden by the **CHKCLNT** attribute of the CHLAUTH rule that matches the client connection. The **CONNAUTH AUTHINFO CHKCLNT** attribute on the queue manager therefore determines the default client checking behavior for client connections that do not match a CHLAUTH rule, or where the CHLAUTH rule matched has **CHKCLNT ASQMGR**.

2. If you select NONE and the client connection matches a CHLAUTH record with **CHKCLNT REQUIRED** (or REQDADM on platforms other than z/OS), the connection fails. You receive the following message:
  -  AMQ9793 on [Multiplatforms](#).
  -  CSQX793E on z/OS.
3. This parameter is valid only with **TYPE (USERMAP)**, **TYPE (ADDRESSMAP)** and **TYPE (SSLPEERMAP)**, and only when **USERSRC** is not set to NOACCESS.
4. This parameter applies only to inbound connections that are server-connection channels.

## CHKLOCL

This attribute determines the authentication requirements for locally bound applications, and is valid only for an **AUTHTYPE** of IDPWOS or IDPWLDAP.

 For information about use of this attribute on IBM MQ Appliance, see [Control commands on the IBM MQ Appliance](#) in the IBM MQ Appliance documentation.

The possible values are:




### NONE

Authentication credentials supplied by client applications are not checked. If a user ID and password is supplied by a locally bound application, the credentials are ignored.

### OPTIONAL

Locally bound applications are not required to provide authentication credentials.

Any applications that do provide a user ID and password in the MQCSP structure have them authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection is only allowed to continue if the user ID and password are valid.




   Authentication tokens cannot be supplied by locally bound applications.

This option might be useful during migration, for example.


### REQUIRED

All locally bound applications must provide authentication credentials in the MQCSP structure.

If an application provides a user ID and password, these credentials are authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection will only be allowed to continue if the user ID and password are valid.

   Authentication tokens cannot be supplied by locally bound applications.

If an application does not provide any authentication credentials, the connection is rejected.

 If your user ID has UPDATE access to the BATCH profile in the MQCONN class, you can treat **CHKLOCL (REQUIRED)** as if it is **CHKLOCL (OPTIONAL)**. That is, you do not have to supply a password, but if you do, the password must be the correct one.

See [Using CHKLOCL on locally bound applications](#).

### REQDADM

All locally bound applications using a privileged user ID must provide authentication credentials in the MQCSP structure. Any locally bound applications using a non-privileged user ID are not required to provide authentication credentials and are treated as with the OPTIONAL setting.

A privileged user is one that has full administrative authorities for IBM MQ. See [Privileged users](#) for more information.

Any provided user ID and password will be authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection will only be allowed to continue if the user ID and password are valid.

Linux V 9.3.4 AIX Authentication tokens cannot be supplied by locally bound applications.

z/OS (This setting is not allowed on z/OS systems.)

### CLASSGRP

The LDAP object class used for group records in the LDAP repository.

If the value is blank, `groupOfNames` is used.

Other commonly used values include `groupOfUniqueNames` or `group`.

### CLASSUSR(LDAP class user)

The LDAP object class used for user records in the LDAP repository.

If blank, the value defaults to `inetOrgPerson`, which is generally the value needed.

For Microsoft Active Directory, the value you require is often `user`.

This attribute is valid only for an **AUTHTYPE** of IDPWLDAP.

### z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered.

#### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

### CONNNAME(connection name)

The host name, IPv4 dotted decimal address, or IPv6 hexadecimal notation of the host on which the LDAP server is running, with an optional port number.

If you specify the connection name as an IPv6 address, only systems with an IPv6 stack are able to resolve this address. If the **AUTHINFO** object is part of the CRL namelist of the queue manager, ensure that any clients using the client channel table generated by the queue manager can resolve the connection name.

z/OS On z/OS, if a **CONNNAME** is to resolve to an IPv6 network address, a level of z/OS that supports IPv6 for connection to an LDAP server is required.

The syntax for **CONNNAME** is the same as for channels. For example,

```
connname('hostname (nnn)')
```

where *nnn* is the port number.

The maximum length for the field is:

- Multi 264 characters on [Multiplatforms](#).
- z/OS 48 characters on z/OS.

This attribute is valid only for an **AUTHTYPE** of CRLLDAP and IDPWLDAP, when the attribute is mandatory.

When used with an **AUTHTYPE** of IDPWLDAP, this can be a comma separated list of connection names.

### **DESCR(string)**

Plain-text comment. It provides descriptive information about the authentication information object when an operator issues the **DISPLAY AUTHINFO** command (see [“DISPLAY AUTHINFO \(display authentication information\)”](#) on page 640 ).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

### **FAILDLAY(delay time)**

When authentication credentials are provided for connection authentication, and the authentication fails due to the credentials being incorrect, this is the delay, in seconds, before the failure is returned to the application.

This can aid in avoiding busy loops from an application that simply retries, continuously, after receiving a failure.

The value must be in the range 0 - 60 seconds. The default value is 1.

The **FAILDLAY** attribute is valid only for an **AUTHTYPE** of IDPWOS and IDPWLDAP.

### **FINDGRP**

Name of the attribute used within an LDAP entry to determine group membership.

When **AUTHORMD** = *SEARCHGRP*, the **FINDGRP** attribute is typically set to *member* or *uniqueMember*.

When **AUTHORMD** = *SEARCHUSR*, the **FINDGRP** attribute is typically set to *memberOf*.

When **AUTHORMD** = *SRCHGRPSN*, the **FINDGRP** attribute is typically set to *memberUid*.

When left blank, if:

- **AUTHORMD** = *SEARCHGRP*, the **FINDGRP** attribute defaults to *memberOf*
- **AUTHORMD** = *SEARCHUSR*, the **FINDGRP** attribute defaults to *member*
- **AUTHORMD** = *SRCHGRPSN*, the **FINDGRP** attribute defaults to *memberUid*


### **GRPFIELD**

LDAP attribute that represents a simple name for the group.

If the value is blank, commands like **setmqaut** must use a qualified name for the group. The value can either be a full DN, or a single attribute.

### **LDAPPWD(LDAP password)**

The password associated with the Distinguished Name of the user who is accessing the LDAP server. Its maximum size is 32 characters.



 On z/OS, the **LDAPPWD** used for accessing the LDAP server might not be the one defined in the AUTHINFO object. If more than one AUTHINFO object is placed in the namelist referred to by the QMGR parameter **SSLCRLNL**, the **LDAPPWD** in the first AUTHINFO object is used for accessing all LDAP Servers.


The **GRPFIELD** attribute is valid only for an **AUTHTYPE** of CRLLDAP and IDPWLDAP.

### **LDAPUSER(LDAP user)**

The Distinguished Name of the user who is accessing the LDAP server. (See the [SSLPEER](#) parameter for more information about distinguished names.)

The maximum size for the user name is:

-  1024 characters on [Multiplatforms](#).
-  256 characters on z/OS.

 On z/OS, the **LDAPUSER** used for accessing the LDAP server might not be the one defined in the AUTHINFO object. If more than one AUTHINFO object is placed in the namelist referred

to by the QMGR parameter **SSLCRLNL**, the **LDAPUSER** in the first AUTHINFO object is used for accessing all LDAP Servers.

From IBM MQ 9.2.0, if more than one OCSP type AUTHINFO objects is referenced in the NAMELIST, only the first entry will be used.

**Multi** On Multiplatforms, the maximum accepted line length is defined to be BUFSIZ, which can be found in stdio.h.

The **LDAPUSER** attribute is valid only for an **AUTHTYPE** of CRLLDAP and IDPWLLDAP.

#### NESTGRP

Group nesting.

#### NO

Only the initially discovered groups are considered for authorization.

#### YES

The group list is searched recursively to enumerate all the groups to which a user belongs.

The group's Distinguished Name is used when searching the group list recursively, regardless of the authorization method selected in AUTHORMD.

#### OCSPURL(Responder URL)

The URL of the OCSP responder used to check for certificate revocation. This value must be an HTTP URL containing the host name and port number of the OCSP responder. If the OCSP responder is using port 80, which is the default for HTTP, then the port number can be omitted. HTTP URLs are defined in RFC 1738.

This field is case sensitive. It must start with the string `http://` in lowercase. The rest of the URL might be case sensitive, depending on the OCSP server implementation. To preserve case, use single quotation marks to specify the OCSPURL parameter value, for example:

```
OCSPURL ('http://ocsp.example.ibm.com')
```

This parameter is applicable only for **AUTHTYPE (OCSP)**, when it is mandatory.

#### **z/OS** QSGDISP

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

<i>Table 127. Behavior for each of the QSGDISP values</i>	
QSGDISP	ALTER
COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (COPY)</b> . Any object residing in the shared repository, or any object defined using a command that had the parameters <b>QSGDISP (QMGR)</b> , is not affected by this command.



Table 127. Behavior for each of the QSGDISP values (continued)

QSGDISP	ALTER
GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameters <b>QSGDISP (GROUP)</b>. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command. If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre data-bbox="613 472 1464 573">DEFINE AUTHINFO (name) REPLACE QSGDISP (COPY)</pre> <p>The ALTER for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
PRIVATE	<p>The object resides on the page set of the queue manager that executes the command, and was defined with <b>QSGDISP (QMGR)</b> or <b>QSGDISP (COPY)</b>. Any object residing in the shared repository is unaffected.</p>
QMGR	<p>The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (QMGR)</b>. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.</p>

**SECCOMM**

Whether connectivity to the LDAP server should be done securely using TLS

**YES**

Connectivity to the LDAP server is made securely using TLS.

The certificate used is the default certificate for the queue manager, named in CERTLABL on the queue manager object, or if that is blank, the one described in [Digital certificate labels, understanding the requirements](#).

The certificate is located in the key repository specified in SSLKEYR on the queue manager object. A cipherspec will be negotiated that is supported by both IBM MQ and the LDAP server.

If the queue manager is configured to use **SSLFIPS (YES)** or SUITEB cipher specs, then this is taken account of in the connection to the LDAP server as well.

**ANON**

Connectivity to the LDAP server is made securely using TLS just as for **SECCOMM (YES)** with one difference.

No certificate is sent to the LDAP server; the connection will be made anonymously. To use this setting, ensure that the key repository specified in SSLKEYR, on the queue manager object, does not contain a certificate marked as the default.

**NO**

Connectivity to the LDAP server does not use TLS.

The **SECCOMM** attribute is valid only for an **AUTHTYPE** of IDPWLDAP.

**SHORTUSR(user name)**

A field in the user record to be used as a short user name in IBM MQ.

This field must contain values of 12 characters or less. This short user name is used for the following purposes:

- If LDAP authentication is enabled, but LDAP authorization is not enabled, this is used as an operating system user ID for authorization checks. In this case, the attribute must represent an operating system user ID.
- If LDAP authentication and authorization are both enabled, this is used as the user ID carried with the message in order for the LDAP user name to be rediscovered when the user ID inside the message needs to be used.

For example, on another queue manager, or when writing report messages. In this case, the attribute does not need to represent an operating system user ID, but must be a unique string. An employee serial number is an example of a good attribute for this purpose.

The **SHORTUSR** attribute is valid only for an **AUTHTYPE** of IDPWLDAP and is mandatory.

### USRFIELD(*user field*)

If the user ID provided by an application for authentication does not contain a qualifier for the field in the LDAP user record, that is, it does not contain an '=' sign, this attribute identifies the field in the LDAP user record that is used to interpret the provided user ID.

This field can be blank. If this is the case, any unqualified user IDs use the **SHORTUSR** parameter to interpret the provided user ID.

The contents of this field are concatenated with an '=' sign, together with the value provided by the application, to form the full user ID to be located in an LDAP user record. For example, the application provides a user of fred and this field has the value cn, then the LDAP repository will be searched for cn=fred.

The **USRFIELD** attribute is valid only for an **AUTHTYPE** of IDPWLDAP.

## z/OS ALTER BUFFPOOL (alter buffer pool settings) on z/OS

Use the MQSC command **ALTER BUFFPOOL** to dynamically change the settings of a predefined buffer pool on z/OS.

### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

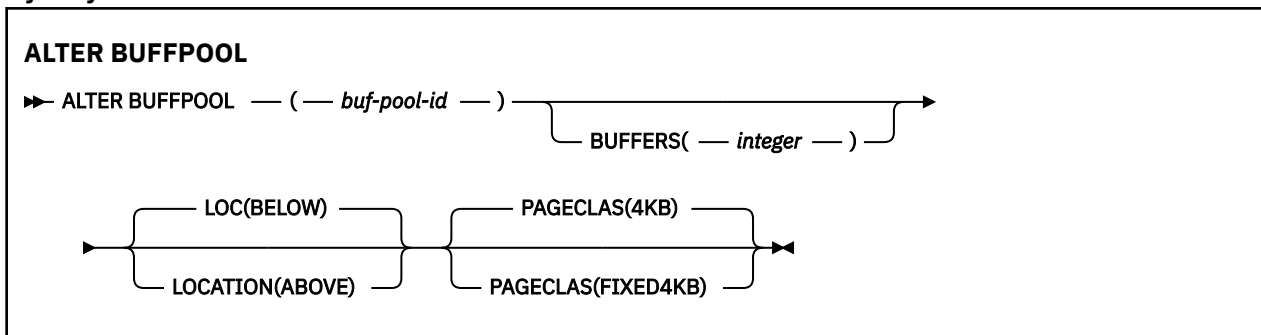
Parameters not specified in the **ALTER BUFFPOOL** command result in the existing values for those parameters being left unchanged.

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for ALTER BUFFPOOL” on page 267](#)
- [“Parameter descriptions for ALTER BUFFPOOL” on page 267](#)

### Syntax diagram

Synonym: ALT BP



## Usage notes for ALTER BUFFPOOL

1. Buffers are added or removed according to whether the value is more than or less than the current allocation (which can be shown by the DISPLAY USAGE command).
2. If there is insufficient storage, of the type specified by the PAGECLAS attribute to add the requested number, as many as possible are added.
3. The command runs asynchronously. Message CSQP023I is sent to the console when the command is complete.
4. **ALTER BUFFPOOL** cannot be issued from CSQINPT.
5. If you have made changes to buffer pools by using the ALTER BUFFPOOL command, particularly if you have reduced the buffer pools by large quantities, you should recycle the queue manager as soon as possible, to clear up any storage fragmentation caused by the change in buffer pool size.

Failure to recycle the queue manager, might result in you receiving the following error code, ABEND878-10 - Virtual private region depleted, caused by the fragmentation of region storage in the IBM MQ MSTR address space.

## Parameter descriptions for ALTER BUFFPOOL

### *(buf-pool-id)*

Buffer pool identifier.

This parameter is an integer in the range zero through 99.

### **BUFFERS**(integer)

This parameter is optional and is the number of 4096 byte buffers to be used in this buffer pool.

If the value of the **LOCATION** parameter is **BELOW**, the minimum value of buffers is 100 and the maximum value is 500,000. If the value of the **LOCATION** parameter is **ABOVE**, then valid values are in the range of 100 to 999999999 (nine nines). The storage used for buffers in a buffer pool with **LOCATION ABOVE** is obtained in multiples of 4MB. Therefore specifying a **BUFFERS** value which is a multiple of 1024 will make the most efficient use of storage.

See [Buffers and buffer pools](#) for guidance on the number of buffers you can define in each buffer pool.

When defining a buffer pool care should be taken to ensure that there is sufficient storage available for it either above or below the bar. For more information, see [Address space storage](#).

**Note:** Creating a large buffer pool can take several minutes depending on size of the buffer pool and machine configuration. In some cases message CSQP061I might be output.

### **LOCATION(LOC)**(**BELOW** or **ABOVE**)

**LOCATION** and **LOC** are synonyms and either, but not both, can be used.

The **LOCATION** or **LOC** parameter specifies where the memory used by the specified buffer pool is located.



**Attention:** Deprecated From IBM MQ 9.1, **LOCATION(BELOW)** is deprecated and you should use **LOCATION(ABOVE)** only.

This memory location can be either **ABOVE** (64 bit) or **BELOW** (31 bit) the bar. Valid values for this parameter are **BELOW** or **ABOVE**, with **BELOW** being the default.

When altering a buffer pool, you should take care to make sure that there is sufficient storage available if increasing the number of buffers, or changing the **LOCATION** value. Switching the location of the buffer pool can be a CPU and I/O intensive task. You should perform this task when the queue manager is not being heavily used.

For more information, see [Address space storage](#).

### **PAGECLAS**(**4KB** or **FIXED4KB**)

Optional parameter that describes the type of virtual storage pages used for backing the buffers in the buffer pool.

This attribute applies to all buffers in the buffer pool, including any that are added later as a result of using the **ALTER BUFFPOOL** command. The default value is 4KB, which means that pageable 4KB pages are used to back the buffers in the pool.

4KB is the only valid value if the buffer pool has its location attribute set to BELOW. If the buffer pool has its LOCATION attribute set to ABOVE, it is also possible to specify FIXED4KB. This means that fixed 4KB pages, which are permanently in real storage and will never be paged out to auxiliary storage, are used to back the buffers in the buffer pool.

The **PAGECLAS** attribute of a buffer pool can be altered at any time. However, the alteration only takes place when the buffer pool switches location from above the bar, to below the bar, or the other way round. Otherwise, the value is stored in the log of the queue manager and is applied when the queue manager next restarts.

The current value of **PAGECLAS** can be checked by issuing the **DISPLAY USAGE PSID(\*)** command. Doing this also results in a CSQP062I message being output, if the current value of **PAGECLAS** is different from the value in the log of the queue manager. For example:

- Buffer pool 7 currently has **LOCATION(ABOVE)** and **PAGECLAS(4KB)** specified. If **ALTER BUFFPOOL(7) PAGECLAS(FIXED4KB)** is specified, the buffer pool continues to be backed by pageable 4KB pages as the **LOCATION** has not been changed.
- Buffer pool 8 currently has **LOCATION(BELOW)** and **PAGECLAS(4KB)** specified. If **ALTER BUFFPOOL(8) LOCATION(ABOVE) PAGECLAS(FIXED4KB)** is specified, the buffer pool is moved above the bar and has its buffers backed by fixed 4KB pages, if any are available.

When you specify **PAGECLAS(FIXED4KB)** the whole buffer pool is backed by page-fixed 4KB pages, so ensure that there is sufficient real storage available on the LPAR. Otherwise, the queue manager might not start, or other address spaces might be impacted; for more information, see [Address space storage](#).

See IBM MQ Support Pac MP16: [IBM MQ for z/OS - Capacity planning & tuning](#) for advice on when to use the FIXED4KB value of the **PAGECLAS** attribute.

## **ALTER CFSTRUCT (alter CF application structure) on z/OS**

On z/OS, use the MQSC command **ALTER CFSTRUCT** to alter the CF application structure backup and recovery parameters, and offload environment parameters for any specified application structure.

### **Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Parameters not specified in the **ALTER CFSTRUCT** command result in the existing values for those parameters being left unchanged.

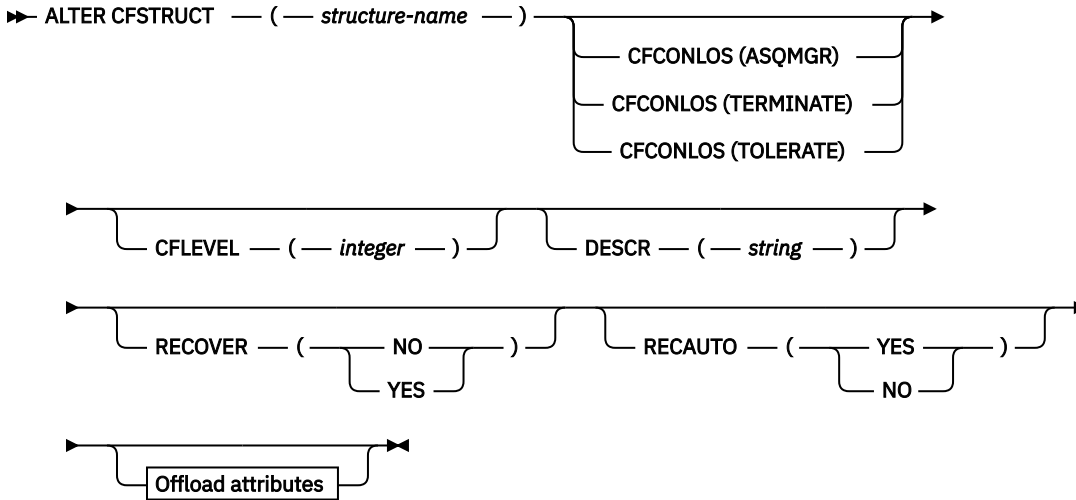
You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 269](#)
- [“Parameter descriptions for ALTER CFSTRUCT” on page 269](#)

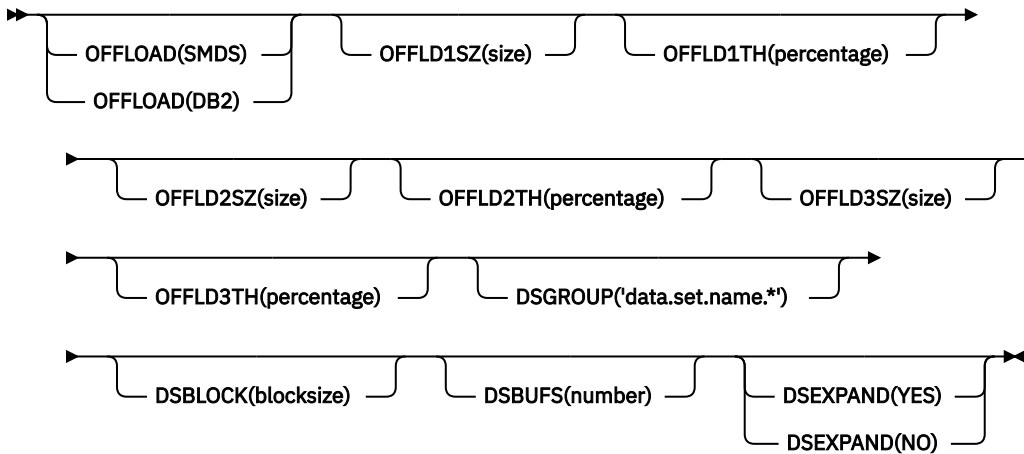
### **Syntax diagram**

**Synonym:** ALT CFSTRUCT

## ALTER CFSTRUCT



### Offload attributes



## Usage notes

- This command cannot specify the CF administration structure (CSQ\_ADMIN).
- This command is valid only when the queue manager is a member of a queue sharing group.

## Parameter descriptions for ALTER CFSTRUCT

### (structure-name)

Name of the coupling facility application structure with queue manager CF level capability and backup and recovery parameters you want to define. This parameter is required.

The name:

- Cannot have more than 12 characters.
- Must start with an uppercase letter (A through Z).
- Can include only the characters A through Z and 0 through 9.

The name of the queue sharing group to which the queue manager is connected is prefixed to the name you supply. The name of the queue sharing group is always four characters, padded with @ symbols if necessary. For example, if you use a queue sharing group named NY03 and you supply the name PRODUCT7, the resultant coupling facility structure name is NY03PRODUCT7. The administrative

structure for the queue sharing group (in this case NY03CSQ\_ADMIN) cannot be used for storing messages.

### **CFCONLOS**

This parameter specifies the action to be taken when a queue manager loses connectivity to the CF structure. The value can be:

#### **ASQMGR**

The action taken is based on the setting of the **CFCONLOS** queue manager attribute.

#### **TERMINATE**

The queue manager terminates when connectivity to the structure is lost. This is the default value when **CFLEVEL** is increased to 5.

#### **TOLERATE**

The queue manager tolerates loss of connectivity to the structure without terminating.

The **CFCONLOS** parameter is only valid from **CFLEVEL (5)**.

### **CFLEVEL(integer)**

Specifies the functional capability level for this CF application structure. Value can be one of the following:

#### **1**

A CF structure that can be "auto-created" by a queue manager at command level 520.

#### **2**

A CF structure at command level 520 that can only be created or deleted by a queue manager at command level 530 or greater.

#### **3**

A CF structure at command level 530. This **CFLEVEL** is required if you want to use persistent messages for either one or both of the following reasons:

- On shared queues, if **RECOVER(YES)** is set.
- For message grouping when a local queue is defined with **INDXTYPE(GROUPID)**.

You can only increase the value of **CFLEVEL** to 3 if all the queue managers in the queue sharing group are at command level 530 or greater - this is to ensure that there are no latent command level 520 connections to queues referencing the structure.

You can only decrease the value of **CFLEVEL** from 3 if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

#### **4**

This **CFLEVEL** supports all the **CFLEVEL (3)** functions. **CFLEVEL (4)** allows queues defined with CF structures at this level to have messages with a length greater than 63 KB.

Only a queue manager with a command level of 600 or greater can connect to a CF structure at **CFLEVEL (4)**.

You can only increase the value of **CFLEVEL** to 4 if all the queue managers in the queue sharing group are at command level 600 or greater.

You can only decrease the value of **CFLEVEL** from 4 if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

#### **5**

This **CFLEVEL** supports all functions for **CFLEVEL (4)**. In addition, **CFLEVEL (5)** enables the following new functions. If altering an existing **CFSTRUCT** to **CFLEVEL (5)**, you must review other attributes as indicated:

- Queues defined with CF structures at this level can have message data offloaded to either shared message data sets (SMDS), or Db2®, under control of the **OFFLOAD** attribute. The offload threshold and size parameters (such as **OFFLD1TH**, and **OFFLD1SZ**) determine whether any particular messages are offloaded given its size and current CF structure utilization. If using SMDS offload, the **DSGROUP**, **DSBUFS**, **DSEXPAND** and **DSBLOCK** attributes are respected.

- Structures at **CFLEVEL (5)** allow the queue manager to tolerate a loss of connectivity to the CF structure. The **CFCONLOS** attribute determines queue manager behavior when a loss of connectivity is detected, and the **RECAUTO** attribute controls subsequent automatic structure recovery behavior.
- Messages containing IBM MQ message properties are stored in a different format on shared queues in a **CFLEVEL (5)** structure. This format leads to internal processing optimizations. Additional application migration capabilities are also available and these are enabled via the queue **PROPCTL** attribute.

Only a queue manager with a command level of 710 or above can connect to a CF structure at **CFLEVEL (5)**.

**Note:** You can decrease the value of **CFLEVEL** from 5 if all the queues that reference the CF structure are both empty, that is the queues, and CF structure have no messages or uncommitted activity, and are closed.

### **DESCR(string)**

Plain-text comment that provides descriptive information about the object when an operator issues the **DISPLAY CFSTRUCT** command.

The string should contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

### **OFFLOAD**

Specify whether offloaded message data is to be stored in a group of shared message data sets or in Db2.

#### **SMDS**

Offload messages from coupling facility to shared message data set (SMDS).

#### **Db2**

Offload messages from coupling facility to Db2. This value is the default assumption when **CFLEVEL** is increased to 5.

Offloading messages using Db2 has significant performance impact. If you want to use offload rules as a means of increasing capacity, the SMDS option should be specified.

This parameter is only valid from **CFLEVEL (5)**. At **CFLEVEL (4)** any message offloading is always to Db2, and only applies to messages greater than the maximum coupling facility entry size.

**Note:** If you change the offload technique (from Db2 to SMDS or the other way) then all new messages will be written using the new method but any existing large messages stored using the previous technique can still be retrieved. The relevant Db2 message table or shared message data sets will continue to be used until the queue managers have detected that there are no further messages stored in the old format.

If SMDS is specified, then the **DSGROUP** parameter is also required. It can be specified either on the same command or on a previous **DEFINE** or **ALTER** command for the same structure.

### **OFFLD1TH(percentage) OFFLD1SZ(size)**

### **OFFLD2TH(percentage) OFFLD2SZ(size)**

### **OFFLD3TH(percentage) OFFLD3SZ(size)**

Specify rules for when messages smaller than the maximum coupling facility entry size are to be offloaded to external storage (shared message data sets or Db2 tables) instead of being stored in the application structure. These rules can be used to increase the effective capacity of the structure. The offloaded message still requires an entry in the coupling facility containing message control information, and a descriptor referring to the offloaded message data, but the amount of structure space required is less than the amount that would be needed to store the whole message.

If the message data is very small (less than approximately 140 bytes) it may fit into the same coupling facility entry as the message control information, without needing additional data elements. In this case, no space can be saved, so any offload rules are ignored and the message data is not offloaded.

Messages exceeding the maximum coupling facility entry size (63.75 KB including control information) are always offloaded as they cannot be stored in a coupling facility entry. Messages where the message body exceeds 63 KB are also offloaded to ensure that enough space is available for the control information. Additional rules to request offloading of smaller messages can be specified using these pairs of keywords. Each rule indicates that when the usage of the structure (in either elements or entries) exceeds the specified threshold percentage value, the message data will be offloaded if the total size of the coupling facility entry required to store the whole message (including message data, headers and descriptors) exceeds the specified size value. Headers and descriptors typically require approximately 400 bytes.

#### **percentage**

The usage threshold percentage value is an integer in the range 0 (meaning this rule always applies) to 100 (meaning this rule only applies when the structure is full).

#### **size**

The message size value should be specified as an integer followed by K, giving the number of kilobytes in the range 0K to 64K. As messages exceeding 63.75 KB are always offloaded, the value 64K is allowed as a simple way to indicate that the rule is not being used.

In general, the smaller the numbers, the more messages are offloaded.

A message is offloaded if any offload rule matches. The normal convention is that a later rule would be for a higher usage level and a smaller message size than an earlier one, but no check is made for consistency or redundancy between the rules.

When structure **ALTER** processing is active, the number of used elements or entries can temporarily exceed the reported total number, giving a percentage exceeding 100, because the new elements or entries are made available during **ALTER** processing but the total is only updated when the **ALTER** completes. At such times, a rule specifying 100 for the threshold may temporarily take effect. If a rule is not intended to be used at all, it should specify 64K for the size.

The default values assumed for the offload rules when defining a new structure at **CFLEVEL (5)** or upgrading an existing structure to **CFLEVEL (5)** depend on the **OFFLOAD** method option. For **OFFLOAD (SMDS)**, the default rules specify increasing amounts of offloading as the structure becomes full. This increases the effective structure capacity with minimal performance impact. For **OFFLOAD (Db2)**, the default rules have the same threshold values as for SMDS but the size values are set to 64K so that the rules never apply and messages are offloaded only if they are too large to be stored in the structure, as for **CFLEVEL (4)**.

For **OFFLOAD (SMDS)** the defaults are:

- **OFFLD1TH(70) OFFLD1SZ(32K)**
- **OFFLD2TH(80) OFFLD2SZ(4K)**
- **OFFLD3TH(90) OFFLD3SZ(0K)**

For **OFFLOAD (Db2)** the defaults are:

- **OFFLD1TH(70) OFFLD1SZ(64K)**
- **OFFLD2TH(80) OFFLD2SZ(64K)**
- **OFFLD3TH(90) OFFLD3SZ(64K)**

If the **OFFLOAD** method option is changed from Db2 to SMDS or back when the current offload rules all match the default values for the old method, the offload rules are switched to the default values for the new method. However, if any of the rules have been changed, the current values are kept when switching method.

These parameters are only valid from **CFLEVEL (5)**. At **CFLEVEL (4)**, any message offloading is always to Db2, and only applies to messages greater than the maximum coupling facility entry size.

#### **DSGROUP**

For **OFFLOAD (SMDS)**, specify the generic data set name to be used for the group of shared message data sets associated with this structure (one for each queue manager), with exactly one asterisk indicating where the queue manager name should be inserted to form the specific data set name.



**'data.set.name.\*'**

The value must be a valid data set name when the asterisk is replaced by a queue manager name of up to four characters. The queue manager name can form all or part of any qualifier in the data set name.

The entire parameter value must be enclosed in quotation marks.

This parameter cannot be changed after any data sets have been activated for the structure.

If SMDS is specified, then the **DSGROUP** parameter must also be specified.

The **DSGROUP** parameter is only valid from **CFLEVEL (5)**.

**DSBLOCK**

For **OFFLOAD (SMDS)**, specify the logical block size, which is the unit in which shared message data set space is allocated to individual queues.

**8K****16K****32K****64K****128K****256K****512K****1M**

Each message is written starting at the next page within the current block and is allocated further blocks as needed. A larger size decreases space management requirements and reduces I/O for large messages, but increases buffer space requirements and disk space requirements for small queues.

This parameter cannot be changed after any data sets have been activated for the structure.

The **DSBLOCK** parameter is only valid from **CFLEVEL (5)**.

**DSBUFS**

For **OFFLOAD (SMDS)**, specify the number of buffers to be allocated in each queue manager for accessing shared message data sets, as a number in the range 1 - 9999. The size of each buffer is equal to the logical block size. SMDS buffers are allocated in memory objects residing in z/OS 64-bit storage (above the bar).

**number**

This parameter can be overridden for individual queue managers using the **DSBUFS** parameter on **ALTER SMDS**.

When this parameter is altered, any queue managers which are already connected to the structure (and which do not have an individual DSBUFS override value) dynamically increase or decrease the number of data set buffers being used for this structure to match the new value. If the specified target value cannot be reached, the affected queue manager adjusts the DSBUFS parameter associated with its own individual SMDS definition (as for the **ALTER SMDS** command) to match the actual new number of buffers.

These buffers use virtual storage. You should work with the z/OS systems programmer to ensure there is sufficient auxiliary storage available before increasing the number of buffers.

The **DSBUFS** parameter is only valid from **CFLEVEL (5)**.

**DSEXPA**

For **OFFLOAD (SMDS)**, this parameter controls whether the queue manager should expand a shared message data set when it becomes nearly full, and further blocks are required in the data set.

**YES**

Expansion is supported.

Each time expansion is required, the data set is expanded by the secondary allocation specified when the data set was defined. If no secondary allocation was specified, or it was specified as zero, then a secondary allocation amount of approximately 10% of the existing size is used

**NO**

No automatic data set expansion is to take place.

This parameter can be overridden for individual queue managers using the **DSEXPAND** parameter on **ALTER SMDS**.

If an expansion attempt fails, the **DSEXPAND** override for the affected queue manager is automatically changed to **NO** to prevent further expansion attempts, but it can be changed back to **YES** using the **ALTER SMDS** command to enable further expansion attempts.

When this parameter is altered, any queue managers which are already connected to the structure (and which do not have an individual **DSEXPAND** override value) immediately start using the new parameter value.

The **DSEXPAND** parameter is only valid from **CFLEVEL (5)**.

**RECOVER**

Specifies whether CF recovery is supported for the application structure. Values are:

**NO**

CF application structure recovery is not supported. (The synonym is N.)

**YES**

CF application structure recovery is supported. (The synonym is Y.)

You can only set **RECOVER(YES)** if the structure has a **CFLEVEL** of 3 or higher. Set **RECOVER(YES)** if you intend to use persistent messages.

You can only change **RECOVER(NO)** to **RECOVER(YES)** if all the queue managers in the queue sharing group are at command level 530 or greater ; this is to ensure that there are no latent command level 520 connections to queues referencing the **CFSTRUCT**.

You can only change **RECOVER(YES)** to **RECOVER(NO)** if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

**RECAUTO**

Specifies the automatic recovery action to be taken when a queue manager detects that the structure is failed or when a queue manager loses connectivity to the structure and no systems in the SysPlex have connectivity to the Coupling Facility that the structure is allocated in. Values can be:

**YES**

The structure and associated shared message data sets which also need recovery are automatically recovered. (The synonym is Y).

**NO**

The structure is not automatically recovered. (The synonym is N). This is the default value when **CFLEVEL** is increased to 5.

This parameter has no effect for structures defined with **RECOVER(NO)**.

The **RECAUTO** parameter is only valid from **CFLEVEL (5)**.

**ALTER CHANNEL (alter channel settings)**

Use the MQSC command **ALTER CHANNEL** to alter the parameters of a channel.

**Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

Parameters not specified in the **ALTER CHANNEL** command result in the existing values for those parameters being left unchanged.

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

**Synonym: ALT CHL**

- [“Syntax diagrams” on page 275](#)

- “Usage notes” on page 275
- “Parameter descriptions for ALTER CHANNEL” on page 275

## Syntax diagrams

The syntax diagrams for **ALTER CHANNEL** are in the subtopics. There is a separate syntax diagram for each channel type.

## Usage notes

- Changes take effect after the channel is next started.
- For cluster channels (the CLUSSDR and CLUSRCVR columns in the table), if an attribute can be set on both channels, set it on both and ensure that the settings are identical. If there is any discrepancy between the settings, those that you specify on the CLUSRCVR channel are likely to be used. This is explained in [Cluster channels](#).
- If you change the **XMITQ** name or the **CONNNAME**, you must reset the sequence number at both ends of the channel. (See “[RESET CHANNEL \(reset message sequence number for a channel\)](#)” on page 895 for information about the **SEQNUM** parameter.)
- Successful completion of the command does not mean that the action completed. To check for true completion, see the [ALTER CHANNEL](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for ALTER CHANNEL

The following table shows the parameters that are relevant for each type of channel. There is a description of each parameter after the table. Parameters are optional unless the description states that they are required.

Parameter	SDR	SVR	RCVR	RQSTR	CLNTCONN	SVRCO NN	CLUSSDR	CLUSRCVR	AMQP
<a href="#">AFFINITY</a>					✓				
<a href="#">AMQPKA</a>									✓
<a href="#">BATCHHB</a>	✓	✓					✓	✓	
<a href="#">BATCHINT</a>	✓	✓					✓	✓	
<a href="#">BATCHLIM</a>	✓	✓					✓	✓	
<a href="#">BATCHSZ</a>	✓	✓	✓	✓			✓	✓	
<a href="#">CERTLABL</a>	✓	✓	✓	✓	✓	✓		✓	✓
<i>channel-name</i>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">CHLTYPE</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">CLNTWGHT</a>					✓				
<a href="#">CLUSNL</a>							✓	✓	
<a href="#">CLUSTER</a>							✓	✓	
<a href="#">CLWLPRTY</a>							✓	✓	

Table 128. DEFINE and ALTER CHANNEL parameters (continued)


Parameter	SDR	SVR	RCVR	RQSTR	CLNTC ONN	SVRCO NN	CLUSSD R	CLUSR CVR	AMQP
<u>CLWLRANK</u>							✓	✓	
<u>CLWLWGHT</u>							✓	✓	
 <u>z/OS</u> <u>CMDSCOPE</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>COMPHDR</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>COMPMSG</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>CONNAME</u>	✓	✓		✓	✓		✓	✓	
<u>CONVERT</u>	✓	✓					✓	✓	
<u>DEFCDISP</u>	✓	✓	✓	✓		✓			
<u>DEFRECON</u>					✓				
<u>DESCR</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>DISCINT</u>	✓	✓				✓	✓	✓	
<u>HBINT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>KAINT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>LIKE</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>LOCLADDR</u>	✓	✓		✓	✓		✓	✓	✓
<u>LONGRTY</u>	✓	✓					✓	✓	
<u>LONGTMR</u>	✓	✓					✓	✓	
<u>MAXINST</u>						✓			✓
<u>MAXINSTC</u>						✓			
<u>MAXMSGL</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>MCANAME</u>	✓	✓		✓			✓	✓	
<u>MCTYPE</u>	✓	✓		✓			✓	✓	
<u>MCAUSER</u>			✓	✓		✓		✓	✓
<u>MODENAME</u>	✓	✓		✓	✓		✓	✓	
<u>MONCHL</u>	✓	✓	✓	✓		✓	✓	✓	
<u>MRDATA</u>			✓	✓				✓	
<u>MREXIT</u>			✓	✓				✓	

Table 128. DEFINE and ALTER CHANNEL parameters (continued)



Parameter	SDR	SVR	RCVR	RQSTR	CLNTC ONN	SVRCO NN	CLUSSD R	CLUSR CVR	AMQP
<u>MRRTY</u>			✓	✓				✓	
<u>MRTMR</u>			✓	✓				✓	
<u>MSGDATA</u>	✓	✓	✓	✓			✓	✓	
<u>MSGEXIT</u>	✓	✓	✓	✓			✓	✓	
<u>NETPRTY</u>								✓	
<u>NPMSPEED</u>	✓	✓	✓	✓			✓	✓	
<u>PASSWORD</u>	✓	✓		✓	✓		✓	✓	
<u>PORT</u>									✓
<u>PROPCTL</u>	✓	✓					✓	✓	
<u>PUTAUT</u>			✓	✓		✓		✓	
<u>QMNAME</u>					✓				
 <u>QSGDISP</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>RCVDATA</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>RCVEXIT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>REPLACE</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SCYDATA</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SCYEXIT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SENDDATA</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SENDEXIT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SEQWRAP</u>	✓	✓	✓	✓			✓	✓	
<u>SHARECNV</u>					✓	✓			
<u>SHORTRTY</u>	✓	✓					✓	✓	
<u>SHORTTMR</u>	✓	✓					✓	✓	
 <u>SPLPROT</u>	✓	✓	✓	✓					
<u>SSLCAUTH</u>		✓	✓	✓		✓		✓	
<u>SSLCIPH</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 128. DEFINE and ALTER CHANNEL parameters (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNTC ONN	SVRCO NN	CLUSSD R	CLUSR CVR	AMQP
<u>SSLPEER</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>STATCHL</u>	✓	✓	✓	✓			✓	✓	
<div style="background-color: #d4c08e; padding: 2px;">Multi</div> <div style="background-color: #4a7ebb; color: white; padding: 2px;">V 9.3.0</div> <u>TMPMODEL</u>									✓
<div style="background-color: #d4c08e; padding: 2px;">Multi</div> <div style="background-color: #4a7ebb; color: white; padding: 2px;">V 9.3.0</div> <u>TMPQPRFX</u>									✓
<u>TPNAME</u>	✓	✓		✓	✓	✓	✓	✓	
<u>TPROOT</u>									✓
<u>TRPTYPE</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>USECLTID</u>									✓
<u>USEDLQ</u>	✓	✓	✓	✓			✓	✓	
<u>USERID</u>	✓	✓		✓	✓		✓		
<u>XMITQ</u>	✓	✓							

### AFFINITY

The channel affinity attribute is used so client applications that connect multiple times using the same queue manager name can choose whether to use the same client channel definition for each connection. This attribute is intended to be used when multiple applicable channel definitions are available.

### PREFERRED

The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions based on the weighting with any applicable **CLNTWGHT (0)** definitions first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non-**CLNTWGHT (0)** definitions are moved to the end of the list. **CLNTWGHT (0)** definitions remain at the start of the list and are selected first for each connection. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT has been modified since the list was created. Each client process with the same host name creates the same list.

### NONE

The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable **CLNTWGHT (0)** definitions selected first in alphabetical order. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT has been modified since the list was created.

For example, suppose the CCDT includes the following definitions:

```
CHLNAME(A) QMNAME(QM1) CLNTWGHT(3)
CHLNAME(B) QMNAME(QM1) CLNTWGHT(4)
CHLNAME(C) QMNAME(QM1) CLNTWGHT(4)
```

The first connection in a process creates its own ordered list based on the weightings. So it might, for example, create the ordered list CHLNAME(B), CHLNAME(A), CHLNAME(C).

For **AFFINITY(PREFERRED)**, each connection in the process attempts to connect using **CHLNAME(B)**. If a connection is unsuccessful the definition is moved to the end of the list which now becomes CHLNAME(A), CHLNAME(C), CHLNAME(B). Each connection in the process then attempts to connect using **CHLNAME(A)**.

For **AFFINITY(NONE)**, each connection in the process attempts to connect using one of the three definitions selected at random based on the weightings.

When sharing conversations is enabled with a non-zero channel weighting and **AFFINITY(NONE)**, multiple connections in a process using the same queue manager name can connect using different applicable definitions rather than sharing an existing channel instance.

#### **Multi** **AMQPKA(integer)**

The keep alive time for an AMQP channel in milliseconds. If the AMQP client has not sent any frames within the keep alive interval, then the connection is closed with a `amqp:resource-limit-exceeded` AMQP error condition.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of AMQP

#### **BATCHHB(integer)**

Specifies whether batch heartbeats are to be used. The value is the length of the heartbeat in milliseconds.

Batch heartbeats allow a sending channel to verify that the receiving channel is still active just before committing a batch of messages, so that if the receiving channel is not active, the batch can be backed out rather than becoming in-doubt, as would otherwise be the case. By backing out the batch, the messages remain available for processing so they could, for example, be redirected to another channel.

If the sending channel has had a communication from the receiving channel within the batch heartbeat interval, the receiving channel is assumed to be still active. If not, a 'heartbeat' is sent to the receiving channel to check.

The value must be in the range zero through 999999. A value of zero indicates that batch heartbeating is not used.

The **BATCHHB** parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, and CLUSRCVR.

#### **BATCHINT(integer)**

The minimum amount of time, in milliseconds, that a channel keeps a batch open.

The batch is terminated when one of the following conditions is met:

- **BATCHSZ** messages have been sent.
- **BATCHLIM** bytes have been sent.
- The transmission queue is empty and **BATCHINT** is exceeded.

The value must be in the range 0 - 999999999. Zero means that the batch is terminated as soon as the transmission queue becomes empty, or the **BATCHSZ** or **BATCHLIM** limit is reached.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

#### **BATCHLIM(integer)**

The limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point. A sync point is taken after the message that caused the limit to be reached has flowed across the channel. A value of zero in this attribute means that no data limit is applied to batches over this channel.

The batch is terminated when one of the following conditions is met:

- **BATCHSZ** messages have been sent.
- **BATCHLIM** bytes have been sent.
- The transmission queue is empty and **BATCHINT** is exceeded.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.





The value must be in the range 0 - 999999. The default value is 5000.

The **BATCHLIM** parameter is supported on all platforms.

### **BATCHSZ(integer)**

The maximum number of messages that can be sent through a channel before taking a sync point.

The maximum batch size used is the lowest of the following values:

- The **BATCHSZ** of the sending channel.
- The **BATCHSZ** of the receiving channel.
-  On z/OS, three less than the maximum number of uncommitted messages allowed at the sending queue manager (or one if this value is zero or less).
-  On Multiplatforms, the maximum number of uncommitted messages allowed at the sending queue manager (or one if this value is zero or less).
-  On z/OS, three less than the maximum number of uncommitted messages allowed at the receiving queue manager (or one if this value is zero or less).
-  On Multiplatforms, the maximum number of uncommitted messages allowed at the receiving queue manager (or one if this value is zero or less).


The maximum number of uncommitted messages is specified by the **MAXUMSGS** parameter of the **ALTER QMGR** command.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR.

The value must be in the range 1 through 9999.

### **CERTLABL**

Certificate label for this channel to use.

The label identifies which personal certificate in the key repository is sent to the remote peer. If this attribute is blank, the certificate is determined by the queue manager **CERTLABL**, or  on z/OS the **CERTQSG** (if the queue manager is part of a queue sharing group) parameter.

Note that inbound channels (including receiver, requester, cluster-receiver, unqualified server, and server-connection channels) only send the configured certificate if the IBM MQ version of the remote peer fully supports certificate label configuration, and the channel is using a TLS CipherSpec. See [Interoperability of Elliptic Curve and RSA CipherSpecs](#) for further information.

An unqualified server channel is one that does not have the CONNAME field set.

In all other cases, the queue manager **CERTLABL** parameter determines the certificate sent. In particular, all current Java and JMS clients only ever receive the certificate configured by the **CERTLABL** parameter of the queue manager, regardless of the channel-specific label setting

You do not need to run the **REFRESH SECURITY TYPE(SSL)** command if you make any changes to **CERTLABL** on a channel. However, you must run a **REFRESH SECURITY TYPE(SSL)** command if you make any changes to **CERTLABL** on the queue manager.

**Note:** It is an error to inquire, or set, this attribute for cluster-sender channels. If you attempt to do so, you receive the error MQRCCF\_WRONG\_CHANNEL\_TYPE. However, the attribute is present in cluster-sender channel objects (including MQCD structures) and a channel auto-definition (CHAD) exit might set it programmatically if required.



### **channel-name)**

The name of the new channel definition.

This parameter is required on all types of channel.

**Multi** On CLUSSDR channels, it can take a different form from the other channel types. If your convention for naming cluster-sender channels includes the name of the queue manager, you can define a cluster-sender channel using the +QMNAME+ construction. After connection to the matching cluster-receiver channel, IBM MQ substitutes the correct repository queue manager name in place of +QMNAME+ in the cluster-sender channel definition. For more information see [Components of a cluster](#).

The name must not be the same as any existing channel defined on this queue manager (unless **REPLACE** or **ALTER** is specified).

**z/OS** On z/OS, client-connection channel names can duplicate others.

The maximum length of the string is 20 characters, and the string must contain only valid characters; see [Rules for naming IBM MQ objects](#).

### **CHLTYPE**

Channel type. This parameter is required. It must follow immediately after the *channel-name*) parameter on all platforms except z/OS.

#### **SDR**

Sender channel

#### **SVR**

Server channel

#### **RCVR**

Receiver channel

#### **RQSTR**

Requester channel

#### **CLNTCONN**

Client-connection channel

#### **SVRCONN**

Server-connection channel

#### **CLUSSDR**

Cluster-sender channel

#### **CLUSRCVR**

Cluster-receiver channel

**Note:** If you are using the **REPLACE** option, you cannot change the channel type.

### **CLNTWGHT**

The client channel weighting attribute is used so client channel definitions can be selected at random based on their weighting when more than one suitable definition is available. Specify a value in the range 0 - 99.

The special value 0 indicates that no random load balancing is performed and applicable definitions are selected in alphabetical order. To enable random load balancing the value can be in the range 1 through 99 where 1 is the lowest weighting and 99 is the highest.

When a client issues an MQCONN with queue manager name "*\*name*" and more than one suitable definition is available in the CCDT the choice of definition to use is randomly selected based on the weighting with any applicable **CLNTWGHT (0)** definitions selected first in alphabetical order. The distribution is not guaranteed.

For example, suppose the CCDT includes the following two definitions:

```
CHLNAME(TO.QM1) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address1) CLNTWGHT(2)
CHLNAME(TO.QM2) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address2) CLNTWGHT(4)
```

A client MQCONN with queue manager name "\*GRP1" would choose one of the two definitions based on the weighting of the channel definition. (A random integer 1 - 6 would be generated. If the integer was in the range 1 through 2 address1 would be used otherwise address2 would be used). If this connection was unsuccessful the client would then use the other definition.

The CCDT might contain applicable definitions with both zero and non-zero weighting. In this situation, the definitions with zero weightings are chosen first and in alphabetical order. If these connections are unsuccessful the definitions with non-zero weighting are chosen based on their weighting.

For example, suppose the CCDT includes the following four definitions:

```
CHLNAME(TO.QM1) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address1) CLNTWGHT(1)
CHLNAME(TO.QM2) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address2) CLNTWGHT(2)
CHLNAME(TO.QM3) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address3) CLNTWGHT(0)
CHLNAME(TO.QM4) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address4) CLNTWGHT(0)
```

A client MQCONN with queue manager name "\*GRP1" would first choose definition "TO.QM3". If this connection was unsuccessful the client would then choose definition "TO.QM4". If this connection was also unsuccessful the client would then randomly choose one of the remaining two definitions based on their weighting.

**CLNTWGHT** support is added for all supported transport protocols.

### **CLUSNL(*nlname*)**

The name of the namelist that specifies a list of clusters to which the channel belongs.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR and CLUSRCVR channels. Only one of the resultant values of CLUSTER or CLUSNL can be nonblank, the other must be blank.

### **CLUSTER(*clustname*)**

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming IBM MQ objects.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR or CLUSRCVR. Only one of the resultant values of CLUSTER or CLUSNL can be nonblank, the other must be blank.

### **CLWLPRTY(*integer*)**

Specifies the priority of the channel for the purposes of cluster workload distribution. The value must be in the range zero through 9 where zero is the lowest priority and 9 is the highest.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR or CLUSRCVR.

For more information about this attribute, see [CLWLPRTY queue attribute](#).

### **CLWLRANK(*integer*)**

Specifies the rank of the channel for the purposes of cluster workload distribution. The value must be in the range zero through 9 where zero is the lowest rank and 9 is the highest.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR or CLUSRCVR.

For more information about this attribute, see [CLWLRANK channel attribute](#).

### **CLWLWGHT(*integer*)**

Specifies the weighting to be applied to the channel for the purposes of cluster workload distribution so that the proportion of messages sent down the channel can be controlled. The value must be in the range 1 through 99 where 1 is the lowest rank and 99 is the highest.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR or CLUSRCVR.

For more information about this attribute, see [CLWLWGHT channel attribute](#).

## **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command is executed when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

• •

The command is executed on the queue manager on which it was entered.

**qmgr-name**

The command is executed on the queue manager you specify, providing the queue manager is active within the queue sharing group. You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

**COMPHDR**

The list of header data compression techniques supported by the channel. For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The mutually supported compression techniques of the channel are passed to the message exit of the sending channel where the compression technique used can be altered on a per message basis. Compression alters the data passed to send and receive exits.

**NONE**

No header data compression is performed.

**SYSTEM**

Header data compression is performed.

**COMPMSG**

The list of message data compression techniques supported by the channel. For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The mutually supported compression techniques of the channel are passed to the message exit of the sending channel where the compression technique used can be altered on a per message basis. Compression alters the data passed to send and receive exits.

**NONE**


No message data compression is performed.

**RLE**

Message data compression is performed using run-length encoding.

**ZLIBFAST**

Message data compression is performed using ZLIB encoding with speed prioritized.

 On z/OS systems with [zEDC Express](#) facility enabled, compression can be offloaded to zEDC Express.

**ZLIBHIGH**

Message data compression is performed using ZLIB encoding with compression prioritized.

**ANY**

Any compression technique supported by the queue manager can be used. This value is only valid for receiver, requester, and server-connection channels.

**CONNNAME(string)**

Connection name.

For cluster-receiver channels (when specified) **CONNNAME** relates to the local queue manager, and for other channels it relates to the target queue manager.

**z/OS** On z/OS, the maximum length of the string is 48 characters.

**Multi** On Multiplatforms, the maximum length of the string is 264 characters

A workaround to the 48 character limit might be one of the following suggestions:

- Set up your DNS servers so that you use, for example, host name of "myserver" instead of "myserver.location.company.com", ensuring you can use the short host name.
- Use IP addresses.

Specify **CONNNAME** as a comma-separated list of names of machines for the stated **TRPTYPE**. Typically only one machine name is required. You can provide multiple machine names to configure multiple connections with the same properties. The connections are usually tried in the order they are specified in the connection list until a connection is successfully established. The order is modified for clients if the **CLNTWIGHT** attribute is provided. If no connection is successful, the channel attempts the connection again, as determined by the attributes of the channel. With client channels, a connection-list provides an alternative to using queue manager groups to configure multiple connections. With message channels, a connection list is used to configure connections to the alternative addresses of a multi-instance queue manager.

This parameter is required for channels with a channel type (**CHLTYPE**) of SDR, RQSTR, CLNTCONN, and CLUSSDR. It is optional for SVR channels, and for CLUSRCVR channels of **TRPTYPE (TCP)**, and is not valid for RCVR or SVRCONN channels.

**Multi** On Multiplatforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

(1415)

The generated **CONNNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

**Note:** If you are using any of the special characters in your connection name (for example, parentheses) you must enclose the string in single quotation marks.

The value you specify depends on the transport type (**TRPTYPE**) to be used:

## LU 6.2

- **Multi** On Multiplatforms, **CONNNAME** is the name of the CPI-C communications side object. Or, if the **TPNAME** is not blank, **CONNNAME** is the fully qualified name of the partner logical unit.
- **z/OS** On z/OS, there are two forms in which to specify the value:

### Logical unit name

The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. Logical unit name can be specified in one of three forms:

Form	Example
luname	IGY12355
luname/TPname	IGY12345/APING
luname/TPname/modename	IGY12345/APINGD/#INTER

For the first form, the TP name and mode name must be specified for the **TPNAME** and **MODENAME** parameters; otherwise these parameters must be blank.

**Note:** For client-connection channels, only the first form is allowed.

### Symbolic name

The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The **TPNAME** and **MODENAME** parameters must be blank.

**Note:** For cluster-receiver channels, the side information is on the other queue managers in the cluster. Alternatively, in this case it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

The specified or implied LU name can be that of a VTAM generic resources group.

For more information, see [Configuration parameters for an LU 6.2 connection](#).

### NetBIOS

A unique NetBIOS name (limited to 16 characters).

### SPX

The 4 byte network address, the 6 byte node address, and the 2 byte socket number. These values must be entered in hexadecimal, with a period separating the network and node addresses. The socket number must be enclosed in brackets, for example:

```
CONNAME('0a0b0c0d.804abcde23a1(5e86)')
```

### TCP

Either the host name, or the network address of the remote machine (or the local machine for cluster-receiver channels). This address can be followed by an optional port number, enclosed in parentheses.

If the **CONNAME** is a host name, the host name is resolved to an IP address.

The IP stack used for communication depends on both the value specified for **CONNAME** and the value specified for **LOCLADDR**. See [LOCLADDR](#) for information about how this value is resolved.

**z/OS** On z/OS, the connection name can include the IP\_name of an z/OS dynamic DNS group or a Network Dispatcher input port.

**Important:** Do not include the IP\_name or input port for channels with a channel type (**CHLTYPE**) of CLUSSDR.

On all platforms, when you define a channel with a channel type (**CHLTYPE**) of CLUSRCVR that is using TCP/IP, you do not need to specify the network address of your queue manager. IBM MQ generates a **CONNAME** for you, assuming the default port and using the current IPv4 address of the system. If the system does not have an IPv4 address, the current IPv6 address of the system is used.

**Note:** If you are using clustering between IPv6-only and IPv4-only queue managers, do not specify an IPv6 network address as the **CONNAME** for CLUSRCVR channels. A queue manager that is capable only of IPv4 communication is unable to start a cluster sender channel definition that specifies the CONNAME in IPv6 hexadecimal form. Consider, instead, using host names in a heterogeneous IP environment.

### CONVERT

Specifies whether the sending message channel agent attempts conversion of the application message data, if the receiving message channel agent cannot perform this conversion.

#### NO

No conversion by sender

#### YES

Conversion by sender

**z/OS** On z/OS, N and Y are accepted as synonyms of NO and YES.

The **CONVERT** parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

**DEFCDISP**

Specifies the default channel disposition of the channel.

**PRIVATE**

The intended disposition of the channel is as a PRIVATE channel.

**FIXSHARED**

The intended disposition of the channel is as a FIXSHARED channel.

**SHARED**

The intended disposition of the channel is as a SHARED channel.

This parameter does not apply to channels with a channel type (**CHLTYPE**) of CLNTCONN, CLUSSDR, or CLUSRCVR.

**DEFRECON**

Specifies whether a client connection automatically reconnects a client application if its connection breaks.

**NO (default)**

Unless overridden by **MQCONN**, the client is not reconnected automatically.

**YES**

Unless overridden by **MQCONN**, the client reconnects automatically.

**QMGR**

Unless overridden by **MQCONN**, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as MQCNO\_RECONNECT\_Q\_MGR.

**DISABLED**

Reconnection is disabled, even if requested by the client program using the **MQCONN** MQI call.

*Table 130. Automatic reconnection depends on the values set in the application and in the channel definition*

DEFRECON	Reconnection options set in the application			
	MQCNO_RECONNECT	MQCNO_RECONNECT_Q_MGR	MQCNO_RECONNECT_AS_DEF	MQCNO_RECONNECT_DISABLED
NO (default)	YES	QMGR	NO	NO
YES	YES	QMGR	YES	NO
QMGR	YES	QMGR	QMGR	NO
DISABLED	NO	NO	NO	NO

**DESCR(string)**

Plain-text comment. It provides descriptive information about the channel when an operator issues the **DISPLAY CHANNEL** command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

**DISCINT(integer)**

The minimum time in seconds for which the channel waits for a message to arrive on the transmission queue, after a batch ends, before terminating the channel. A value of zero causes the message channel agent to wait indefinitely.

The value must be in the range zero through 999 999.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SVRCONN , SDR, SVR, CLUSSDR, CLUSRCVR.

For SVRCONN channels using the TCP protocol, this parameter is the minimum time in seconds for which the SVRCONN instance remains active without any communication from its partner client. A value of zero disables this disconnect processing. The SVRCONN inactivity interval only applies between IBM MQ API calls from a client, so no client is disconnected during an extended MQGET with wait call. This attribute is ignored for SVRCONN channels using protocols other than TCP.

### **HBINT(integer)**

This attribute specifies the approximate time between heartbeat flows that are to be passed from a sending MCA when there are no messages on the transmission queue.

Heartbeat flows unblock the receiving MCA, which is waiting for messages to arrive or for the disconnect interval to expire. When the receiving MCA is unblocked it can disconnect the channel without waiting for the disconnect interval to expire. Heartbeat flows also free any storage buffers that have been allocated for large messages and close any queues that have been left open at the receiving end of the channel.

The value is in seconds and must be in the range 0 through 999999. A value of zero means that no heartbeat flows are to be sent. The default value is 300. To be most useful, the value needs to be less than the disconnect interval value.

For server-connection and client-connection channels, heartbeats can flow from both the server side as well as the client side independently. If no data has been transferred across the channel for the heartbeat interval, the client-connection MQI agent sends a heartbeat flow and the server-connection MQI agent responds to it with another heartbeat flow. This happens irrespective of the state of the channel, for example, irrespective of whether it is inactive while making an API call, or is inactive waiting for client user input. The server-connection MQI agent is also capable of initiating a heartbeat to the client, again irrespective of the state of the channel. To prevent both server-connection and client-connection MQI agents heart beating to each other at the same time, the server heartbeat is flowed after no data has been transferred across the channel for the heartbeat interval plus 5 seconds.

For more information, see [Heartbeat interval \(HBINT\)](#).

### **KALIVE(integer)**

The value passed to the communications stack for KeepAlive timing for this channel.

For this attribute to be effective, TCP/IP keepalive must be enabled both in the queue manager and in TCP/IP.

**z/OS** On z/OS, you enable TCP/IP keepalive in the queue manager by issuing the **ALTER QMGR TCPKEEP(YES)** command; if the **TCPKEEP** queue manager parameter is **NO**, the value is ignored, and the KeepAlive facility is not used.

**Multi** On [Multiplatforms](#), TCP/IP keepalive is enabled when the **KEEPALIVE=YES** parameter is specified in the TCP stanza in the distributed queuing configuration file, `qm.ini`, or through the IBM MQ Explorer.

Keepalive must also be enabled within TCP/IP itself. Refer to your TCP/IP documentation for information about configuring keepalive:

- **AIX** On AIX, use the **no** command.
- **Windows** On Windows, edit the registry.
- **z/OS** On z/OS, update your TCP/IP PROFILE data set and add or change the **INTERVAL** parameter in the TCPCONFIG section.

**z/OS** Although this parameter is available on all platforms, its setting is implemented only on z/OS.

**Multi** On Multiplatforms, you can access and modify the parameter, but it is only stored and forwarded; there is no functional implementation of the parameter. This functionality is useful in a clustered environment where a value set in a cluster-receiver channel definition on AIX, for example, flows to (and is implemented by) z/OS queue managers that are in, or join, the cluster.

**Multi** On Multiplatforms, if you need the functionality provided by the **KAINT** parameter, use the Heartbeat Interval (**HBINT**) parameter, as described in [HBINT](#).

**(integer)**

The KeepAlive interval to be used, in seconds, in the range 1 through 99 999.

**0**

The value used is that specified by the INTERVAL statement in the TCP profile configuration data set.

**AUTO**

The KeepAlive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated **HBINT** is greater than zero, KeepAlive interval is set to that value plus 60 seconds.
- If the negotiated **HBINT** is zero, the value used is that specified by the INTERVAL statement in the TCP profile configuration data set.

This parameter is valid for all channel types. It is ignored for channels with a **TRPTYPE** other than TCP or SPX.

**LIKE(channel-name)**

The name of a channel. The parameters of this channel are used to model this definition.

If this field is not completed, and you do not complete the parameter fields related to the command, the values are taken from one of the following default channels, depending upon the channel type:

**SYSTEM.DEF.SENDER**

Sender channel

**SYSTEM.DEF.SERVER**

Server channel

**SYSTEM.DEF.RECEIVER**

Receiver channel

**SYSTEM.DEF.REQUESTER**

Requester channel

**SYSTEM.DEF.SVRCONN**

Server-connection channel

**SYSTEM.DEF.CLNTCONN**

Client-connection channel

**SYSTEM.DEF.CLUSSDR**

Cluster-sender channel

**SYSTEM.DEF.CLUSRCVR**

Cluster-receiver channel

This parameter is equivalent to defining the following object for a sender channel, and similarly for other channel types:

```
LIKE(SYSTEM.DEF.SENDER)
```

These default channel definitions can be altered by the installation to the default values required.





On z/OS, the queue manager searches page set zero for an object with the name you specify and a disposition of QMGR or COPY. The disposition of the **LIKE** object is not copied to the object and channel type you are defining.

**Note:**

1. **QSGDISP (GROUP)** objects are not searched.
2. # **LIKE** is ignored if **QSGDISP (COPY)** is specified. However, the group object defined is used as a **LIKE** object.

**LOCLADDR(string)**

**LOCLADDR** is the local communications address for the channel. For channels other than AMQP channels, use this parameter if you want a channel to use a particular IP address, port, or port range for outbound communications. **LOCLADDR** might be useful in recovery scenarios where a channel is restarted on a different TCP/IP stack. **LOCLADDR** is also useful to force a channel to use an IPv4 or IPv6 stack on a dual-stack system. You can also use **LOCLADDR** to force a channel to use a dual-mode stack on a single-stack system.

**Note:** AMQP channels do not support the same format of **LOCLADDR** as other IBM MQ channels. For the format supported by AMQ, see the next parameter **AMQP: LOCLADDR**.

For channels other than AMQP channels, the **LOCLADDR** parameter is valid only for channels with a transport type (**TRPTYPE**) of TCP. If **TRPTYPE** is not TCP, the data is ignored and no error message is issued.

The value is the optional IP address, and optional port or port range used for outbound TCP/IP communications. The format for this information is as follows:

```
LOCLADDR([ip-addr][(low-port[,high-port])][, [ip-addr][(low-port[,high-port])]])
```

The maximum length of **LOCLADDR**, including multiple addresses, is MQ\_LOCAL\_ADDRESS\_LENGTH.

If you omit **LOCLADDR**, a local address is automatically allocated.

Note, that you can set **LOCLADDR** for a C client using the Client Channel Definition Table (CCDT).

All the parameters are optional. Omitting the `ip-addr` part of the address is useful to enable the configuration of a fixed port number for an IP firewall. Omitting the port number is useful to select a particular network adapter without having to identify a unique local port number. The TCP/IP stack generates a unique port number.

Specify `[, [ip-addr][(low-port[,high-port])]]` multiple times for each additional local address. Use multiple local addresses if you want to specify a specific subset of local network adapters. You can also use `[, [ip-addr][(low-port[,high-port])]]` to represent a particular local network address on different servers that are part of a multi-instance queue manager configuration.

**ip-addr**

`ip-addr` is specified in one of three forms:

**IPv4 dotted decimal**

For example, 192.0.2.1

**IPv6 hexadecimal notation**

For example, 2001:DB8:0:0:0:0:0:0

**Alphanumeric host name form**

For example WWW.EXAMPLE.COM

**low-port and high-port**

`low-port` and `high-port` are port numbers enclosed in parentheses.

The following table shows how the **LOCLADDR** parameter can be used:

<i>Table 131. Examples of how the <b>LOCLADDR</b> parameter can be used</i>	
<b>LOCLADDR</b>	<b>Meaning</b>
9.20.4.98	Channel binds to this address locally
9.20.4.98, 9.20.4.99	Channel binds to either IP address. The address might be two network adapters on one server, or a different network adapter on two different servers in a multi-instance configuration.
9.20.4.98(1000)	Channel binds to this address and port 1000 locally
9.20.4.98(1000,2000)	Channel binds to this address and uses a port in the range 1000 - 2000 locally
(1000)	Channel binds to port 1000 locally
(1000,2000)	Channel binds to port in range 1000 - 2000 locally

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, OR CLUSRCVR.

On CLUSSDR channels, the IP address and port to which the outbound channel binds, is a combination of fields. It is a concatenation of the IP address, as defined in the **LOCLADDR** parameter, and the port range from the cluster cache. If there is no port range in the cache, the port range defined in the **LOCLADDR** parameter is used.

**z/OS** This port range does not apply to z/OS systems.

Even though this parameter is similar in form to **CONNAME**, it must not be confused with it. The **LOCLADDR** parameter specifies the characteristics of the local communications, whereas the **CONNAME** parameter specifies how to reach a remote queue manager.

When a channel is started, the values specified for **CONNAME** and **LOCLADDR** determine the IP stack to be used for communication; see [Table 3](#) and [Local Address \( LOCLADDR\)](#).

If the TCP/IP stack for the local address is not installed or configured, the channel does not start and an exception message is generated.

**z/OS** For example, on z/OS systems, the message is "CSQO015E: Command issued but no reply received." The message indicates that the connect() request specifies an interface address that is not known on the default IP stack. To direct the connect() request to the alternative stack, specify the **LOCLADDR** parameter in the channel definition as either an interface on the alternative stack, or a DNS host name. The same specification also works for listeners that might not use the default stack. To find the value to code for **LOCLADDR**, run the **NETSTAT HOME** command on the IP stacks that you want to use as alternatives.

Table 132. How the IP stack to be used for communication is determined

Protocols supported	CONNAME	LOCLADDR	Action of channel
IPv4 only	IPv4 address <sup>1</sup>		Channel binds to IPv4 stack
	IPv6 address <sup>2</sup>		Channel fails to resolve <b>CONNAME</b>
	IPv4 and 6 host name <sup>3</sup>		Channel binds to IPv4 stack
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve <b>CONNAME</b>
	IPv4 and 6 host name	IPv4 address	Channel binds to IPv4 stack
	Any address <sup>4</sup>	IPv6 address	Channel fails to resolve <b>LOCLADDR</b>
	IPv4 address	IPv4 and 6 host name	Channel binds to IPv4 stack
	IPv6 address	IPv4 and 6 host name	Channel fails to resolve <b>CONNAME</b>
	IPv4 and 6 host name	IPv4 and 6 host name	Channel binds to IPv4 stack
IPv4 and IPv6	IPv4 address		Channel binds to IPv4 stack
	IPv6 address		Channel binds to IPv6 stack
	IPv4 and 6 host name		Channel binds to stack determined by <b>IPADDRV</b>
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve <b>CONNAME</b>
	IPv4 and 6 host name	IPv4 address	Channel binds to IPv4 stack
	IPv4 address	IPv6 address	Channel maps <b>CONNAME</b> to IPv6 <sup>5</sup>
	IPv6 address	IPv6 address	Channel binds IPv6 stack
	IPv4 and 6 host name	IPv6 address	Channel binds IPv6 stack
	IPv4 address	IPv4 and 6 host name	Channel binds to IPv4 stack
	IPv6 address	IPv4 and 6 host name	Channel binds to IPv6 stack
IPv4 and 6 host name	IPv4 and 6 host name	Channel binds to stack determined by <b>IPADDRV</b>	

Table 132. How the IP stack to be used for communication is determined (continued)

Protocols supported	CONNAME	LOCLADDR	Action of channel
IPv6 only	IPv4 address		Channel maps <b>CONNAME</b> to IPv6 <sup>5</sup>
	IPv6 address		Channel binds to IPv6 stack
	IPv4 and 6 host name		Channel binds to IPv6 stack
	Any address	IPv4 address	Channel fails to resolve <b>LOCLADDR</b>
	IPv4 address	IPv6 address	Channel maps <b>CONNAME</b> to IPv6 <sup>5</sup>
	IPv6 address	IPv6 address	Channel binds to IPv6 stack
	IPv4 and 6 host name	IPv6 address	Channel binds to IPv6 stack
	IPv4 address	IPv4 and 6 host name	Channel maps <b>CONNAME</b> to IPv6 <sup>5</sup>
	IPv6 address	IPv4 and 6 host name	Channel binds to IPv6 stack
	IPv4 and 6 host name	IPv4 and 6 host name	Channel binds to IPv6 stack

**Notes:**

1. IPv4 address. An IPv4 host name that resolves only to an IPv4 network address or a specific dotted notation IPv4 address, for example 1 . 2 . 3 . 4. This note applies to all occurrences of ' IPv4 address' in this table.
2. IPv6 address. An IPv6 host name that resolves only to an IPv6 network address or a specific hexadecimal notation IPv6 address, for example 4321 : 54bc. This note applies to all occurrences of ' IPv6 address' in this table.
3. IPv4 and 6 host name. A host name that resolves to both IPv4 and IPv6 network addresses. This note applies to all occurrences of ' IPv4 and 6 host name' in this table.
4. Any address. IPv4 address, IPv6 address, or IPv4 and 6 host name. This note applies to all occurrences of 'Any address' in this table.
5. Maps IPv4 **CONNAME** to IPv4 mapped IPv6 address. IPv6 stack implementations that do not support IPv4 mapped IPv6 addressing fail to resolve the **CONNAME**. Mapped addresses might require protocol translators in order to be used. The use of mapped addresses is not recommended.

**AMQP: LOCLADDR(ip-addr)**

**Note:** For the format of **LOCLADDR** that other IBM MQ channels use, see the previous parameter **LOCLADDR**.

For AMQP channels, **LOCLADDR** is the local communications address for the channel. Use this parameter if you want to force the client to use a particular IP address. **LOCLADDR** is also useful to force a channel to use an IPv4 or IPv6 address if a choice is available, or to use a particular network adapter on a system with multiple network adapters.

The maximum length of **LOCLADDR** is MQ\_LOCAL\_ADDRESS\_LENGTH.

If you omit **LOCLADDR**, a local address is automatically allocated.

**ip-addr**

ip-addr is a single network address, specified in one of three forms:

**IPv4 dotted decimal**

For example, 192.0.2.1

**IPv6 hexadecimal notation**

For example, 2001:DB8:0:0:0:0:0:0

**Alphanumeric host name form**

For example, WWW.EXAMPLE.COM

If an IP address is entered, only the address format is validated. The IP address itself is not validated.

**LONGRTY(*integer*)**

When a sender, server, or cluster-sender channel is attempting to connect to the remote queue manager, and the count specified by **SHORTRTY** has been exhausted, this parameter specifies the maximum number of further attempts that are made to connect to the remote queue manager, at intervals specified by **LONGTMR**.

If this count is also exhausted without success, an error is logged to the operator, and the channel is stopped. The channel must then be restarted with a command (it is not started automatically by the channel initiator).

The value must be in the range zero through 999999999.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

**LONGTMR(*integer*)**

For long retry attempts, this parameter is the maximum number of seconds to wait before reattempting connection to the remote queue manager.

The time is approximate; zero means that another connection attempt is made as soon as possible.

The interval between retries might be extended if the channel has to wait to become active.

The value must be in the range zero through 999999999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999,999; values exceeding this maximum are treated as 999,999. Similarly, the minimum retry interval that can be used is 2; values less than this minimum are treated as 2.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

**MAXINST(*integer*)**

The maximum number of simultaneous instances of an individual server-connection channel or AMQP channel that can be started.

The value must be in the range zero through 999999999.

A value of zero prevents all client access on this channel.

If the value of this parameter is reduced to a number that is less than the number of instances of the server-connection channel that are currently running, then those running instances are not affected. However, new instances cannot start until sufficient existing instances have ceased to run so that the number of currently running instances is less than the value of this parameter.

If an AMQP client attempts to connect to an AMQP channel, and the number of connected clients has reached **MAXINST**, the channel closes the connection with a close frame. The close frame contains the following message: `amqp:resource-limit-exceeded`. If a client connects with an ID that is already connected (that is, it performs a client-takeover), and the client is permitted to take over the connection, the takeover will succeed regardless of whether the number of connected clients has reached **MAXINST**.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SVRCONN or AMQP.

**MAXINSTC(integer)**

The maximum number of simultaneous individual server-connection channels that can be started from a single client. In this context, connections that originate from the same remote network address are regarded as coming from the same client.

The value must be in the range zero through 999999999.

A value of zero prevents all client access on this channel.


If the value of this parameter is reduced to a number that is less than the number of instances of the server-connection channel that is currently running from individual clients, then those running instances are not affected. However, new instances from those clients cannot start until sufficient instances have ceased to run that the number of running instances is less than the value of this parameter.


This parameter is valid only for channels with a channel type (**CHLTYPE**) of SVRCONN.

**MAXMSGL(integer)**

Specifies the maximum message length that can be transmitted on the channel. This parameter is compared with the value for the partner and the actual maximum used is the lower of the two values. The value is ineffective if the MQCB function is being executed and the channel type (**CHLTYPE**) is SVRCONN.

The value zero means the maximum message length for the queue manager.

 On Multiplatforms, specify a value in the range zero through to the maximum message length for the queue manager.

 On z/OS, specify a value in the range zero through 104857600 bytes (100 MB).

See the **MAXMSGL** parameter of the **ALTER QMGR** command for more information.

**MCANAME(string)**

Message channel agent name.

This parameter is reserved, and if specified must only be set to blanks (maximum length 20 characters).

**MCAATYPE**

Specifies whether the message-channel-agent program on an outbound message channel runs as a thread or a process.


**PROCESS**


The message channel agent runs as a separate process.

**THREAD**

The message channel agent runs as a separate thread

In situations where a threaded listener is required to service many incoming requests, resources can become strained. In this case, use multiple listener processes and target incoming requests at specific listeners through the port number specified on the listener.

 On Multiplatforms, this parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RQSTR, CLUSSDR, or CLUSRCVR.

 On z/OS, this parameter is supported only for channels with a channel type of CLUSRCVR. When specified in a CLUSRCVR definition, **MCAATYPE** is used by a remote machine to determine the corresponding CLUSSDR definition.

**MCAUSER(string)**

Message channel agent user identifier.

**Note:** An alternative way of providing a user ID for a channel to run under is to use channel authentication records. With channel authentication records, different connections can use the same channel while using different credentials. If both **MCAUSER** on the channel is set and channel authentication records are used to apply to the same channel, the channel authentication records

take precedence. The **MCAUSER** on the channel definition is only used if the channel authentication record uses **USERSRC (CHANNEL)**. For more details, see [Channel authentication records](#).

This parameter interacts with **PUTAUT**, see the definition of that parameter for more information.

If it is nonblank, it is the user identifier that is to be used by the message channel agent for authorization to access IBM MQ resources, including (if **PUTAUT** is DEF) authorization to put the message to the destination queue for receiver or requester channels.

If it is blank, the message channel agent uses its default user identifier.

The default user identifier is derived from the user ID that started the receiving channel. The possible values are:

- ▶ **z/OS** On z/OS, the user ID assigned to the channel-initiator started task by the z/OS started-procedures table.
- ▶ **Multi** For TCP/IP, on [Multiplatforms](#), the user ID from the `inetd.conf` entry, or the user that started the listener.
- ▶ **Multi** For SNA, on [Multiplatforms](#), the user ID from the SNA server entry or, in the absence of this user ID the incoming attach request, or the user that started the listener.
- For NetBIOS or SPX, the user ID that started the listener.

The maximum length of the string is:

- ▶ **Windows** 64 characters on Windows.

For channels with a **CHLTYPE** of AMQP, before IBM MQ 9.2.0, the MCAUSER user ID setting is only supported for user IDs up to 12 characters in length. From IBM MQ 9.2.0, the 12 character limit is removed.

- 12 characters on platforms other than Windows.

▶ **Windows** On Windows, you can optionally qualify a user identifier with the domain name in the format `user@domain`.

This parameter is not valid for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLNTCONN, CLUSSDR.

### **MODENAME(string)**

LU 6.2 mode name (maximum length 8 characters).

This parameter is valid only for channels with a transport type (**TRPTYPE**) of LU 6.2. If **TRPTYPE** is not LU 6.2, the data is ignored and no error message is issued.

If specified, this parameter must be set to the SNA mode name unless the **CONNAME** contains a side-object name, in which case it must be set to blanks. The actual name is then taken from the CPI-C Communications Side Object, or APPC side information data set.

▶ **z/OS** See [Configuration parameters for an LU 6.2 connection](#) for more information about configuration parameters for an LU 6.2 connection for your platform.

This parameter is not valid for channels with a channel type (**CHLTYPE**) of RCVR or SVRCONN.

### **MONCHL**

Controls the collection of online monitoring data for channels:

#### **QMGR**

Collect monitoring data according to the setting of the queue manager parameter MONCHL.

#### **OFF**

Monitoring data collection is turned off for this channel.

**LOW**

If the value of the queue manager **MONCHL** parameter is not NONE, online monitoring data collection is turned on, with a low rate of data collection, for this channel.

**MEDIUM**

If the value of the queue manager **MONCHL** parameter is not NONE, online monitoring data collection is turned on, with a moderate rate of data collection, for this channel.

**HIGH**

If the value of the queue manager **MONCHL** parameter is not NONE, online monitoring data collection is turned on, with a high rate of data collection, for this channel.

For cluster channels, the value of this parameter is not replicated in the repository and, therefore, not used in the auto-definition of cluster-sender channels.

For auto-defined cluster-sender channels, the value of this parameter is taken from the queue manager attribute **MONACLS**. If you want to modify the value, use the command `ALTER QMGR MONACLS(HIGH)`, then restart the auto-defined sender channel.

Changes to this parameter take effect only on channels started after the change occurs.

**MRDATA(string)**

Channel message-retry exit user data. The maximum length is 32 characters.

This parameter is passed to the channel message-retry exit when it is called.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**MREXIT(string)**

Channel message-retry exit name.

The format and maximum length of the name is the same as for MSGEXIT, however you can only specify one message-retry exit.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**MRRTY(integer)**

The number of times the channel tries again before it decides it cannot deliver the message.

This parameter controls the action of the MCA only if the message-retry exit name is blank. If the exit name is not blank, the value of **MRRTY** is passed to the exit to use, but the number of retries performed (if any) is controlled by the exit, and not by this parameter.

The value must be in the range zero through 999999999. A value of zero means that no retries are performed.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**MRTMR(integer)**

The minimum interval of time that must pass before the channel can try the MQPUT operation again. This time interval is in milliseconds.

This parameter controls the action of the MCA only if the message-retry exit name is blank. If the exit name is not blank, the value of **MRTMR** is passed to the exit to use, but the retry interval is controlled by the exit, and not by this parameter.

The value must be in the range zero through 999 999 999. A value of zero means that the retry is performed as soon as possible (if the value of **MRRTY** is greater than zero).

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**MSGDATA(string)**

User data for the channel message exit. The maximum length is 32 characters.



This data is passed to the channel message exit when it is called.

**ALW** On AIX, Linux, and Windows, you can specify data for more than one exit program by specifying multiple strings separated by commas. The total length of the field must not exceed 999 characters.

**IBM i** On IBM i, you can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

**z/OS** On z/OS, you can specify up to eight strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

On other platforms, you can specify only one string of message exit data for each channel.

**Note:** This parameter is accepted but ignored for server-connection and client-connection channels.

### MSGEXIT(*string*)

Channel message exit name.

If this name is nonblank, the exit is called at the following times:

- Immediately after a message has been retrieved from the transmission queue (sender or server), or immediately before a message is put to a destination queue (receiver or requester).

The exit is given the entire application message and transmission queue header for modification.

- At initialization and termination of the channel.

**ALW** On AIX, Linux, and Windows, you can specify the name of more than one exit program by specifying multiple strings separated by commas. However, the total number of characters specified must not exceed 999.

**IBM i** On IBM i, you can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

**z/OS** On z/OS, you can specify the names of up to eight exit programs by specifying multiple strings separated by commas.

On other platforms, you can specify only one message exit name for each channel.

For channels with a channel type (**CHLTYPE**) of CLNTCONN or SVRCONN, this parameter is accepted but ignored, because message exits are not invoked for such channels.

The format and maximum length of the name depends on the environment:

- **Linux** **AIX** On AIX and Linux, it is of the form:

```
libraryname(functionname)
```

The maximum length of the string is 128 characters.

- **Windows** On Windows, it is of the form:

```
dllname(functionname)
```

where *dllname* is specified without the suffix .DLL. The maximum length of the string is 128 characters.

- **IBM i** On IBM i, it is of the form:

```
progrname libname
```

where *progrname* occupies the first 10 characters and *libname* the second 10 characters (both padded to the right with blanks if necessary). The maximum length of the string is 20 characters.

- ▶ **z/OS** On z/OS, it is a load module name, maximum length 8 characters (128 characters are allowed for exit names for client-connection channels, subject to a maximum total length including commas of 999).

### **NETPRTY(integer)**

The priority for the network connection. Distributed queuing chooses the path with the highest priority if there are multiple paths available. The value must be in the range zero through 9; zero is the lowest priority.

This parameter is valid only for CLUSRCVR channels.

### **NPMSPEED**

The class of service for nonpersistent messages on this channel:

#### **FAST**

Fast delivery for nonpersistent messages; messages might be lost if the channel is lost. Messages are retrieved using MQGMO\_SYNCPOINT\_IF\_PERSISTENT and so are not included in the batch unit of work.

#### **NORMAL**

Normal delivery for nonpersistent messages.

If the sending side and the receiving side do not agree about this parameter, or one does not support it, NORMAL is used.

#### **Notes:**

1. If the active recovery logs for IBM MQ for z/OS are switching and archiving more frequently than expected, given that the messages being sent across a channel are non-persistent, setting NPMSPEED(FAST) on both the sending and receiving ends of the channel can minimize the SYSTEM.CHANNEL.SYNCQ updates.
2. If you are seeing high CPU usage relating to updates to the SYSTEM.CHANNEL.SYNCQ, setting NPMSPEED(FAST) can significantly reduce the CPU usage.

This parameter is valid only for channels with a **CHLTYPE** of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR.

### **PASSWORD(string)**

Password used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent. The maximum length is 12 characters.

▶ **Multi** On Multiplatforms, this parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RQSTR, CLNTCONN, or CLUSSDR.

▶ **z/OS** On z/OS, it is supported only for channels with a channel type (**CHLTYPE**) of CLNTCONN.

Although the maximum length of the parameter is 12 characters, only the first 10 characters are used.

### **PORT(integer)**

The port number used to connect an AMQP channel. The default port for AMQP 1.0 connections is 5672. If you are already using port 5672, you can specify a different port.

### **PROPCTL**

Property control attribute.

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor).

This parameter is applicable to Sender, Server, Cluster Sender, and Cluster Receiver channels.

This parameter is optional.

Permitted values are:

## COMPAT

COMPAT allows applications which expect JMS-related properties to be in an MQRFH2 header in the message data to continue to work unmodified.

Table 133. Range of results, depending on which message properties are set, when PROPCTL value is COMPAT

Message properties	Result
The message contains a property with a prefix of mcd., jms., usr. or mqext.	All optional message properties (where the <b>Support</b> value is MQPD_SUPPORT_OPTIONAL), except properties in the message descriptor or extension, are placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.
The message does not contain a property with a prefix of mcd., jms., usr. or mqext.	All message properties, except properties in the message descriptor or extension, are removed from the message before the message is sent to the remote queue manager.
The message contains a property where the <b>Support</b> field of the property descriptor is not set to MQPD_SUPPORT_OPTIONAL	The message is rejected with reason MQRC_UNSUPPORTED_PROPERTY and treated in accordance with its report options.
The message contains one or more properties where the <b>Support</b> field of the property descriptor is set to MQPD_SUPPORT_OPTIONAL but other fields of the property descriptor are set to non-default values.	The properties with non-default values are removed from the message before the message is sent to the remote queue manager.
The MQRFH2 folder that would contain the message property needs to be assigned with the <i>content='properties'</i> attribute	The properties are removed to prevent MQRFH2 headers with unsupported syntax flowing to a V6 or prior queue manager.

### NONE

All properties of the message, except properties in the message descriptor or extension, are removed from the message before the message is sent to the remote queue manager.

If the message contains a property where the **Support** field of the property descriptor is not set to MQPD\_SUPPORT\_OPTIONAL then the message is rejected with reason MQRC\_UNSUPPORTED\_PROPERTY and treated in accordance with its report options.

### ALL


All properties of the message are included with the message when it is sent to the remote queue manager. The properties, except properties in the message descriptor (or extension), are placed in one or more MQRFH2 headers in the message data.

### PUTAUT

Specifies which user identifiers are used to establish authority to put messages to the destination queue (for messages channels) or to execute an MQI call (for MQI channels).


### DEF

The default user ID is used.

 On z/OS, DEF might involve using both the user ID received from the network and that derived from **MCAUSER**.

### CTX

The user ID from the *UserIdentifier* field of the message descriptor is used.

 On z/OS, CTX might involve also using the user ID received from the network or that derived from **MCAUSER**, or both.

**z/OS ONLYMCA**

The user ID derived from **MCAUSER** is used. Any user ID received from the network is not used. This value is supported only on z/OS.

**z/OS ALTMCA**

The user ID from the *UserIdentifier* field of the message descriptor is used. Any user ID received from the network is not used. This value is supported only on z/OS.

**z/OS** On z/OS, the user IDs that are checked, and how many user IDs are checked, depends on the setting of the MQADMIN RACF® class hlq.RESLEVEL profile. Depending on the level of access the user ID of the channel initiator has to hlq.RESLEVEL, zero, one or two user IDs are checked. To see how many user IDs are checked, see RESLEVEL and channel initiator connections. For more information about which user IDs are checked, see [User IDs used by the channel initiator](#).

**z/OS** On z/OS, this parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, CLUSRCVR, or SVRCONN. CTX and ALTMCA are not valid for SVRCONN channels.

**Multi** On Multiplatforms, this parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**QMNAME(string)**

Queue manager name.

For channels with a channel type (**CHLTYPE**) of CLNTCONN, this parameter is the name of a queue manager to which an application that is running in a client environment and using the client channel definition table can request connection. This parameter need not be the name of the queue manager on which the channel is defined, to allow a client to connect to different queue managers.

For channels of other types, this parameter is not valid.

**z/OS QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

<i>Table 134. Behavior for each of the QSGDISP values</i>	
<b>QSGDISP</b>	<b>ALTER</b>
COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (COPY)</b> . Any object residing in the shared repository, or any object defined using a command that had the parameters <b>QSGDISP (QMGR)</b> , is not affected by this command.
GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameters <b>QSGDISP (GROUP)</b>. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command. If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE CHANNEL(channel-name) CHLTYPE(type) REPLACE QSGDISP(COPY)</pre> <p>The <b>ALTER</b> for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>

Table 134. Behavior for each of the QSGDISP values (continued)

QSGDISP	ALTER
PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with <b>QSGDISP(QMGR)</b> or <b>QSGDISP(COPY)</b> . Any object residing in the shared repository is unaffected.
QMGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP(QMGR)</b> . Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

### RCVDATA(string)

Channel receive exit user data (maximum length 32 characters).

This parameter is passed to the channel receive exit when it is called.

**ALW** On AIX, Linux, and Windows, you can specify data for more than one exit program by specifying multiple strings separated by commas. The total length of the field must not exceed 999 characters.

**IBM i** On IBM i, you can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first receive exit specified, the second string to the second exit, and so on.

**z/OS** On z/OS, you can specify up to eight strings, each of length 32 characters. The first string of data is passed to the first receive exit specified, the second string to the second exit, and so on.

On other platforms, you can specify only one string of receive exit data for each channel.

### RCVEXIT(string)

Channel receive exit name.

If this name is nonblank, the exit is called at the following times:

- Immediately before the received network data is processed.

The exit is given the complete transmission buffer as received. The contents of the buffer can be modified as required.

- At initialization and termination of the channel.

**ALW** On AIX, Linux, and Windows, you can specify the name of more than one exit program by specifying multiple strings separated by commas. However, the total number of characters specified must not exceed 999.

**IBM i** On IBM i, you can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

**z/OS** On z/OS, you can specify the names of up to eight exit programs by specifying multiple strings separated by commas.

On other platforms, you can specify only one receive exit name for each channel.

The format and maximum length of the name is the same as for **MSGEXIT**.

### REPLACE and NOREPLACE

Whether the existing definition **z/OS** (and on z/OS, with the same disposition) is to be replaced with this one. This parameter is optional. Any object with a different disposition is not changed.

#### REPLACE

The definition replaces any existing definition of the same name. If a definition does not exist, one is created. REPLACE does not alter the channel status.

## **NOREPLACE**

The definition does not replace any existing definition of the same name.

## **SCYDATA(string)**

Channel security exit user data (maximum length 32 characters).

This parameter is passed to the channel security exit when it is called.

## **SCYEXIT(string)**

Channel security exit name.

If this name is nonblank, the exit is called at the following times:

- Immediately after establishing a channel.

Before any messages are transferred, the exit is able to instigate security flows to validate connection authorization.

- Upon receipt of a response to a security message flow.

Any security message flows received from the remote processor on the remote queue manager are given to the exit.

- At initialization and termination of the channel.

The format and maximum length of the name is the same as for **MSGEXIT** but only one name is allowed.

## **SENDDATA(string)**

Channel send exit user data. The maximum length is 32 characters.

This parameter is passed to the channel send exit when it is called.

**ALW** On AIX, Linux, and Windows, you can specify data for more than one exit program by specifying multiple strings separated by commas. The total length of the field must not exceed 999 characters.

**IBM i** On IBM i, you can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first send exit specified, the second string to the second exit, and so on.

**z/OS** On z/OS, you can specify up to eight strings, each of length 32 characters. The first string of data is passed to the first send exit specified, the second string to the second exit, and so on.

On other platforms, you can specify only one string of send exit data for each channel.

## **SENDEXIT(string)**

Channel send exit name.

If this name is nonblank, the exit is called at the following times:

- Immediately before data is sent out on the network.

The exit is given the complete transmission buffer before it is transmitted. The contents of the buffer can be modified as required.

- At initialization and termination of the channel.

**ALW** On AIX, Linux, and Windows, you can specify the name of more than one exit program by specifying multiple strings separated by commas. However, the total number of characters specified must not exceed 999.

**IBM i** On IBM i, you can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

**z/OS** On z/OS, you can specify the names of up to eight exit programs by specifying multiple strings separated by commas.

On other platforms, you can specify only one send exit name for each channel.

The format and maximum length of the name is the same as for **MSGEXIT**.

### **SEQWRAP**(integer)

When this value is reached, sequence numbers wrap to start again at 1.

This value is nonnegotiable and must match in both the local and remote channel definitions.

The value must be in the range 100 through 999999999.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR.

### **SHARECNV**(integer)

Specifies the maximum number of conversations that can be sharing each TCP/IP channel instance. A **SHARECNV** value of:

**1**

Specifies no sharing of conversations over a TCP/IP channel instance. Client heartbeating is available whether in an MQGET call or not. Read ahead and client asynchronous consumption are also available, and channel quiescing is more controllable.

**0**

Specifies no sharing of conversations over a TCP/IP channel instance.

The value must be in the range zero through 999999999.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLNTCONN or SVRCONN. If the client-connection **SHARECNV** value does not match the server-connection **SHARECNV** value, the lower of the two values is used. This parameter is ignored for channels with a transport type (**TRPTYPE**) other than TCP.

All the conversations on a socket are received by the same thread.

High **SHARECNV** limits have the advantage of reducing queue manager thread usage. However, if many conversations sharing a socket are all busy, there is a possibility of delays as the conversations contend with one another to use the receiving thread. In this situation, a lower **SHARECNV** value is better.

The number of shared conversations does not contribute to the **MAXINST** or **MAXINSTC** totals.

**Note:** You should restart the client for this change to take effect.

### **SHORTRTY**(integer)

The maximum number of attempts that are made by a sender, server, or cluster-sender channel to connect to the remote queue manager, at intervals specified by **SHORTTMR**, before the (normally longer) **LONGRTY** and **LONGTMR** are used.

Retry attempts are made if the channel fails to connect initially (whether it is started automatically by the channel initiator or by an explicit command), and also if the connection fails after the channel has successfully connected. However, if the cause of the failure is such that more attempts are unlikely to be successful, they are not attempted.

The value must be in the range zero through 999999999.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

### **SHORTTMR**(integer)

For short retry attempts, this parameter is the maximum number of seconds to wait before reattempting connection to the remote queue manager.

The time is approximate; zero means that another connection attempt is made as soon as possible.

The interval between retries might be extended if the channel has to wait to become active.

The value must be in the range zero through 999999999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this maximum are treated as 999999. Similarly, the minimum retry interval that can be used is 2; values less than this minimum are treated as 2.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

### **SPLPROT**

SPLPROT (Security Policy Protection) specifies how a server-to-server Message Channel Agent should deal with message protection when AMS is active and an applicable policy exists.

This parameter applies to z/OS only, from IBM MQ 9.1.3 onwards.

The permitted values are:

#### **PASSTHRU**

Pass through, unchanged, any messages sent or received by the message channel agent for this channel.

This value is valid for channels with a channel type (**CHLTYPE**) of SDR, SVR, RCVR, or RQSTR, and is the default value.

#### **REMOVE**

Remove any AMS protection from messages retrieved from the transmission queue by the message channel agent, and send the messages to the partner.

When the message channel agent gets a message from the transmission queue, if an AMS policy is defined for the transmission queue, it is applied to remove any AMS protection from the message prior to sending the message across the channel. If an AMS policy is not defined for the transmission queue, the message is sent as is.

This value is valid only for channels with a channel type of SDR or SVR.

#### **ASPOLICY**

Based on the policy defined for the target queue, apply AMS protection to inbound messages prior to putting them on to the target queue.

When the message channel agent receives an inbound message, if an AMS policy is defined for the target queue, AMS protection is applied to the message prior to the message being put to the target queue. If an AMS policy is not defined for the target queue, the message is put to the target queue as is.

This value is valid only for channels with a channel type of RCVR or RQSTR.

### **SSLCAUTH**

Defines whether IBM MQ requires a certificate from the TLS client. The initiating end of the channel acts as the TLS client, so this parameter applies to the end of the channel that receives the initiation flow, which acts as the TLS server.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, SVRCONN, CLUSRCVR, SVR, or RQSTR.

The parameter is used only for channels with **SSLCIPH** specified. If **SSLCIPH** is blank, the data is ignored and no error message is issued.

#### **REQUIRED**

IBM MQ requires and validates a certificate from the TLS client.



#### **OPTIONAL**

The peer TLS client system might still send a certificate. If it does, the contents of this certificate are validated as normal.

### **SSLCIPH(string)**

Specifies the CipherSpec that is used on the channel. The maximum length is 32 characters.



**Attention:**   On IBM MQ for z/OS, you can also specify the four digit hexadecimal code of a CipherSpec, whether or not it appears in the following table. On IBM i, you can also specify the two digit hexadecimal code of a CipherSpec, whether or not it appears



in the following table. Also, on IBM i, installation of AC3 is a prerequisite for the use of TLS. You should not specify hexadecimal cipher values in SSLCIPH, because it is unclear from the value which cipher will be used, and the choice of which protocol to be used is indeterminate. Using hexadecimal cipher values can lead to CipherSpec mismatch errors.

The **SSLCIPH** values must specify the same CipherSpec on both ends of the channel.

This parameter is valid on all channel types that use transport type **TRPTYPE(TCP)**. If the parameter is blank, no attempt is made to use TLS on the channel.


**Multi** **V 9.3.0** If **SecureCommsOnly** is enabled, plain text communication is not supported and the channel fails to start.

The value for this parameter is also used to set the value of SECPROT, which is an output field on the **DISPLAY CHSTATUS** command.

**Note:** When **SSLCIPH** is used with a telemetry channel, it means TLS Cipher Suite. See the **SSLCIPH** description for **DEFINE CHANNEL (MQTT)**.

**z/OS** **ALW** From IBM MQ 9.1.1, you can specify a value of ANY\_TLS12, which represents a subset of acceptable CipherSpecs that use the TLS 1.2 protocol; these CipherSpecs are listed in the following table.

**ALW** From IBM MQ 9.1.4, on AIX, Linux, and Windows, IBM MQ provides an expanded set of alias CipherSpecs that includes ANY\_TLS12\_OR\_HIGHER, and ANY\_TLS13\_OR\_HIGHER. These alias CipherSpecs are listed in the following table.

 **Attention:** If your enterprise needs to guarantee that a certain CipherSpec is negotiated and used, you must not use an alias CipherSpec value such as ANY\_TLS12.

For information on changing your existing security configurations to use the ANY\_TLS12\_OR\_HIGHER CipherSpec, see [Migrating existing security configurations to use the ANY\\_TLS12\\_OR\\_HIGHER CipherSpec](#).

Platform support "1" on page 307	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 307	Suite B
<b>Alias CipherSpecs</b>							
All	ANY_TLS13_OR_HIGHER "3" on page 307 "4" on page 307	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS13 "4" on page 307 "5" on page 307	N/A	TLS 1.3	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS12_OR_HIGHER "4" on page 307 "6" on page 307	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS12 "7" on page 307	N/A	TLS 1.2	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY "8" on page 307	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
<b>CipherSpecs for TLS 1.3</b>							
All	TLS_AES_128_GCM_SHA256	1301	TLS 1.3	GCM	AES-128 with GCM (128)	Yes	No

Table 135. CipherSpecs you can use with IBM MQ TLS support (continued)











Platform support "1" on page 307	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 307	Suite B
All	TLS_AES_256_GCM_SHA384	1302	TLS 1.3	GCM	AES-256 with GCM (256)	Yes	No
All	TLS_CHACHA20_POLY1305_SHA256	1303	TLS 1.3	POLY1305	CHACHA20 (256)	No	No
 ALW	TLS_AES_128_CCM_SHA256	1304	TLS 1.3	CBC-MAC	AES-128 with CTR (128)	Yes	No
 ALW	TLS_AES_128_CCM_8_SHA256 <a href="#">"10" on page 307</a>	1305	TLS 1.3	CBC-MAC	AES-128 with CTR (128)	Yes	No
<b>CipherSpecs for TLS 1.2</b>							
All	TLS_RSA_WITH_AES_128_CBC_SHA256 <a href="#">"9" on page 307</a>	003C	TLS 1.2	SHA-256	AES (128)	Yes	No
All	TLS_RSA_WITH_AES_256_CBC_SHA256 <a href="#">"9" on page 307</a> <a href="#">"11" on page 307</a>	003D	TLS 1.2	SHA-256	AES (256)	Yes	No
All	TLS_RSA_WITH_AES_128_GCM_SHA256 <a href="#">"9" on page 307</a> <a href="#">"12" on page 307</a>	009C	TLS 1.2	SHA-256 and AEAD GCM	AES (128)	Yes	No
All	TLS_RSA_WITH_AES_256_GCM_SHA384 <a href="#">"9" on page 307</a> <a href="#">"11" on page 307</a> <a href="#">"12" on page 307</a>	009D	TLS 1.2	SHA-384 and AEAD GCM	AES (256)	Yes	No
All	ECDHE_ECDSA_AES_128_CBC_SHA256 <a href="#">"9" on page 307</a>	C023	TLS 1.2	SHA-256	AES (128)	Yes	No
All	ECDHE_ECDSA_AES_256_CBC_SHA384 <a href="#">"9" on page 307</a> <a href="#">"11" on page 307</a>	C024	TLS 1.2	SHA-384	AES (256)	Yes	No
All	ECDHE_RSA_AES_128_CBC_SHA256 <a href="#">"9" on page 307</a>	C027	TLS 1.2	SHA-256	AES (128)	Yes	No
All	ECDHE_RSA_AES_256_CBC_SHA384 <a href="#">"9" on page 307</a> <a href="#">"11" on page 307</a>	C028	TLS 1.2	SHA-384	AES (256)	Yes	No
 Multi	ECDHE_ECDSA_AES_128_GCM_SHA256 <a href="#">"11" on page 307</a> <a href="#">"12" on page 307</a>	C02B	TLS 1.2	SHA-256 and AEAD GCM	AES (SHA384)	Yes	128 bit
 Multi	ECDHE_ECDSA_AES_256_GCM_SHA384 <a href="#">"11" on page 307</a> <a href="#">"12" on page 307</a>	C02C	TLS 1.2	SHA-384 and AEAD GCM	AES (SHA384)	Yes	192 bit
All	ECDHE_RSA_AES_128_GCM_SHA256 <a href="#">"12" on page 307</a>	C02F	TLS 1.2	SHA-256 and AEAD GCM	AES (128)	Yes	No

Table 135. CipherSpecs you can use with IBM MQ TLS support (continued)

Platform support "1" on page 307	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 307	Suite B
All	ECDHE_RSA_AES_256_GCM_SHA384 <small>"11" on page 307 "12" on page 307</small>	C030	TLS 1.2	AEAD AES-128 GCM	AES (SHA384)	Yes	No

**Notes:**

- For a list of platforms covered by each platform icon, see [Icons used in the product documentation](#).
- Specifies whether the CipherSpec is FIPS-certified on a FIPS-certified platform. See [Federal Information Processing Standards \(FIPS\)](#) for an explanation of FIPS.
-  The ANY\_TLS13\_OR\_HIGHER alias CipherSpec negotiates the highest level of security that the remote end will allow but will only connect using a TLS 1.3 or higher protocol.
-  To use TLS 1.3, or the ANY CipherSpec, on IBM i the underlying operating system version must support TLS 1.3. See [System TLS support for TLSv1.3](#) for more information.
-  The ANY\_TLS13 alias CipherSpec represents a subset of acceptable CipherSpecs that use the TLS 1.3 protocol, as listed in this table for each platform.
-  The ANY\_TLS12\_OR\_HIGHER alias CipherSpec negotiates the highest level of security that the remote end will allow but will only connect using a TLS 1.2 or higher protocol.
- The ANY\_TLS12 CipherSpec represents a subset of acceptable CipherSpecs that use the TLS 1.2 protocol, as listed in this table for each platform.
-  The ANY alias CipherSpec negotiates the highest level of security that the remote end will allow.
-  These CipherSpecs are not enabled on IBM i 7.4 systems that have System Value QSSLCSLCTL set to \*OPSSYS.
-  These CipherSpecs use an 8-octet Integrity Check Value (ICV) instead of a 16-octet ICV.
- This CipherSpec cannot be used to secure a connection from the IBM MQ Explorer to a queue manager unless the appropriate unrestricted policy files are applied to the JRE used by the Explorer.
-  Following a recommendation by GSKit, TLS 1.2 GCM CipherSpecs have a restriction which means that after 2<sup>24.5</sup> TLS records are sent, using the same session key, the connection is terminated with message [AMQ9288E](#). This GCM restriction is active, regardless of the FIPS mode being used.

To prevent this error from happening, avoid using TLS 1.2 GCM Ciphers, enable secret key reset, or start your IBM MQ queue manager or client with the environment variable GSK\_ENFORCE\_GCM\_RESTRICTION=GSK\_FALSE set. For GSKit libraries, you must set this environment variable on both sides of the connection, and apply it to both client to queue manager connections and queue manager to queue manager connections. Note that this setting affects unmanaged .NET clients, but not Java or managed .NET clients. For more information, see [AES-GCM cipher restriction](#).

This restriction does not apply to IBM MQ for z/OS.

For more information about CipherSpecs, see [Enabling CipherSpecs](#).

When you request a personal certificate, you specify a key size for the public and private key pair. The key size that is used during the SSL handshake can depend on the size stored in the certificate and on the CipherSpec:

- **ALW** **z/OS** On z/OS, AIX, Linux, and Windows, when a CipherSpec name includes `_EXPORT`, the maximum handshake key size is 512 bits. If either of the certificates exchanged during the SSL handshake has a key size greater than 512 bits, a temporary 512-bit key is generated for use during the handshake.
- **z/OS** For z/OS, System SSL state that if a TLS V1.3 connection is being negotiated:
  - The minimum key size for an RSA peer certificate is the larger of the following two values: 2048, or the value specified in the `GSK_PEER_RSA_MIN_KEY_SIZE` attribute.
  - The minimum key size for an ECC peer certificate is the larger of the following two values: 256, or the value specified in the `GSK_PEER_ECC_MIN_KEY_SIZE` attribute.
- **ALW** On AIX, Linux, and Windows, when a CipherSpec name includes `_EXPORT1024`, the handshake key size is 1024 bits.
- Otherwise the handshake key size is the size stored in the certificate.

### SSLPEER(string)

Specifies the filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel. (A Distinguished Name is the identifier of the TLS certificate.) If the Distinguished Name in the certificate received from the peer does not match the **SSLPEER** filter, the channel does not start.

**Note:** An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both **SSLPEER** on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns in order to connect. For more information, see [Channel authentication records](#).

This parameter is optional; if it is not specified, the Distinguished Name of the peer is not checked at channel startup. (The Distinguished Name from the certificate is still written into the **SSLPEER** definition held in memory, and passed to the security exit). If **SSLCIPH** is blank, the data is ignored and no error message is issued.

This parameter is valid for all channel types.

The **SSLPEER** value is specified in the standard form used to specify a Distinguished Name. For example:

```
SSLPEER( ' SERIALNUMBER=4C:D0:49:D5:02:5F:38,CN="H1_C_FR1",O=IBM,C=GB' )
```

You can use a semi-colon as a separator instead of a comma.

The possible attribute types supported are:

Summary attribute	Description
SERIALNUMBER	Certificate serial number
MAIL	Email address
<b>Deprecated</b> E	Email address (Deprecated in preference to MAIL)
UID or USERID	User identifier
CN	Common Name
T	Title
OU	Organizational Unit name

Table 136. Attribute types supported by SSLPEER (continued)

Summary attribute	Description
DC	Domain component
O	Organization name
STREET	Street / First line of address
L	Locality name
ST (or SP or S)	State or Province name
PC	Postal code / zip code
C	Country
UNSTRUCTUREDNAME	Host name
UNSTRUCTUREDADDRESS	IP address
DNQ	Distinguished name qualifier

IBM MQ accepts only uppercase letters for the attribute types.

If any of the unsupported attribute types are specified in the **SSLPEER** string, an error is output either when the attribute is defined or at run time (depending on which platform you are running on), and the string is deemed not to have matched the Distinguished Name of the flowed certificate.

If the Distinguished Name of the flowed certificate contains multiple OU (organizational unit) attributes, and **SSLPEER** specifies these attributes to be compared, they must be defined in descending hierarchical order. For example, if the Distinguished Name of the flowed certificate contains the OUs OU=Large Unit, OU=Medium Unit, OU=Small Unit, specifying the following **SSLPEER** values works:

```
('OU=Large Unit,OU=Medium Unit')
('OU=*,OU=Medium Unit,OU=Small Unit')
('OU=*,OU=Medium Unit')
```

but specifying the following **SSLPEER** values fails:

```
('OU=Medium Unit,OU=Small Unit')
('OU=Large Unit,OU=Small Unit')
('OU=Medium Unit')
('OU=Small Unit, Medium Unit, Large Unit')
```

As indicated in these examples, attributes at the low end of the hierarchy can be omitted. For example, ('OU=Large Unit,OU=Medium Unit') is equivalent to ('OU=Large Unit,OU=Medium Unit,OU=\*')

If two DNs are equal in all respects except for their DC values, the same matching rules apply as for OUs except that in DC values the left-most DC is the lowest-level (most specific) and the comparison ordering differs accordingly.

Any or all the attribute values can be generic, either an asterisk (\*) on its own, or a stem with initiating or trailing asterisks. Asterisks allow the **SSLPEER** to match any Distinguished Name value, or any value starting with the stem for that attribute.

If an asterisk is specified at the beginning or end of any attribute value in the Distinguished Name on the certificate, you can specify '\\*' to check for an exact match in **SSLPEER**. For example, if you have an attribute of CN='Test\*' in the Distinguished Name of the certificate, you can use the following command:

```
SSLPEER('CN=Test\*')
```

 The maximum length of the parameter is 1024 bytes on AIX, Linux, and Windows.

**IBM i** The maximum length of the parameter is 1024 bytes on IBM i.

**z/OS** The maximum length of the parameter is 256 bytes on z/OS.

Channel [authentication records](#) provide greater flexibility when using **SSLPEER** and support 1024 bytes on all platforms.

## STATCHL

Controls the collection of statistics data for channels:

### QMGR

The value of the **STATCHL** parameter of the queue manager is inherited by the channel.

### OFF

Statistics data collection is turned off for this channel.

### LOW

If the value of the **STATCHL** parameter of the queue manager is not NONE, statistics data collection is turned on, with a low rate of data collection, for this channel.

### MEDIUM

If the value of the **STATCHL** parameter of the queue manager is not NONE, statistics data collection is turned on, with a moderate rate of data collection, for this channel.

### HIGH

If the value of the **STATCHL** parameter of the queue manager is not NONE, statistics data collection is turned on, with a high rate of data collection, for this channel.

Changes to this parameter take effect only on channels started after the change occurs.

**z/OS** On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

For cluster channels, the value of this parameter is not replicated in the repository and used in the auto-definition of cluster-sender channels. For auto-defined cluster-sender channels, the value of this parameter is taken from the attribute **STATACLS** of the queue manager. This value might then be overridden in the channel auto-definition exit.

## **Multi** **V 9.3.0** **TMPMODEL(string)**

The name of the model queue to be used while creating a temporary queue (maximum length 48 characters).

The default queue is SYSTEM.DEFAULT.MODEL.QUEUE.

## **Multi** **V 9.3.0** **TMPQPRFX(string)**

The temporary queue name prefix to add to the beginning of the model queue when deriving a temporary queue name (maximum length 48 characters).

The default is AMQP.\*.

## **TPNAME(string)**

LU 6.2 transaction program name (maximum length 64 characters).

This parameter is valid only for channels with a transport type (**TRPTYPE**) of LU 6.2.

Set this parameter to the SNA transaction program name, unless the **CONNAME** contains a side-object name in which case set it to blanks. The actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set.

**z/OS** See [Configuration parameters for an LU 6.2 connection](#) for more information about configuration parameters for an LU 6.2 connection for your platform.

**Windows** **z/OS** On Windows SNA Server, and in the side object on z/OS, the **TPNAME** is wrapped to uppercase.

This parameter is not valid for channels with a channel type (**CHLTYPE**) of RCVR.

### **TPROOT**

The topic root for an AMQP channel. The default value for **TPROOT** is SYSTEM.BASE.TOPIC. With this value, the topic string an AMQP client uses to publish or subscribe has no prefix, and the client can exchange messages with other IBM MQ publish/subscribe applications. To have AMQP clients publish and subscribe under a topic prefix, first create an IBM MQ topic object with a topic string set to the prefix you want, then set **TPROOT** to the name of the IBM MQ topic object you created.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of AMQP

### **TRPTYPE**

Transport type to be used.

On all platforms, this parameter is optional because, if you do not enter a value, the value specified in the SYSTEM.DEF.*channel-type* definition is used. However, no check is made that the correct transport type has been specified if the channel is initiated from the other end.

**z/OS** On z/OS, if the SYSTEM.DEF.*channel-type* definition does not exist, the default is LU62.

This parameter is required on all other platforms.

### **LU62**

SNA LU 6.2

### **NETBIOS**

**Windows** NetBIOS (supported only on Windows, and DOS).

**z/OS** This attribute also applies to z/OS for defining client-connection channels that connect to servers on the platforms supporting NetBIOS.

### **SPX**

**Windows** Sequenced packet exchange (supported only on Windows, and DOS).

**z/OS** This attribute also applies to z/OS for defining client-connection channels that connect to servers on the platforms supporting SPX.

### **TCP**

Transmission Control Protocol - part of the TCP/IP protocol suite

### **Multi** **USECLTID**

Specifies that the client ID should be used for authorization checks for an AMQP channel, instead of the **MCAUSER** attribute value.

### **NO**

The MCA user ID should be used for authorization checks.

### **YES**

The client ID should be used for authorization checks.

### **USEDLQ**

Determines whether the dead-letter queue is used when messages cannot be delivered by channels.

### **NO**

Messages that cannot be delivered by a channel are treated as a failure. The channel either discards the message, or the channel ends, in accordance with the **NPMSPEED** setting.

### **YES**

When the **DEADQ** queue manager attribute provides the name of a dead-letter queue, then it is used, else the behavior is as for NO. YES is the default value.

### **USERID(string)**

Task user identifier. The maximum length is 12 characters.

This parameter is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

**Multi** On Multiplatforms, this parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RQSTR, CLNTCONN, or CLUSSDR.

**z/OS** On z/OS, it is supported only for CLNTCONN channels.

Although the maximum length of the parameter is 12 characters, only the first 10 characters are used.

On the receiving end, if passwords are kept in encrypted format and the LU 6.2 software is using a different encryption method, an attempt to start the channel fails with invalid security details. You can avoid invalid security details by modifying the receiving SNA configuration to either:

- Turn off password substitution, or
- Define a security user ID and password.

### **XMITQ(string)**

Transmission queue name.

The name of the queue from which messages are retrieved. See [Rules for naming IBM MQ objects](#).

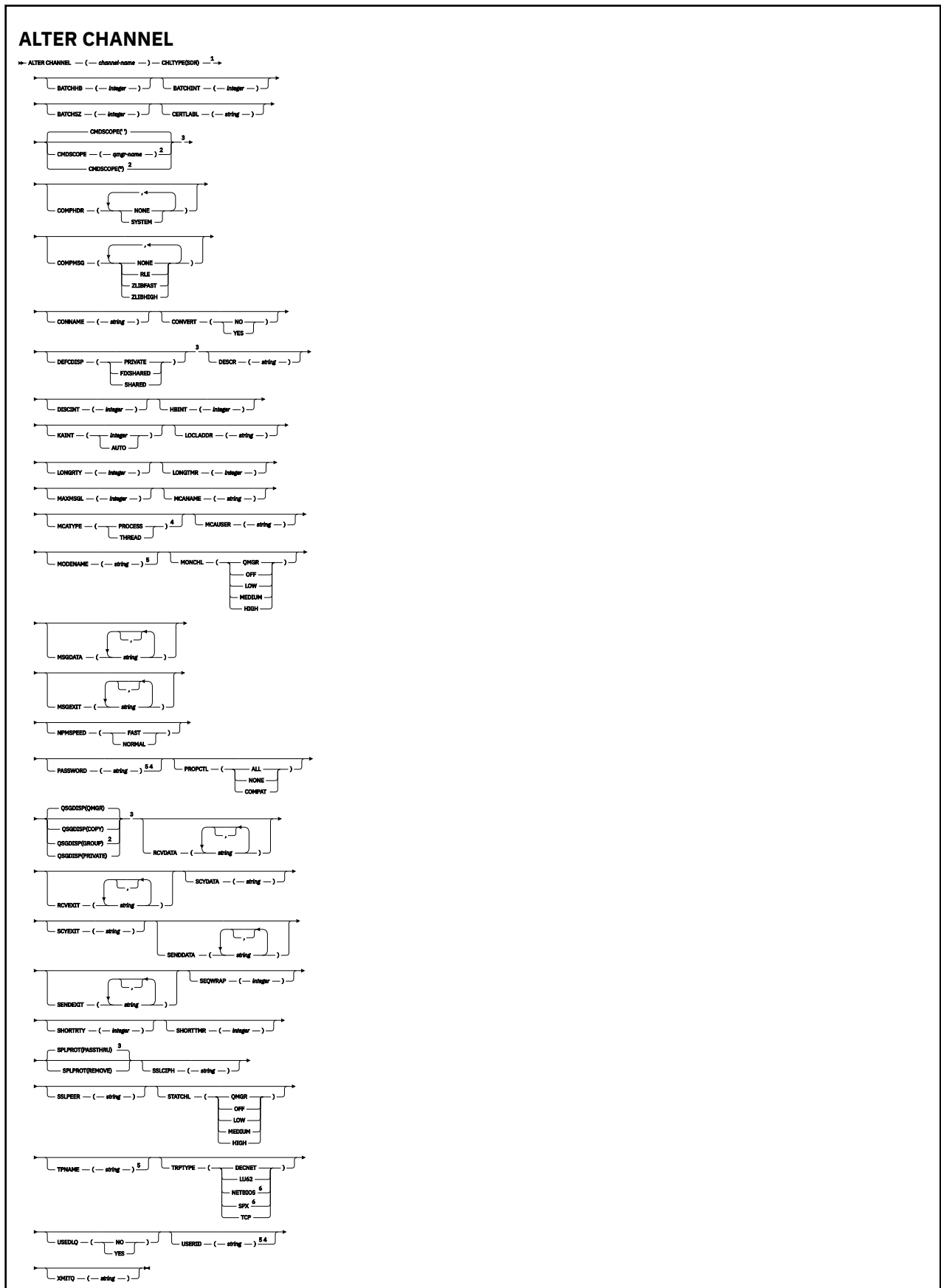
This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR or SVR. For these channel types, this parameter is required.

There is a separate syntax diagram for each type of channel:



## Sender channel

Syntax diagram for a sender channel when using the **ALTER CHANNEL** command.



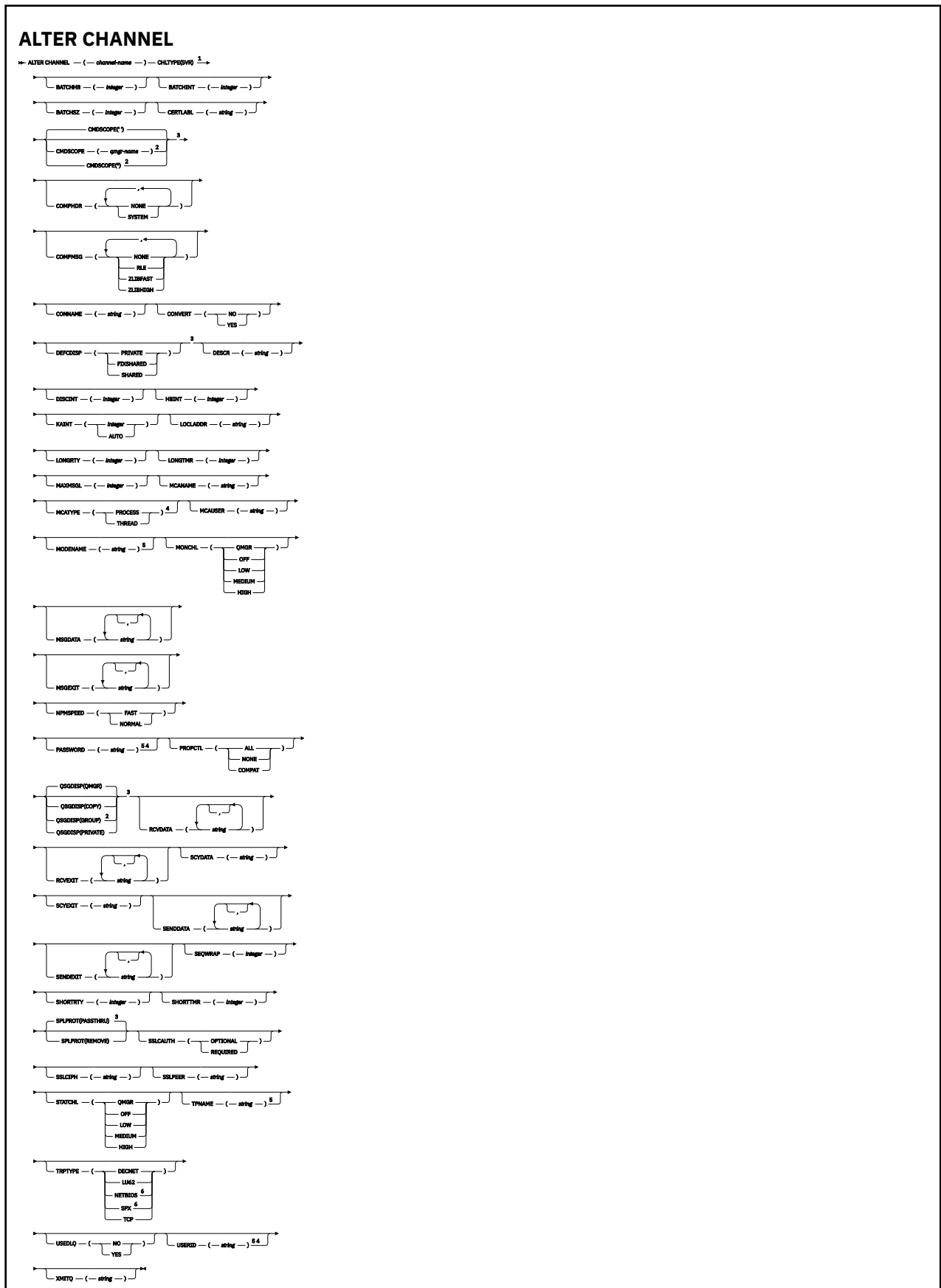
Notes:

- <sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Not valid on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Valid only Windows.

The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.

## Server channel

Syntax diagram for a server channel when using the **ALTER CHANNEL** command.



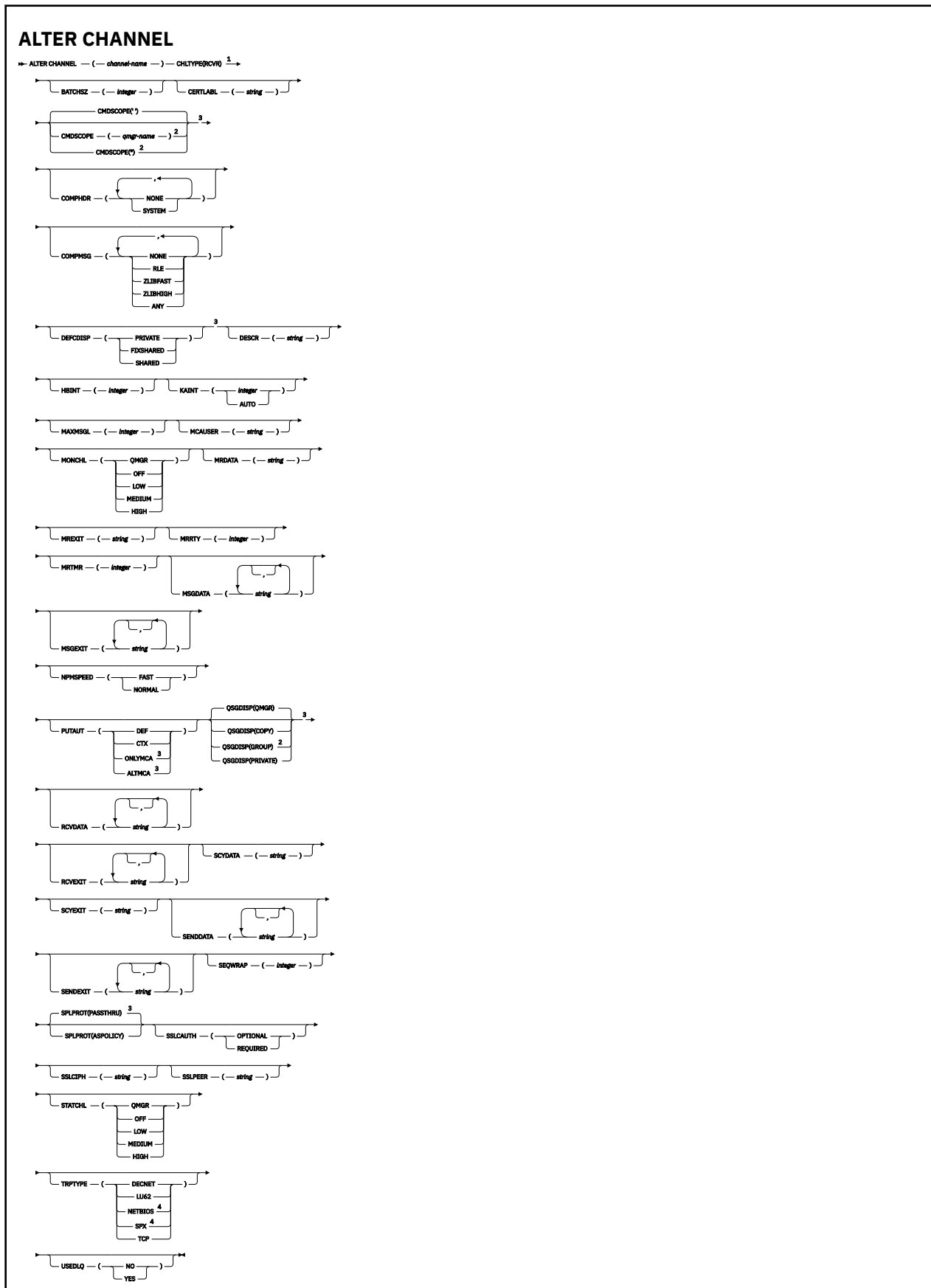
Notes:

- <sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Not valid on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Valid only on Windows.

The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.

## Receiver channel

Syntax diagram for a receiver channel when using the **ALTER CHANNEL** command.



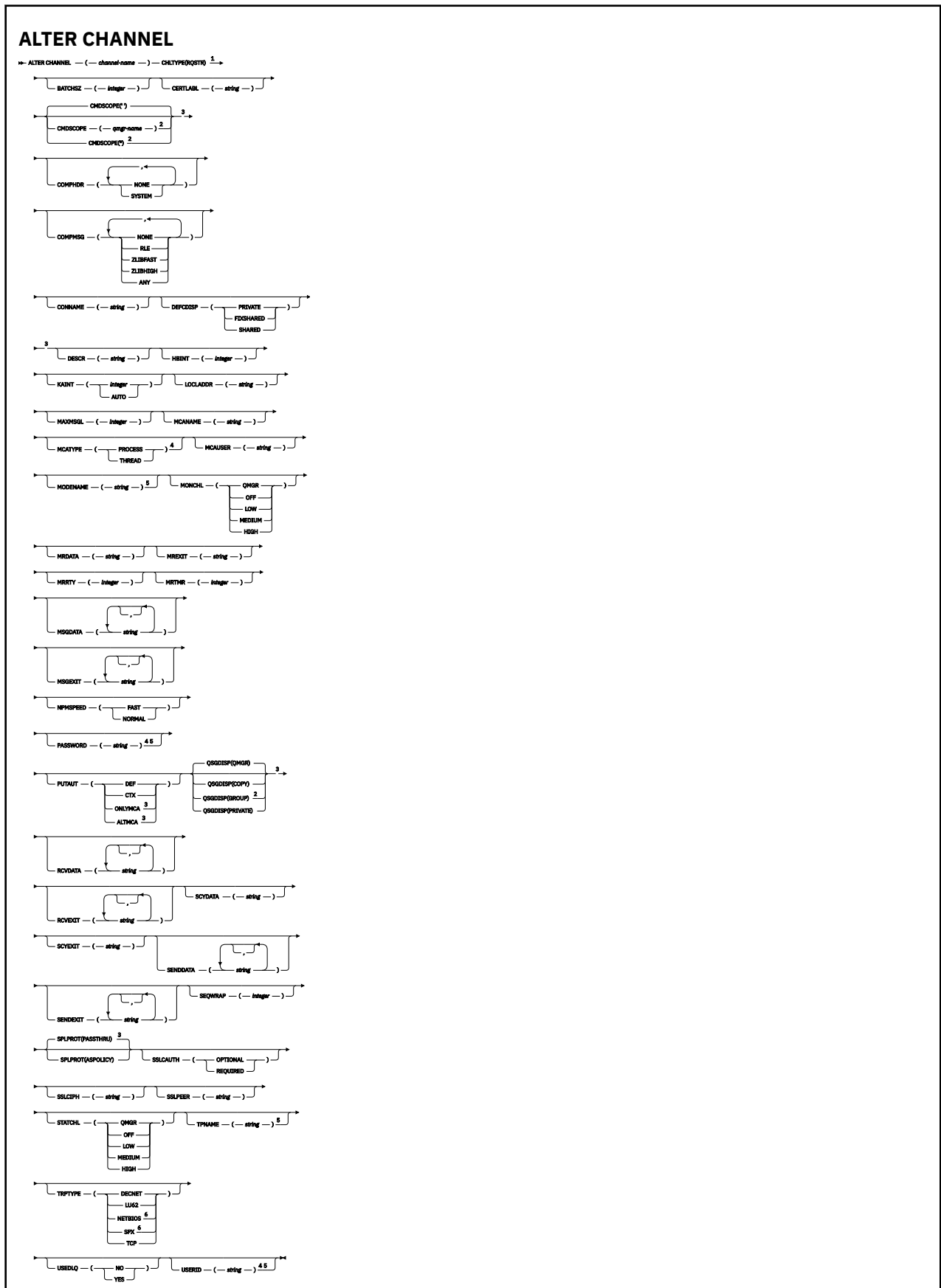
Notes:

- <sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Valid only on Windows.

The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.

## Requester channel

Syntax diagram for a requester channel when using the **ALTER CHANNEL** command.



Notes:

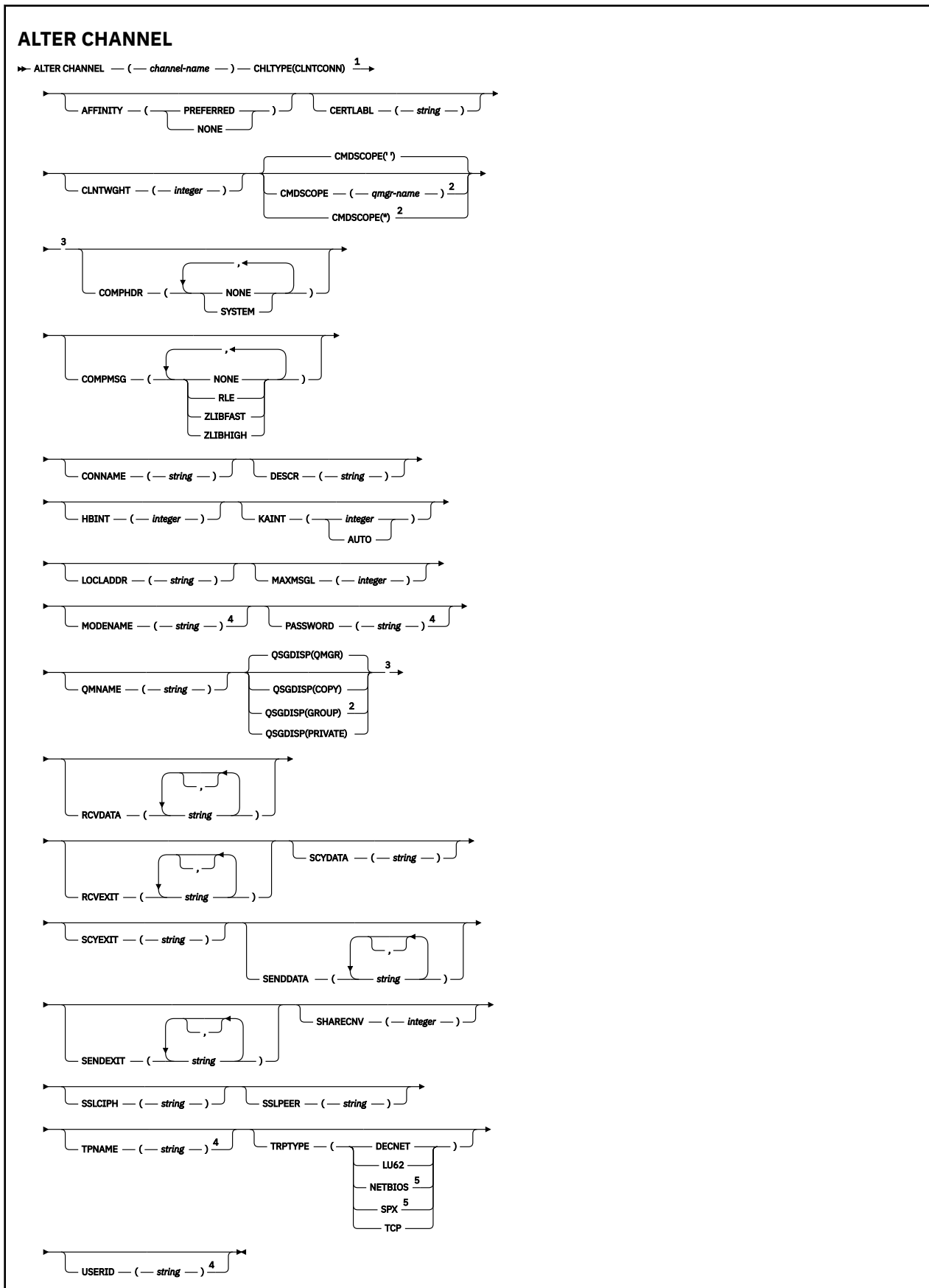
- <sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Not valid on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Valid only on Windows.

The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.



## Client-connection channel

Syntax diagram for a client-connection channel when using the **ALTER CHANNEL** command.



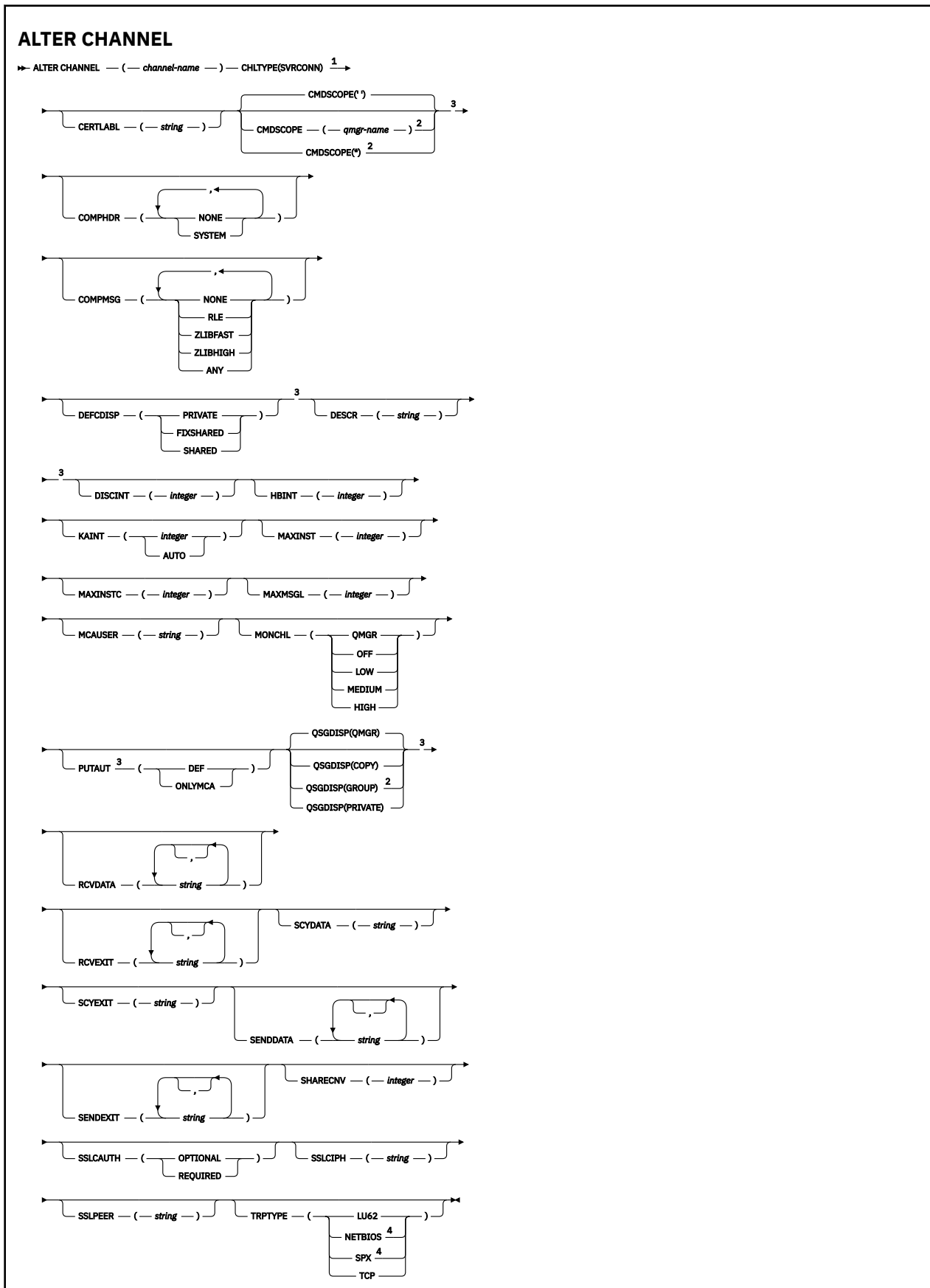
**Notes:**

- <sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Valid only if TRPTYPE is LU62.
- <sup>5</sup> Valid only for clients to be run on DOS and Windows.

The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.

## Server-connection channel

Syntax diagram for a server-connection channel when using the **ALTER CHANNEL** command.



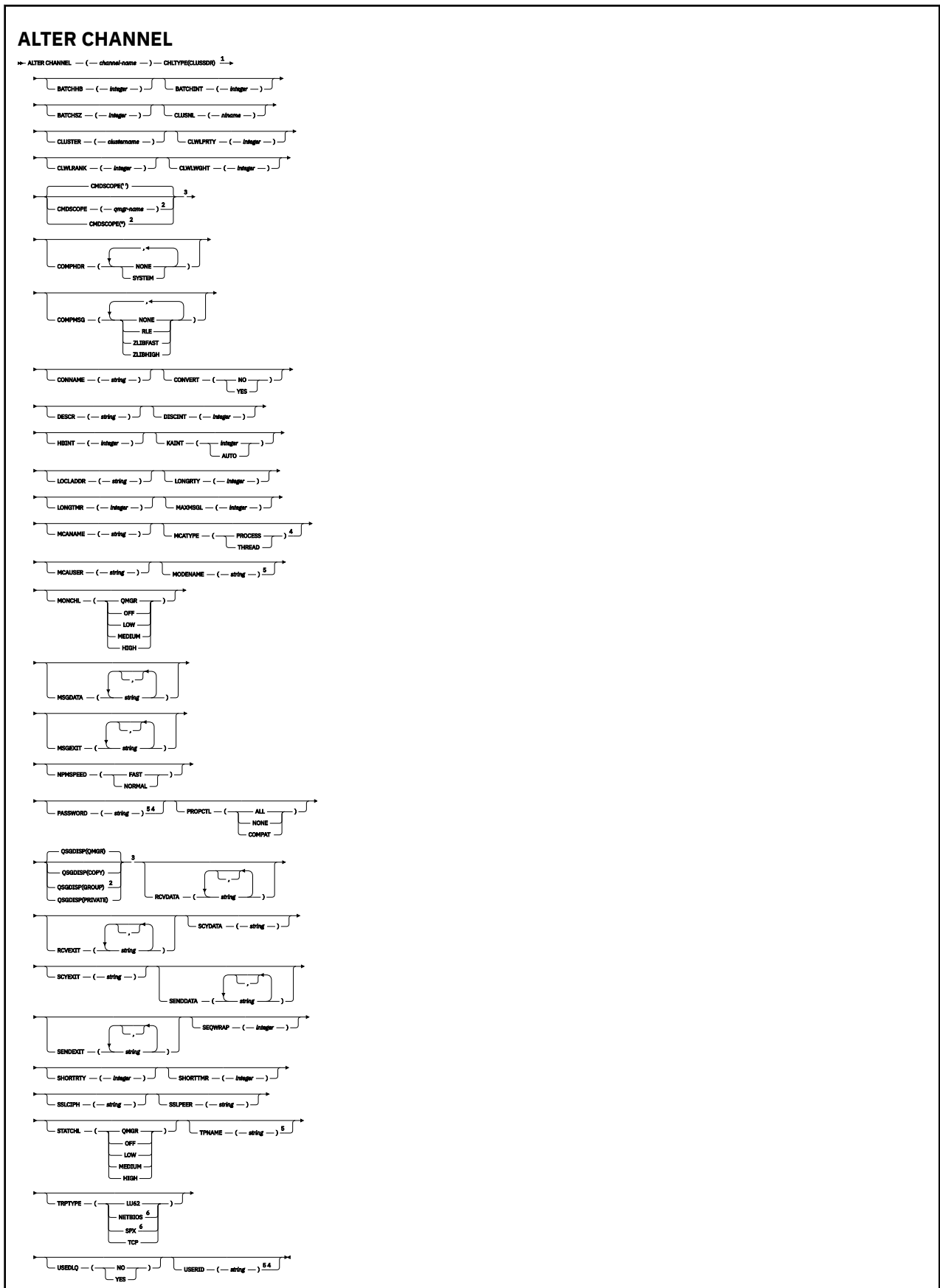
**Notes:**

- <sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Valid only for clients to be run on Windows.

The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.

## Cluster-sender channel

Syntax diagram for a cluster-sender channel when using the **ALTER CHANNEL** command.



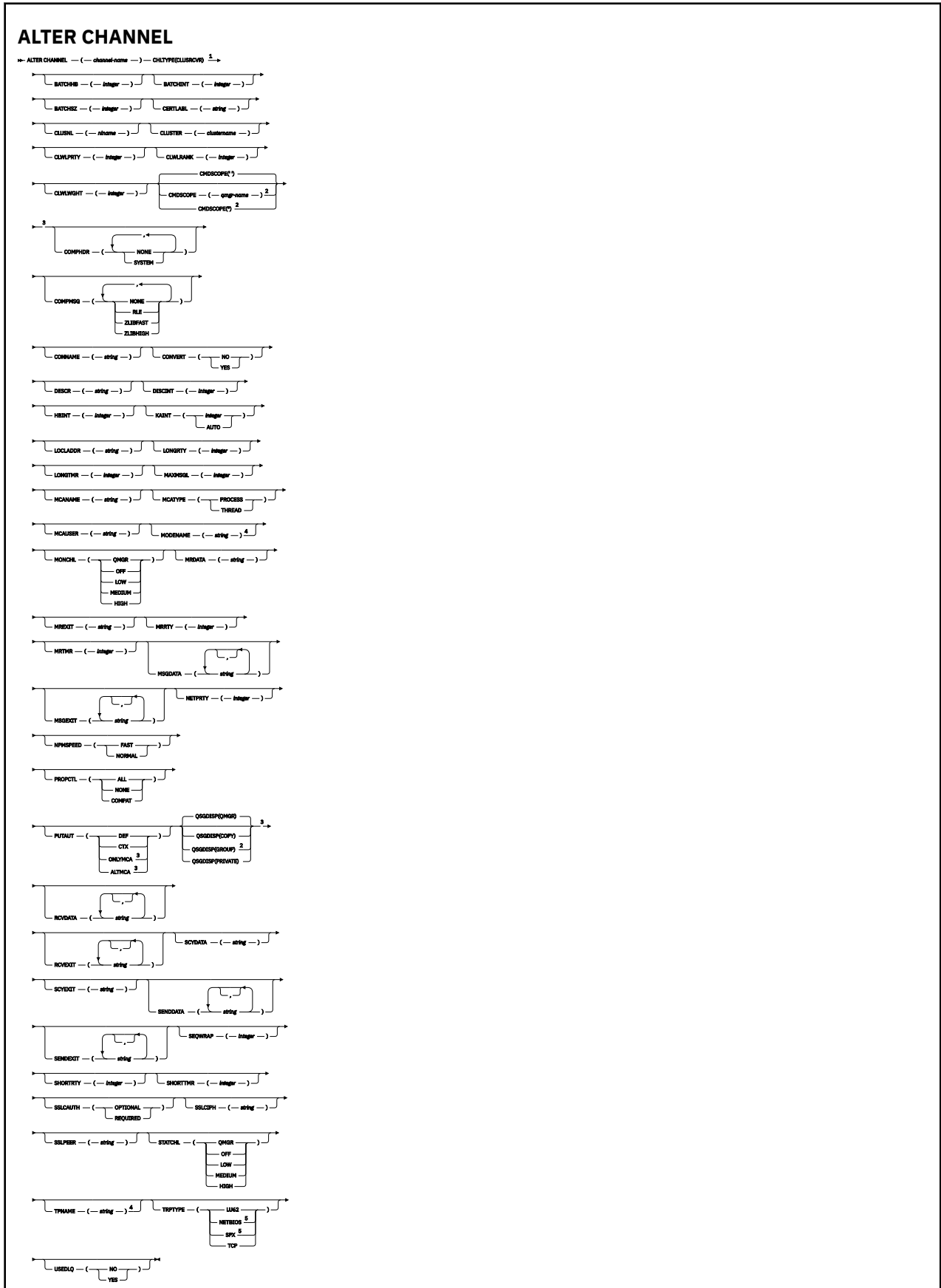
Notes:

- <sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Not valid on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Valid only Windows.

The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.

## Cluster-receiver channel

Syntax diagram for a cluster-receiver channel when using the **ALTER CHANNEL** command.



Notes:

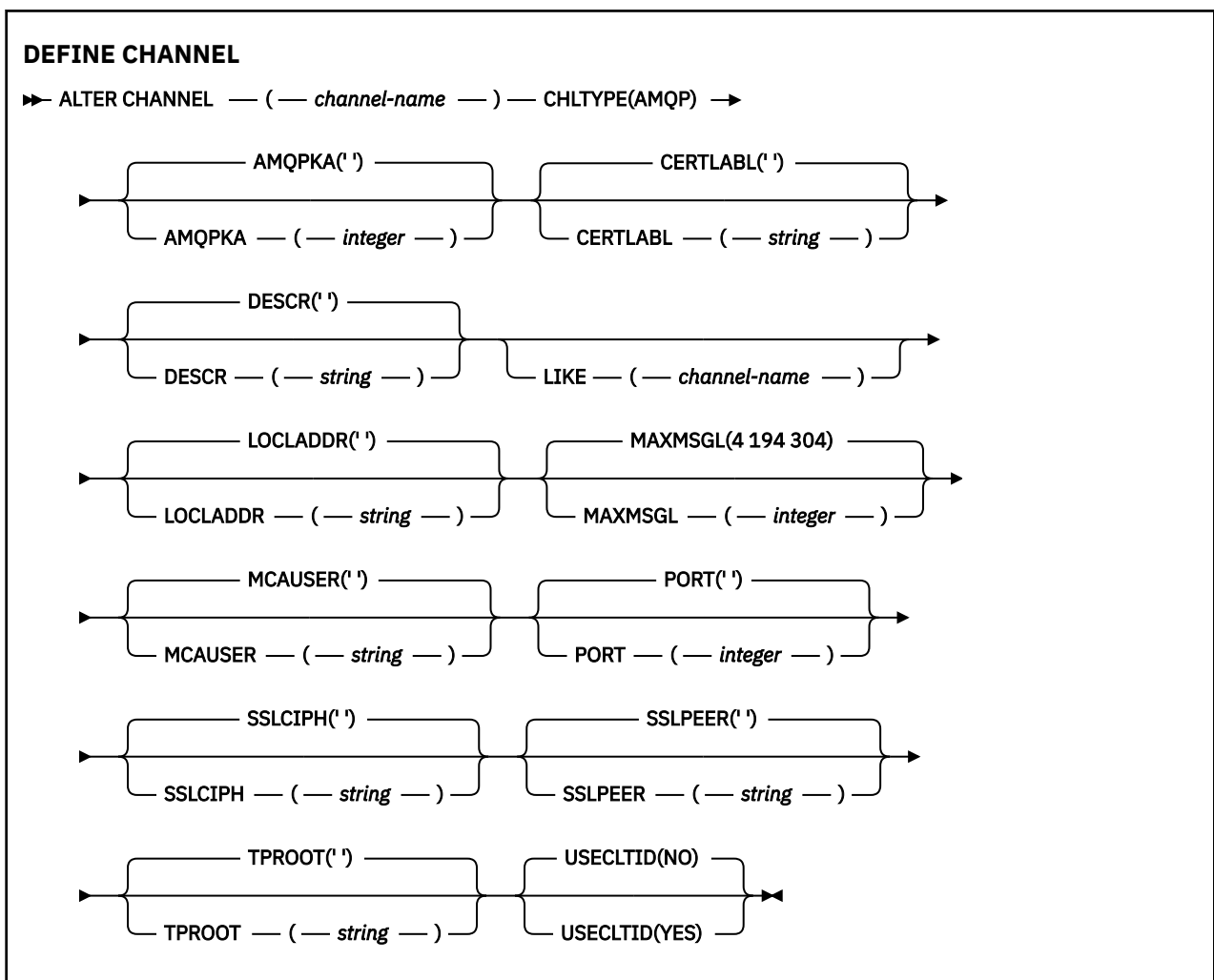
- <sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Valid only if TRPTYPE is LU62.
- <sup>5</sup> Valid only on Windows.

The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.

**ALW** **AMQP channel**

Syntax diagram for an AMQP channel when using the **ALTER CHANNEL** command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



The parameters are described in [“ALTER CHANNEL \(alter channel settings\)”](#) on page 274.



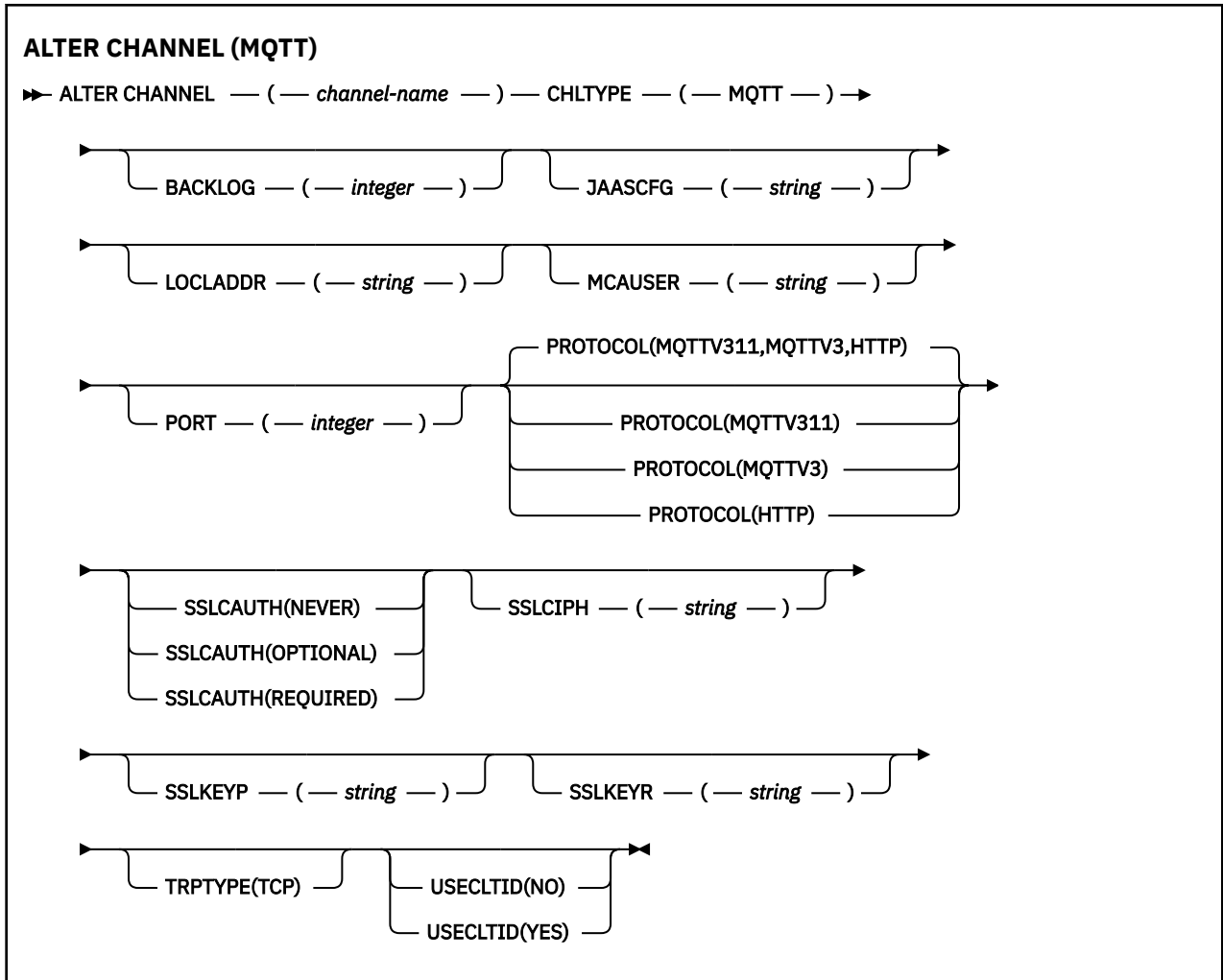
## ALW ALTER CHANNEL (alter channel settings) MQTT

Syntax diagram for a telemetry channel when using the **ALTER CHANNEL** command.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**Synonym:** ALT CHL



### Usage notes

The telemetry (MQXR) service must be running when you issue this command. For instructions on how to start the telemetry (MQXR) service, see [Configuring a queue manager for telemetry on Linux or Configuring a queue manager for telemetry on Windows](#).

### Parameter descriptions for ALTER CHANNEL (MQTT)

#### (*channel-name*)

The name of the channel definition.

#### **BACKLOG**(*integer*)

The number of outstanding connection requests that the telemetry channel can support at any one time. When the backlog limit is reached, any further clients trying to connect will be refused connection until the current backlog is processed.

The value is in the range 0 - 999999999.

The default value is 4096.

## **CHLTYPE**

Channel type. MQTT (telemetry) channel.

## **JAASCFG (string)**

The name of a stanza in the JAAS configuration file.

See [Authenticating an MQTT client Java app with JAAS](#)

## **LOCLADDR (ip-addr)**

LOCLADDR is the local communications address for the channel. Use this parameter if you want to force the client to use a particular IP address. LOCLADDR is also useful to force a channel to use an IPv4 or IPv6 address if a choice is available, or to use a particular network adapter on a system with multiple network adapters.

The maximum length of **LOCLADDR** is MQ\_LOCAL\_ADDRESS\_LENGTH.

If you omit **LOCLADDR**, a local address is automatically allocated.

### **ip-addr**

*ip-addr* is a single network address, specified in one of three forms:

#### **IPv4 dotted decimal**

For example 192.0.2.1

#### **IPv6 hexadecimal notation**

For example 2001:DB8:0:0:0:0:0:0

#### **Alphanumeric host name form**

For example WWW.EXAMPLE.COM

If an IP address is entered, only the address format is validated. The IP address itself is not validated.

## **MCAUSER(string)**

Message channel agent user identifier.

The maximum length of the string is 12 characters. On Windows, you can optionally qualify a user identifier with the domain name in the format user@domain.

If this parameter is nonblank, and **USECLNTID** is set to NO, then this user identifier is used by the telemetry service for authorization to access IBM MQ resources.

If this parameter is blank, and **USECLNTID** is set to NO, then the user name flowed in the MQTT CONNECT Packet is used. See [MQTT client identity and authorization](#).

## **PORT(integer)**

The port number on which the telemetry (MQXR) service accepts client connections. The default port number for a telemetry channel is 1883; and the default port number for a telemetry channel secured using SSL is 8883. Specifying a port value of 0 causes MQTT to dynamically allocate an available port number.

## **PROTOCOL**

The following communication protocols are supported by the channel:

### **MQTTV311**

The channel accepts connections from clients using the protocol defined by the [MQTT 3.1.1](#) Oasis standard. The functionality provided by this protocol is almost identical to that provided by the pre-existing MQTTV3 protocol.

### **MQTTV3**

The channel accepts connections from clients using the [MQTT V3.1 Protocol Specification](#) from mqtt.org.

## HTTP

The channel accepts HTTP requests for pages, or WebSockets connections to MQ Telemetry.

To accept connections from clients using different protocols, specify the acceptable values as a comma-delimited list. For example if you specify MQTTV3, HTTP the channel accepts connections from clients using either MQTTV3 or HTTP. If you specify no client protocols, the channel accepts connections from clients using any of the supported protocols.

If you are using IBM MQ 8.0.0 Fix Pack 3 or later, and your configuration includes an MQTT channel that was last modified in an earlier version of the product, you must explicitly change the protocol setting to prompt the channel to use the MQTTV311 option. This is so even if the channel does not specify any client protocols, because the specific protocols to use with the channel are stored at the time the channel is configured, and previous versions of the product have no awareness of the MQTTV311 option. To prompt a channel in this state to use the MQTTV311 option, explicitly add the option then save your changes. The channel definition is now aware of the option. If you subsequently change the settings again, and specify no client protocols, the MQTTV311 option is still included in the stored list of supported protocols.

## SSLCAUTH

Defines whether IBM MQ requires a certificate from the TLS client. The initiating end of the channel acts as the TLS client, so this parameter applies to the end of the channel that receives the initiation flow, which acts as the TLS server.

### NEVER

IBM MQ never requests a certificate from the TLS client.

### REQUIRED

IBM MQ requires and validates a certificate from the TLS client.

### OPTIONAL

IBM MQ lets the TLS client decide whether to provide a certificate. If the client sends a certificate, the contents of this certificate are validated as normal.


## SSLCIPH(*string*)

When **SSLCIPH** is used with a telemetry channel, it means TLS Cipher Suite. The TLS cipher suite is the one supported by the JVM that is running the telemetry (MQXR) service. If the parameter is blank, no attempt is made to use TLS on the channel.

If you plan to use SHA-2 cipher suites, see [System requirements for using SHA-2 cipher suites with MQTT channels](#).

## SSLKEYP(*string*)




The passphrase for the TLS key repository.

 If the MQXR service is configured for encryption of passphrases by specifying the **-sf** option in STARTARG for the service, then the pass phrase will be encrypted. For more information on encryption of passphrases, see [Encryption of passphrases for MQTT TLS channels](#).

## SSLKEYR(*string*)

The full path name of the TLS key repository file, the store for digital certificates and their associated private keys. If you do not specify a key file, TLS is not used.

The maximum length of the string is 256 characters;

-   On AIX and Linux, the name is of the form *pathname/keyfile*.
-  On Windows, the name is of the form *pathname\keyfile*.

where *keyfile* is specified without the suffix `.jks`, and identifies a Java keystore file.

## TRPTYPE (*string*)

The transmission protocol to be used:

## TCP

TCP/IP.

## USECLTID

Decide whether you want to use the MQTT client ID for the new connection as the IBM MQ user ID for that connection. If this property is specified, the user name supplied by the client is ignored.

If you set this parameter to YES, then **MCAUSER** must be blank.

If **USECLNTID** is set to NO, and **MCAUSER** is blank, then the user name flowed in the MQTT CONNECT Packet is used. See [MQTT client identity and authorization](#).

### Related concepts

[Telemetry channel configuration for MQTT client authentication using TLS](#)

[Telemetry channel configuration for channel authentication using TLS](#)

[CipherSpecs and CipherSuites](#)

[System requirements for using SHA-2 cipher suites with MQTT channels](#)

### Related reference

[“DEFINE CHANNEL \(define a new channel\) for MQTT” on page 523](#)

Syntax diagram for a telemetry channel when using the **DEFINE CHANNEL** command.

Multi

## **ALTER COMMINFO (alter communication information object) on Multiplatforms**

Use the MQSC command ALTER COMMINFO to alter the parameters of a communication information object.

### Using MQSC commands

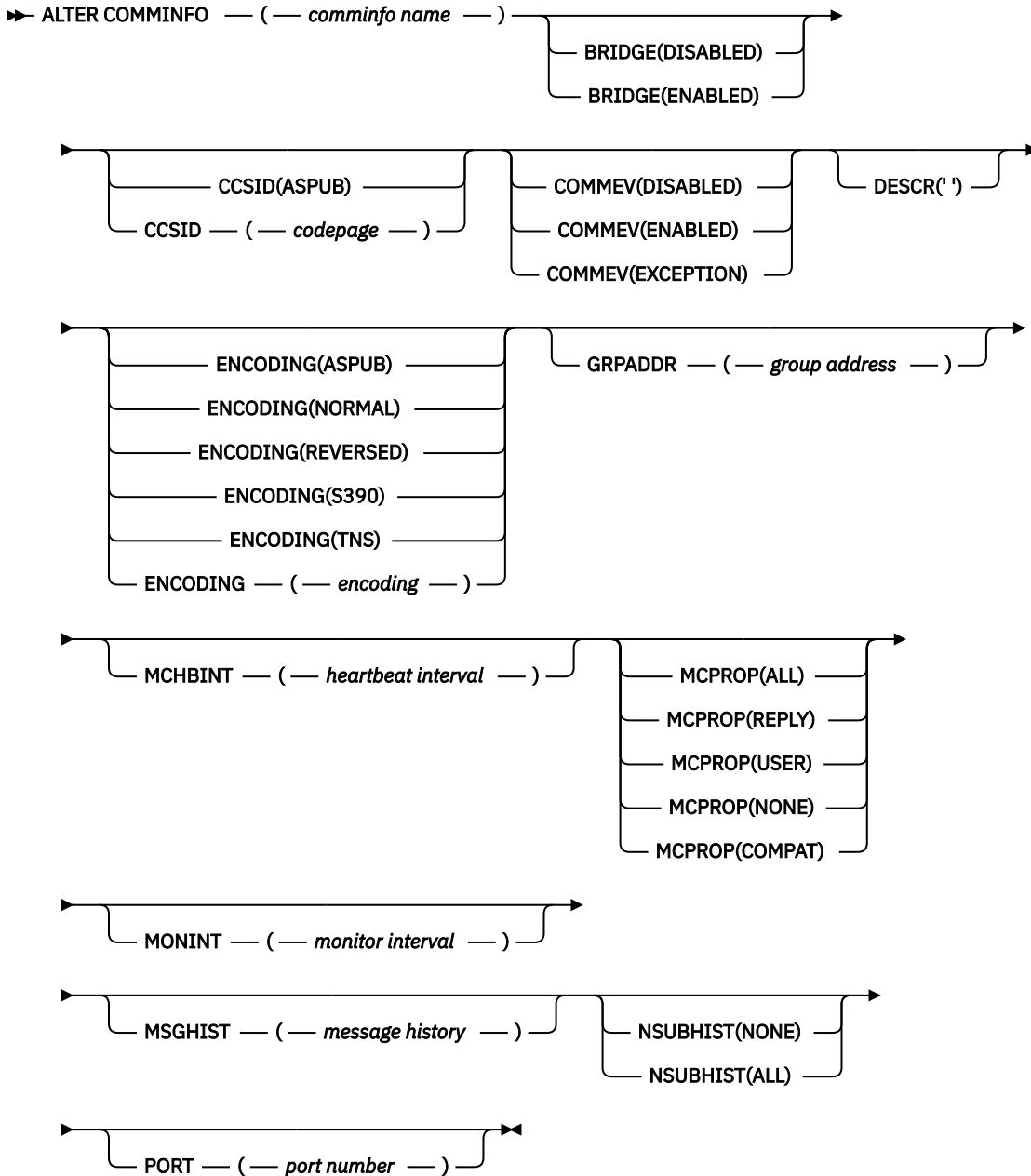
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

Parameters not specified in the **ALTER COMMINFO** command result in the existing values for those parameters being left unchanged.

- [Syntax diagram](#)
- [“Parameter descriptions for ALTER COMMINFO” on page 333](#)

**Synonym:** ALT COMMINFO

## ALTER COMMINFO



Notes:

### Parameter descriptions for ALTER COMMINFO

#### (*comminfo name*)

Name of the communications information object. This parameter is required.

The name must not be the same as any other communications information object name currently defined on this queue manager. See [Rules for naming IBM MQ objects](#).

#### BRIDGE

Controls whether publications from applications not using Multicast are bridged to applications using Multicast. Bridging does not apply to topics that are marked as **MCAST(ONLY)**. As these topics can only be Multicast traffic, it is not applicable to bridge to the queue's publish/subscribe domain.

**DISABLED**

Publications from applications not using Multicast are not bridged to applications that do use Multicast.

**ENABLED**

Publications from applications not using Multicast are bridged to applications that do use Multicast.

**CCSID(*integer*)**

The coded character set identifier that messages are transmitted on. Specify a value in the range 1 through 65535.

The CCSID must specify a value that is defined for use on your platform, and use a character set that is appropriate to the queue manager's platform. If you use this parameter to change the CCSID, applications that are running when the change is applied continue to use the original CCSID therefore you must stop and restart all running applications before you continue. Running applications include the command server and channel programs. Stop and restart all running applications, stop and restart the queue manager after changing this parameter.

The CCSID can also be set to ASPUB, which means that the coded character set is taken from that supplied in the published message.

**COMMEV**

Controls whether event messages are generated for Multicast handles that are created using this COMMINFO object. Events are only generated if they are enabled using the **MONINT** parameter.

**DISABLED**

Publications from applications not using Multicast are not bridged to applications that do use Multicast.

**ENABLED**

Publications from applications not using Multicast are bridged to applications that do use Multicast.

**EXCEPTION**

Event messages are written if the message reliability is below the reliability threshold. The reliability threshold is set to 90 by default.

**DESCR(*string*)**

Plain-text comment. It provides descriptive information about the communication information object when an operator issues the DISPLAY COMMINFO command (see [“DISPLAY COMMINFO \(display communication information\) on Multiplatforms”](#) on page 722).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

**ENCODING**

The encoding that the messages are transmitted in.

**AS PUB**

The encoding of the message is taken from that supplied in the published message.

**NORMAL****REVERSED****S390****TNS****encoding****GRPADDR**

The group IP address or DNS name.

It is the responsibility of the administrator to manage the group addresses. It is possible for all multicast clients to use the same group address for every topic; only the messages that match outstanding subscriptions on the client are delivered. Using the same group address can be inefficient because every client has to examine and process every multicast packet in the network. It is more efficient to allocate different IP group addresses to different topics or sets of topics, but this allocation requires careful management, especially if other non-MQ multicast applications are in use on the network.

### **MCHBINT**

The heartbeat interval is measured in milliseconds, and specifies the frequency at which the transmitter notifies any receivers that there is no further data available.

### **MCPROP**

The multicast properties control how many of the MQMD properties and user properties flow with the message.

#### **All**

All user properties and all the fields of the MQMD are transported.

#### **Reply**

Only user properties, and MQMD fields that deal with replying to the messages, are transmitted. These properties are:

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

#### **User**


Only the user properties are transmitted.

#### **NONE**

No user properties or MQMD fields are transmitted.

#### **COMPAT**

This value causes the transmission of the message to be done in a compatible mode to RMM allowing some inter-operation with the current XMS applications and Broker RMM applications.

 XMS .NET Multicast messaging (using RMM) was deprecated from IBM MQ 9.2 and is removed at IBM MQ 9.3.

### **MONINT( *integer* )**

How frequently, in seconds, that monitoring information is updated. If events messages are enabled, this parameter also controls how frequently event messages are generated about the status of the Multicast handles created using this COMMINFO object.

A value of 0 means that there is no monitoring.

### **MSGHIST**

The maximum message history is the amount of message history that is kept by the system to handle retransmissions in the case of NACKs (negative acknowledgments).

A value of 0 gives the least level of reliability.

### **NSUBHIST**

The new subscriber history controls whether a subscriber joining a publication stream receives as much data as is currently available, or receives only publications made from the time of the subscription.

#### **NONE**

A value of NONE causes the transmitter to transmit only publication made from the time of the subscription.

**ALL**

A value of ALL causes the transmitter to retransmit as much history of the topic as is known. In some circumstances, this retransmission can give a similar behavior to retained publications.

**Note:** Using the value of ALL might have a detrimental effect on performance if there is a large topic history because all the topic history is retransmitted.

**PORT(*integer*)**

The port number to transmit on.

Multi**ALTER LISTENER (alter an existing listener) on Multiplatforms**

Use MQSC command **ALTER LISTENER** to alter the parameters of an existing IBM MQ listener definition. If the listener is already running, any changes you make to its definition are effective only after the next time that the listener is started.

**Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

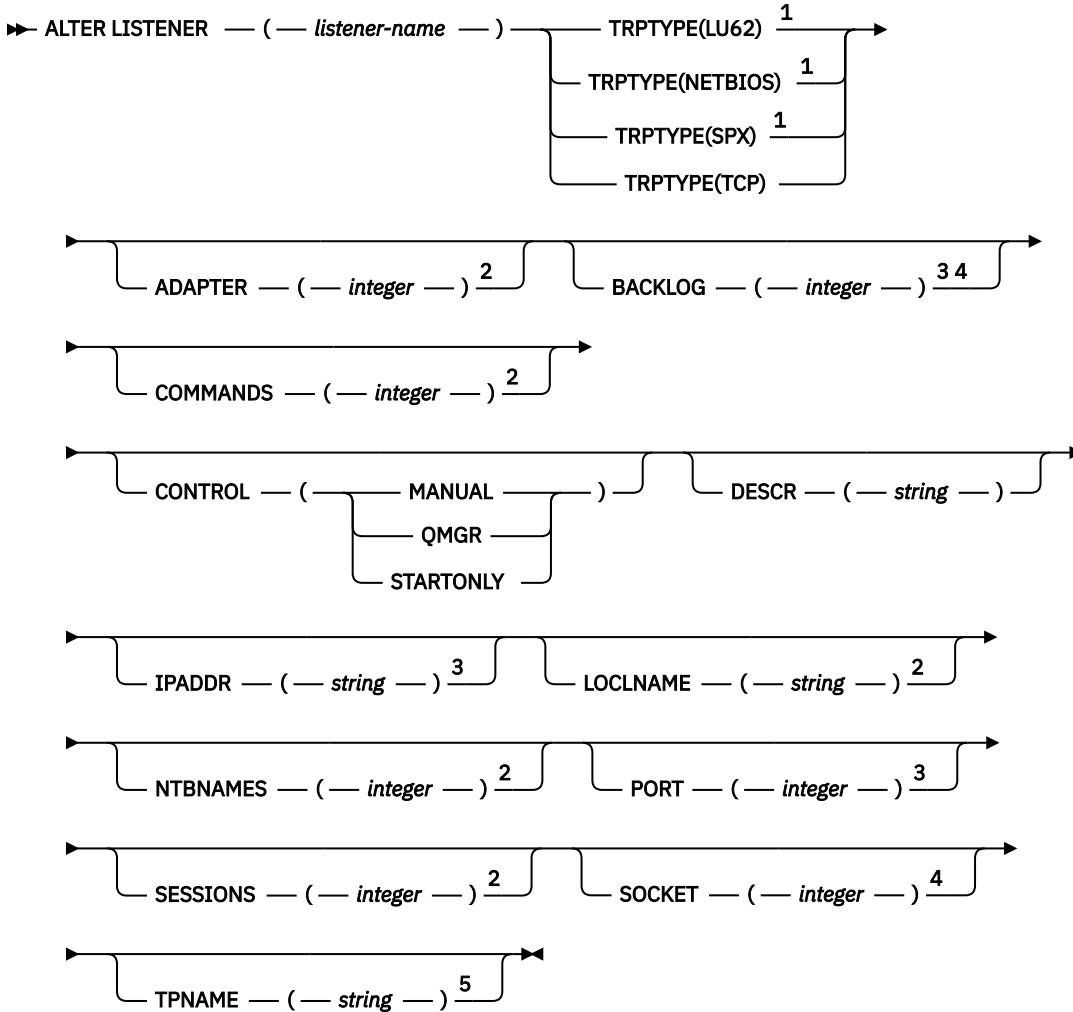
Parameters not specified in the **ALTER LISTENER** command result in the existing values for those parameters being left unchanged.

- [Syntax diagram](#)
- [“Parameter descriptions for ALTER LISTENER” on page 337](#)

**Synonym:** ALT LSTR



## ALTER LISTENER



### Notes:

- <sup>1</sup> Valid only on Windows.
- <sup>2</sup> Valid only on Windows when TRPTYPE is NETBIOS.
- <sup>3</sup> Valid when TRPTYPE is TCP.
- <sup>4</sup> Valid on Windows when TRPTYPE is SPX.
- <sup>5</sup> Valid only on Windows when TRPTYPE is LU62.

## Parameter descriptions for ALTER LISTENER

### (*listener-name*)

Name of the IBM MQ listener definition (see [Rules for naming IBM MQ objects](#)). This is required.

The name must not be the same as any other listener definition currently defined on this queue manager (unless REPLACE is specified).

### **Windows** ADAPTER(*integer*)

The adapter number on which NetBIOS listens. This parameter is valid only on Windows when TRPTYPE is NETBIOS.

### BACKLOG(*integer*)

The number of concurrent connection requests that the listener supports.

**Windows** **COMMANDS(integer)**

The number of commands that the listener can use. This parameter is valid only on Windows when **TRPTYPE** is NETBIOS.

**CONTROL(string)**

Specifies how the listener is to be started and stopped.:

**MANUAL**

The listener is not to be started automatically or stopped automatically. It is to be controlled by use of the **START LISTENER** and **STOP LISTENER** commands.

**QMGR**

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

**STARTONLY**

The listener is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**DESCR(string)**

Plain-text comment. It provides descriptive information about the listener when an operator issues the **DISPLAY LISTENER** command (see [“DISPLAY LISTENER \(display listener information\) on Multiplatforms”](#) on page 742 ).

It should contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

**IPADDR(string)**

IP address for the listener specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric host name form. If you do not specify a value for this parameter, the listener listens on all configured IPv4 and IPv6 stacks.

**LIKE(listener-name)**

The name of a listener, with parameters that are used to model this definition.

This parameter applies only to the **DEFINE LISTENER** command.

If this field is not filled in, and you do not complete the parameter fields related to the command, the values are taken from the default definition for listeners on this queue manager. This is equivalent to specifying:

```
LIKE(SYSTEM.DEFAULT.LISTENER)
```

A default listener is provided but it can be altered by the installation of the default values required. See [Rules for naming IBM MQ objects](#).

**Windows** **LOCLNAME(string)**

The NetBIOS local name that the listener uses. This parameter is valid only on Windows when **TRPTYPE** is NETBIOS.

**Windows** **NTBNAMES(integer)**

The number of names that the listener can use. This parameter is valid only on Windows when **TRPTYPE** is NETBIOS.

**PORT(integer)**

The port number for TCP/IP. This is valid only when **TRPTYPE** is TCP. It must not exceed 65535.

**Windows** **SESSIONS(integer)**

The number of sessions that the listener can use. This parameter is valid only on Windows when **TRPTYPE** is NETBIOS.

**SOCKET(*integer*)**

The SPX socket on which to listen. This is valid only if **TRPTYPE** is SPX.

**Windows TPNAME(*string*)**

The LU 6.2 transaction program name (maximum length 64 characters). This parameter is valid only on Windows when **TRPTYPE** is LU62.

**TRPTYPE(*string*)**

The transmission protocol to be used:

**Windows LU62**

SNA LU 6.2. This is valid only on Windows.

**Windows NETBIOS**

NetBIOS. This is valid only on Windows.

**Windows SPX**

Sequenced packet exchange. This is valid only on Windows.

**TCP**

TCP/IP.

**Related tasks**

[Using the TCP listener backlog option on AIX and Linux](#)

**ALTER NAMELIST (alter a list of names)**

Use the MQSC command **ALTER NAMELIST** to alter a list of names. This list is most commonly a list of cluster names or queue names.

**Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

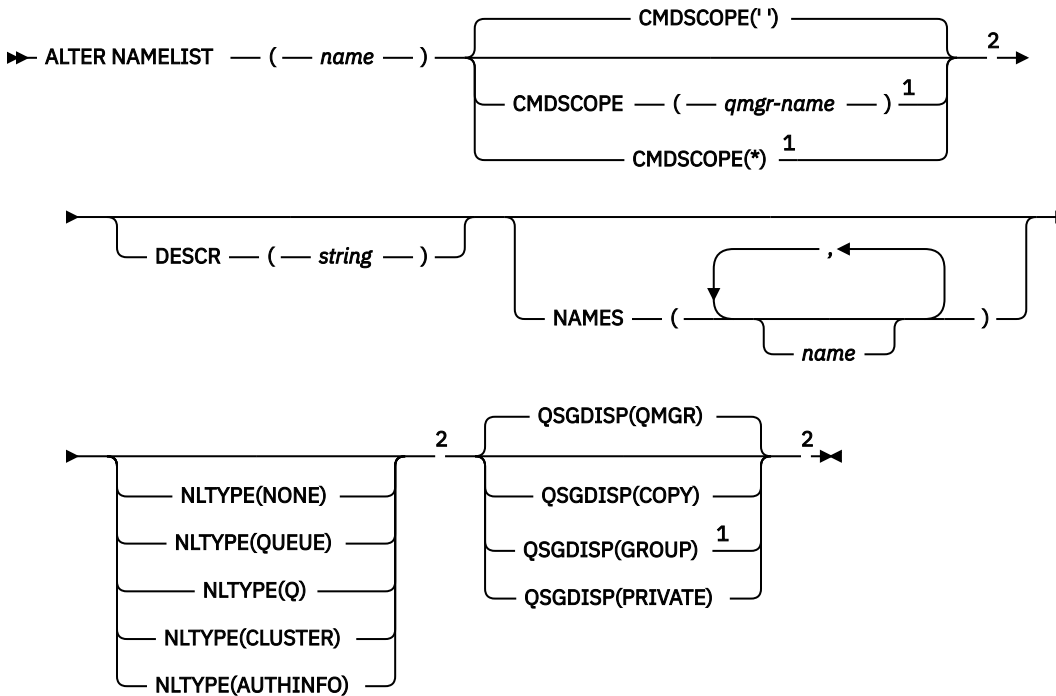
Parameters not specified in the **ALTER NAMELIST** command result in the existing values for those parameters being left unchanged.

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 340](#)
- [“Parameter descriptions for ALTER NAMELIST” on page 340](#)

**Synonym:** ALT NL

## ALTER NAMELIST



### Notes:

- <sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>2</sup> Valid only on z/OS.

## Usage notes

Successful completion of the command does not mean that the action completed. To check for true completion, see the [ALTER NAMELIST](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for ALTER NAMELIST

### (name)

Name of the list.

The name must not be the same as any other namelist name currently defined on this queue manager (unless **REPLACE** or **ALTER** is specified). See [Rules for naming IBM MQ objects](#).

### z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of specifying \* is the same as entering the command on every queue manager in the queue sharing group.

### **DESCR(string)**

Plain-text comment. It provides descriptive information about the namelist when an operator issues the **DISPLAY NAMELIST** command (see “[DISPLAY NAMELIST \(display a list of names\)](#)” on page 751).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

### **NAMES(name, ...)**

List of names.

The names can be of any type, but must conform to the rules for naming IBM MQ objects, with a maximum length of 48 characters.

An empty list is valid: specify **NAMES()**. The maximum number of names in the list is 256.

### **z/OS NLTYPE**

Indicates the type of names in the namelist.

This parameter is valid only on z/OS.

#### **NONE**

The names are of no particular type.

#### **QUEUE or Q**

A namelist that holds a list of queue names.

#### **CLUSTER**

A namelist that is associated with clustering, containing a list of the cluster names.

#### **AUTHINFO**

This namelist is associated with TLS and contains a list of authentication information object names.

Namelists used for clustering must have **NLTYPE (CLUSTER)** or **NLTYPE (NONE)**.

Namelists used for TLS must have **NLTYPE (AUTHINFO)**.

### **z/OS QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

QSGDISP	ALTER
COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (COPY)</b> . Any object residing in the shared repository, or any object defined using a command that had the parameters <b>QSGDISP(QMGR)</b> , is not affected by this command.

Table 137. Behavior for each of the QSGDISP values (continued)


QSGDISP	ALTER
GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameters <b>QSGDISP (GROUP)</b>. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command. If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre data-bbox="561 474 1466 573"> DEFINE NAMELIST(name) REPLACE QSGDISP(COPY) </pre> <p>The <b>ALTER</b> for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
PRIVATE	<p>The object resides on the page set of the queue manager that executes the command, and was defined with <b>QSGDISP (QMGR)</b> or <b>QSGDISP (COPY)</b>. Any object residing in the shared repository is unaffected.</p>
QMGR	<p>The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (QMGR)</b>. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.</p>

## ALTER PROCESS (alter an existing process definition)

Use the MQSC command **ALTER PROCESS** to alter the parameters of an existing IBM MQ process definition.

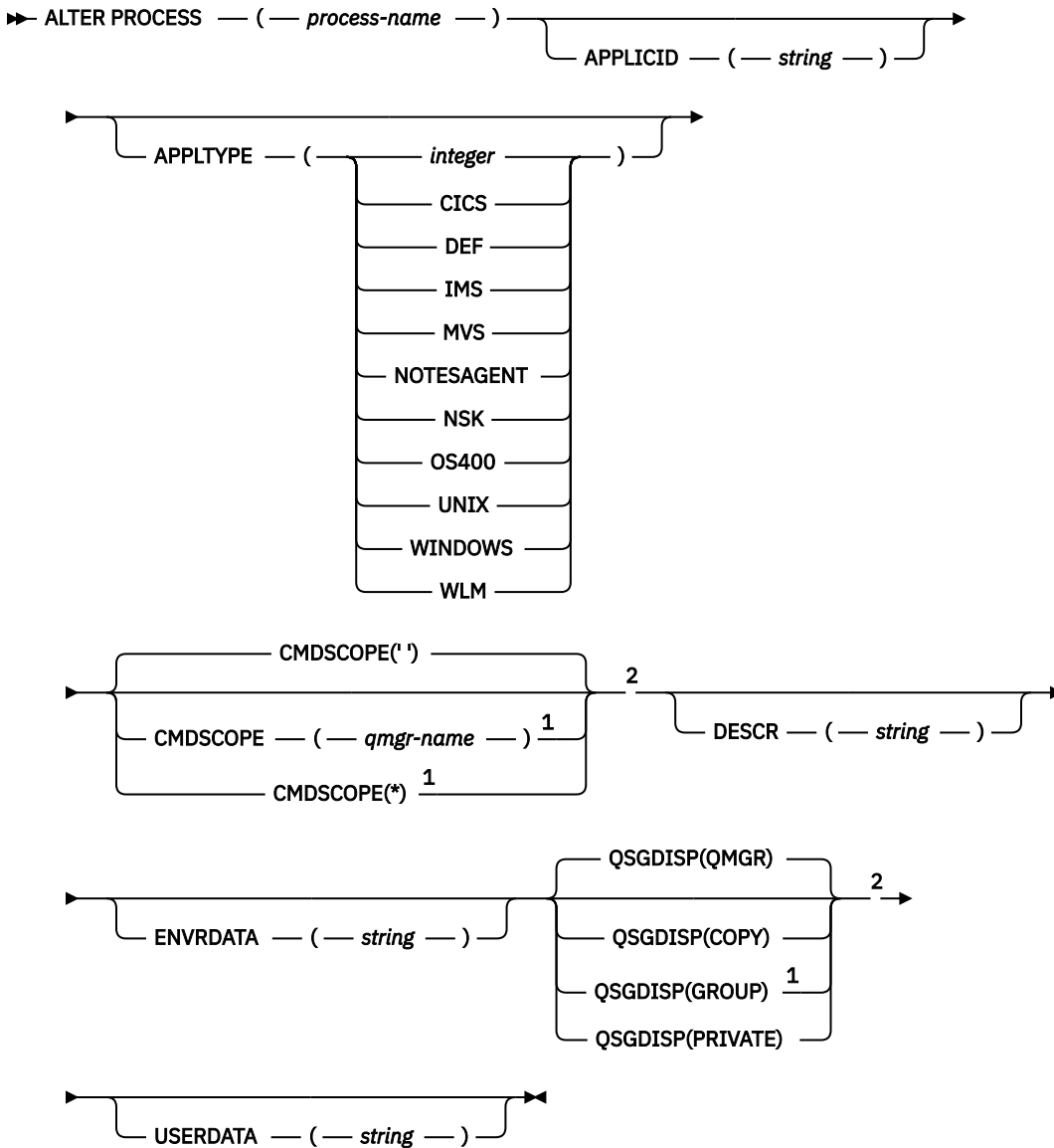
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

**Synonym:** ALT PRO

## ALTER PROCESS



### Notes:

<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

<sup>2</sup> Valid only on z/OS.

## Parameter descriptions for ALTER PROCESS

### *process-name*

Name of the IBM MQ process definition (see [Rules for naming IBM MQ objects](#) ). *process-name* is required.

The name must not be the same as any other process definition currently defined on this queue manager (unless **REPLACE** is specified).

### APPLICID(*string*)

The name of the application to be started. The name might typically be a fully qualified file name of an executable object. Qualifying the file name is particularly important if you have multiple IBM MQ installations, to ensure the correct version of the application is run. The maximum length is 256 characters.

For a CICS® application the name is a CICS transaction ID, and for an IMS application it is an IMS transaction ID.

**z/OS** On z/OS, for distributed queuing, it must be "CSQX start".

### **APPLTYPE(string)**

The type of application to be started. Valid application types are:

#### **integer**

A system-defined application type in the range zero through 65 535 or a user-defined application type in the range 65 536 through 999 999 999.

For certain values in the system range, a parameter from the following list can be specified instead of a numeric value:

#### **CICS**

Represents a CICS transaction.

**z/OS** **IMS**

Represents an IMS transaction.

**z/OS** **MVS**

Represents a z/OS application (batch or TSO).

#### **NOTESAGENT**

Represents a Lotus Notes® agent.

**IBM i** **OS400**

Represents an IBM i application.

**UNIX** **UNIX**

Represents a Linux or AIX application.

**Windows** **WINDOWS**

Represents a Windows application.

**z/OS** **WLM**

Represents a z/OS workload manager application.

#### **DEF**

Specifying DEF causes the default application type for the platform at which the command is interpreted to be stored in the process definition. This default cannot be changed by the installation. If the platform supports clients, the default is interpreted as the default application type of the server.

Only use application types (other than user-defined types) that are supported on the platform at which the command runs:

- **z/OS** On z/OS: CICS, IMS, MVS, UNIX, WINDOWS, WLM, and DEF are supported
- **IBM i** On IBM i: OS400, CICS, and DEF are supported
- **Linux** **AIX** On AIX and Linux: UNIX, WINDOWS, CICS, and DEF are supported
- **Windows** On Windows, UNIX, WINDOWS, CICS, and DEF are supported

**z/OS** **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered.



### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

In a shared queue environment, you can provide a different queue manager name from the one you are using to enter the command. The command server must be enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect is the same as entering the command on every queue manager in the queue sharing group.

### **DESCR(string)**

Plain-text comment. It provides descriptive information about the object when an operator issues the **DISPLAY PROCESS** command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** Use characters from the coded character set identifier (CCSID) for this queue manager. Other characters might be translated incorrectly if the information is sent to another queue manager.

### **ENVRDATA(string)**

A character string that contains environment information pertaining to the application to be started. The maximum length is 128 characters.

The meaning of **ENVRDATA** is determined by the trigger-monitor application. The trigger monitor provided by IBM MQ appends **ENVRDATA** to the parameter list passed to the started application. The parameter list consists of the MQTMC2 structure, followed by one blank, followed by **ENVRDATA** with trailing blanks removed.

#### **Note:**

- ▶ **z/OS** On z/OS, **ENVRDATA** is not used by the trigger-monitor applications provided by IBM MQ.
- ▶ **z/OS** On z/OS, if **APPLTYPE** is WLM, the default values for the ServiceName and ServiceStep fields in the work information header (MQWIH) can be supplied in **ENVRDATA**. The format must be:

```
SERVICENAME=servname, SERVICESTEP=stepname
```

where:

#### **SERVICENAME=**

is the first 12 characters of **ENVRDATA**.

#### **servname**

is a 32-character service name. It can contain embedded blanks or any other data, and have trailing blanks. It is copied to the MQWIH as is.

#### **SERVICESTEP=**

is the next 13 characters of **ENVRDATA**.

#### **stepname**

is a 1 - 8 character service step name. It is copied as-is to the MQWIH, and padded to eight characters with blanks.

If the format is incorrect, the fields in the MQWIH are set to blanks.

- ▶ **Linux** ▶ **AIX** On AIX and Linux, **ENVRDATA** can be set to the ampersand character to make the started application run in the background.

### ▶ **z/OS** **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

<i>Table 138. Behavior for each of the QSGDISP values</i>	
<b>QSGDISP</b>	<b>ALTER</b>
COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (COPY)</b> . Any object residing in the shared repository, or any object defined using a command that had the parameters <b>QSGDISP (QMGR)</b> , is not affected by this command.
GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameters <b>QSGDISP (GROUP)</b>. On the page set of the queue manager that executes the command, only a local copy of the object is altered by this command. If the command is successful, the following command is generated.</p> <pre>DEFINE PROCESS (process-name) REPLACE QSGDISP (COPY)</pre> <p>The command is sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero. The <b>ALTER</b> for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with <b>QSGDISP (QMGR)</b> or <b>QSGDISP (COPY)</b> . Any object residing in the shared repository is unaffected.
QMGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (QMGR)</b> . Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

### **USERDATA**(string)

A character string that contains user information pertaining to the application defined in the **APPLICID** that is to be started. The maximum length is 128 characters.

The meaning of **USERDATA** is determined by the trigger-monitor application. The trigger monitor provided by IBM MQ simply passes **USERDATA** to the started application as part of the parameter list. The parameter list consists of the MQTMC2 structure (containing **USERDATA**), followed by one blank, followed by **ENVRDATA** with trailing blanks removed.

For IBM MQ message channel agents, the format of this field is a channel name of up to 20 characters. See [Managing objects for triggering](#) for information about what **APPLICID** to provide to message channel agents.

**Windows** For Microsoft Windows, the character string must not contain double quotation marks if the process definition is going to be passed to **runmqtrm**.

### **z/OS** **ALTER PSID (change page set expansion method) on z/OS**

Use the MQSC command **ALTER PSID** to change the expansion method for a page set.

### **Using MQSC commands on z/OS**

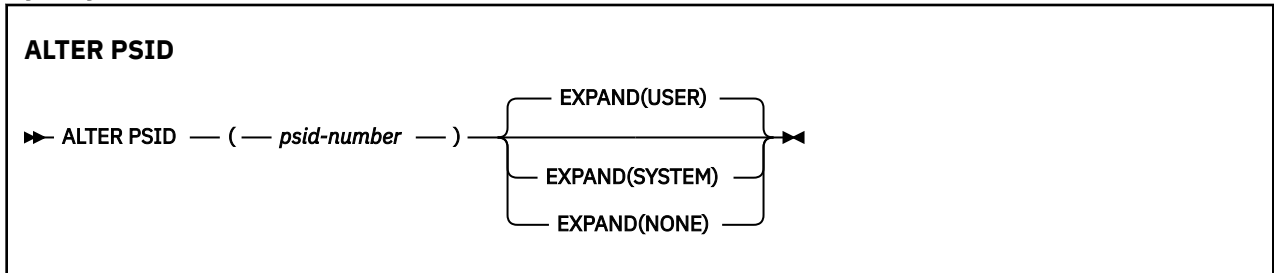
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Parameters not specified in the **ALTER PSID** command result in the existing values for those parameters being left unchanged.

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for ALTER PSID” on page 347](#)

**Synonym:** ALT PSID



## Parameter descriptions for ALTER PSID

### **(*psid-number*)**

Identifier of the page set. This is required.

### **EXPAND**

Controls how the queue manager should expand a page set when it becomes nearly full, and further pages are required in it.

#### **USER**

The secondary extent size that was specified when the page set was defined is used. If no secondary extent size was specified, or if it was specified as zero, then no dynamic page set expansion can take place.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set. Only one extent is required to reach this size.

#### **SYSTEM**

A secondary extent size that is approximately 10 per cent of the current size of the page set is used. It might be rounded up depending on the characteristics of the DASD.

The secondary extent size that was specified when the page set was defined is ignored; dynamic expansion can occur if it was zero or not specified.

#### **NONE**

No further page set expansion is to take place.

## Usage note

You can use **ALTER PSID** to reset an internal IBM MQ indicator that prevents the page set from being expanded; for example, after the data set has been **ALTERed** to **ADDVOLUMES**.

In this instance, although the **EXPAND** keyword must be specified with a value, you do not have to change the value from that already configured. For example, if **DISPLAY USAGE** shows page set 3 configured with **EXPAND(SYSTEM)**, you issue the following command to allow IBM MQ to retry page set expansion:

```
ALTER PSID(3) EXPAND(SYSTEM)
```

## Related reference

[“DISPLAY USAGE \(display usage information\) on z/OS” on page 873](#)

Use the MQSC command DISPLAY USAGE to display information about the current state of a page set, to display information about the log data sets, or to display information about the shared message data sets.

## ALTER QMGR (alter queue manager settings)

Use the MQSC command **ALTER QMGR** to alter the queue manager parameters for the local queue manager.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

Parameters not specified in the **ALTER QMGR** command result in the existing values for those parameters being left unchanged.

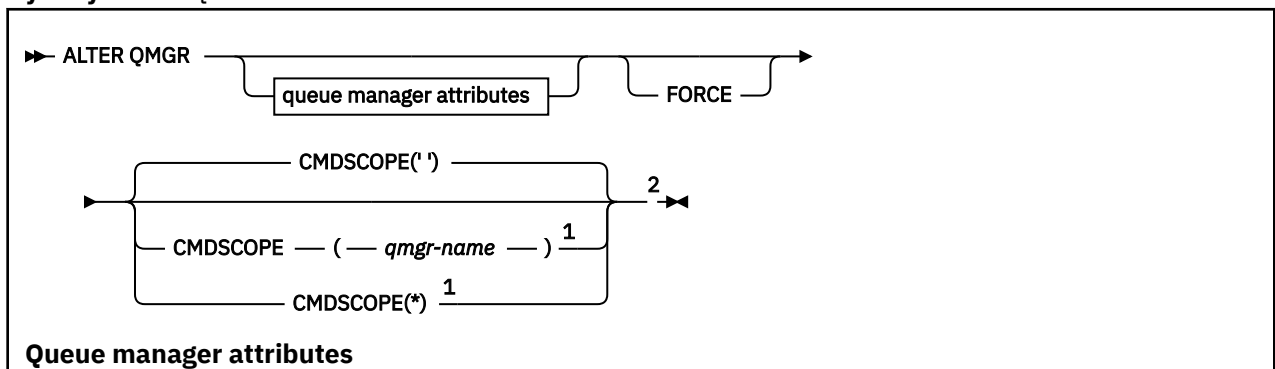
**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

This information is divided into three sections:

- [“ALTER QMGR” on page 348](#)
- [“Parameter descriptions for ALTER QMGR” on page 350](#)
- [“Queue manager parameters” on page 350](#)

## ALTER QMGR

**Synonym:** ALT QMGR





<sup>3</sup> Not valid on z/OS.

<sup>4</sup> Valid only on AIX, Linux, and Windows.

<sup>5</sup> Not valid on IBM i.

## Parameter descriptions for ALTER QMGR

The parameters you specify override the current values. Attributes that you do not specify are unchanged.

### Note:

1. If you do not specify any parameters, the command completes successfully, but no queue manager options are changed.
2. Changes made using this command persist when the queue manager is stopped and restarted.

### FORCE

Specify this parameter to force completion of the command if both of the following statements are true:

- The **DEFXMITQ** parameter is specified
- An application has a remote queue open, the resolution for which would be affected by this change

If **FORCE** is not specified in these circumstances, the command is unsuccessful.

## Queue manager parameters

These parameters are the queue manager parameters for the **ALTER QMGR** command:

**Multi**

### ACCTCONO

Specifies whether applications can override the settings of the **ACCTQ** and **ACCTMQI** queue manager parameters:

#### DISABLED

Applications cannot override the settings of the **ACCTQ** and **ACCTMQI** parameters.

This is the queue manager's initial default value.

#### ENABLED

Applications can override the settings of the **ACCTQ** and **ACCTMQI** parameters by using the options field of the MQCNO structure of the MQCONNX API call.

Changes to this parameter are effective for connections to the queue manager that occur after the change.

This parameter is valid only on [Multiplatforms](#).

**Multi**

### ACCTINT(integer)

The time interval, in seconds, at which intermediate accounting records are written.

Specify a value in the range 1 through 604800.

Changes to this parameter are effective for connections to the queue manager that occur after the change.

This parameter is valid only on [Multiplatforms](#).

**Multi**

### ACCTMQI

Specifies whether accounting information for MQI data is to be collected:

#### OFF

MQI accounting data collection is disabled.

This is the queue manager's initial default value.

**ON**

MQI accounting data collection is enabled.

If queue manager attribute **ACCTCONO** is set to ENABLED, the value of this parameter can be overridden using the options field of the MQCNO structure.

Changes to this parameter are effective for connections to the queue manager that occur after the change.

This parameter is valid only on Multiplatforms.

**ACCTQ**

Specifies whether accounting data is to be collected for all queues.

**z/OS** On z/OS, the data collected is class 3 accounting data (thread-level and queue-level accounting).

**OFF**

Accounting data collection is disabled for all queues which specify QMGR as the value for their ACCTQ parameter.

**ON**

Accounting data collection is enabled for all queues which specify QMGR as the value of their ACCTQ parameter.

**z/OS** On z/OS systems, you must switch on class 3 accounting by the START TRACE command.

**NONE**

Accounting data collection for all queues is disabled regardless of the value of the ACCTQ parameter of the queue.

Changes to this parameter are effective only for connections to the queue manager occurring after the change to the parameter.

**z/OS ACTCHL(integer)**

The maximum number of channels that can be *active* at any time, unless the value is reduced below the number of currently active channels.

Specify a value from 1 through 9999 that is not greater than the value of MAXCHL. MAXCHL defines the maximum number of channels available.

If you change this value, you must also review the MAXCHL, LU62CHL, and TCPCHL values to ensure that there is no conflict of values

For an explanation of which channel states are considered active; see Channel states.

If the value of ACTCHL is reduced to less than its value when the channel initiator was initialized, channels continue to run until they stop. When the number of running channels falls below the value of ACTCHL, more channels can be started. Increasing the value of ACTCHL to more than its value when the channel initiator was initialized does not have immediate effect. The higher value of ACTCHL takes effect at the next channel initiator restart.

Sharing conversations do not contribute to the total for this parameter.

This parameter is valid only on z/OS.

**ACTIVREC**

Specifies whether activity reports are generated if requested in the message:

**DISABLED**

Activity reports are not generated.

**MSG**

Activity reports are generated and sent to the reply queue specified by the originator in the message causing the report.

This is the queue manager's initial default value.

#### **QUEUE**

Activity reports are generated and sent to `SYSTEM.ADMIN.ACTIVITY.QUEUE`

See [Activity recording](#).

#### **Multi**

#### **ACTVCONO**

Specifies whether applications can override the settings of the **ACTVTRC** queue manager parameter:

#### **DISABLED**

Applications cannot override the settings of the **ACTVTRC** queue manager parameter.

This is the queue manager's initial default value.

#### **ENABLED**

Applications can override the settings of the **ACTVTRC** queue manager parameter by using the options field of the MQCNO structure of the MQCONN API call.

Changes to this parameter are effective for connections to the queue manager that occur after the change.

This parameter is valid only on [Multiplatforms](#).

#### **Multi**

#### **ACTVTRC**

Specifies whether MQI application activity tracing information is to be collected. See [Setting ACTVTRC to control collection of activity trace information](#).

#### **OFF**

IBM MQ MQI application activity tracing information collection is not enabled.

This is the queue manager's initial default value.

#### **ON**

IBM MQ MQI application activity tracing information collection is enabled.

If the queue manager attribute **ACTVCONO** is set to **ENABLED**, the value of this parameter can be overridden using the options field of the MQCNO structure.

Changes to this parameter are effective for connections to the queue manager that occur after the change.

This parameter is valid only on [Multiplatforms](#).

#### **z/OS**

#### **ADOPTCHK**

Specifies which elements are checked to determine whether an MCA is adopted. The check is made when a new inbound channel is detected with the same name as an already active MCA.

#### **ALL**

Check the queue manager name and the network address. Perform this check to prevent your channels from being inadvertently or maliciously shut down.

This is the queue manager's initial default value.

#### **NETADDR**

Check the network address.

#### **NONE**

Do no checking.

#### **QMNAME**

Check the queue manager name.

Changes to this parameter take effect the next time that a channel attempts to adopt an MCA.

This parameter is valid only on [z/OS](#).



**z/OS ADOPTMCA**

Specifies whether an orphaned instance of an MCA restarts immediately when a new inbound channel request matching the **ADOPTCHK** parameter is detected:

**ALL**

Adopt all channel types.

This is the queue manager's initial default value.

**NO**

Adoption of orphaned channels is not required.

Changes to this parameter take effect the next time that a channel attempts to adopt an MCA.

This parameter is valid only on z/OS.

**AUTHOREV**

Specifies whether authorization (Not Authorized) events are generated:

**DISABLED**

Authorization events are not generated.

This is the queue manager's initial default value.

**ENABLED**

Authorization events are generated.

**z/OS** This value is not supported on z/OS.

**z/OS BRIDGEEV**

Specifies whether IMS bridge events are generated.

**DISABLED**

IMS bridge events are not generated.

This is the queue manager's initial default value.

**ENABLED**

All IMS bridge events are generated.

This parameter is valid only on z/OS.

**Multi CCSID(integer)**

The coded character set identifier for the queue manager. The CCSID is the identifier used with all character string fields defined by the API. If the CCSID in the message descriptor is set to the value MQCCSI\_Q\_MGR, the value applies to application data in the body of a message. The value is set when the message is put to a queue.

Specify a value in the range 1 through 65535. The CCSID specifies a value that is defined for use on your platform, and use a character set that is appropriate to the platform.

If you use this parameter to change the CCSID, applications that are running when the change is applied continue to use the original CCSID. Therefore, stop and restart all running applications before you continue including the command server and channel programs. To stop and restart all running applications, stop and restart the queue manager after changing the parameter value.

This parameter is valid only on Multiplatforms. See Code page conversion for details of the supported CCSIDs for each platform.

**z/OS** To carry out the equivalent tasks on z/OS, use CSQ6SYSP to set your system parameters.

**CERTLABL**

Certificate label for this queue manager to use. The label identifies which personal certificate in the key repository has been selected.

The default and migrated queue manager values are:

- **ALW** On AIX, Linux, and Windows: *ibmwebspheremqxxxx* where *xxxx* is the queue manager name folded to lowercase.
- **IBM i** On IBM i:
  - If you specified `SSLKEYR(*SYSTEM)`, the value is blank.  
Note that it is forbidden to use a nonblank queue manager `CERTLABL` with `SSLKEYR(*SYSTEM)`. Attempting to do so results in an `MQRCCF_Q_MGR_ATTR_CONFLICT` error.
  - Otherwise, *ibmwebspheremqxxxx* where *xxxx* is the queue manager name folded to lowercase.
- **z/OS** On z/OS: *ibmWebSphereMQXXXX* where *XXXX* is the queue manager name.  
See [z/OS systems](#) for more information.

You should specify the preceding values. However, leaving **CERTLABL** as a blank value on the queue manager is interpreted by the system to mean the default values specified.

**Important:** You must run a `REFRESH SECURITY TYPE(SSL)` command if you make any changes to **CERTLABL** on the queue manager. However, you do not need to run the `REFRESH SECURITY TYPE(SSL)` command if you make any changes to **CERTLABL** on a channel.

### **z/OS** **CERTQSG**

Queue sharing group (QSG) certificate label.

This parameter takes precedence over **CERTLABL** in the event that the queue manager is a member of a QSG.

The default value for this parameter is *ibmWebSphereMQXXXX* where *XXXX* is the queue sharing group name.

This parameter is valid only on z/OS.

See [z/OS systems](#) for more information.

### **Multi** **CERTVPOL**

Specifies which TLS certificate validation policy is used to validate digital certificates received from remote partner systems. This attribute can be used to control how strictly the certificate chain validation conforms to industry security standards.

#### **ANY**

Apply each of the certificate validation policies supported by the secure sockets library and accept the certificate chain if any of the policies considers the certificate chain valid. This setting can be used for maximum backwards compatibility with older digital certificates which do not comply with the modern certificate standards.

#### **RFC5280**

Apply only the RFC 5280 compliant certificate validation policy. This setting provides stricter validation than the ANY setting, but rejects some older digital certificates.

For more information about certificate validation policies, see [Certificate validation policies in IBM MQ](#).

Changes to the parameter take effect only after a **REFRESH SECURITY TYPE(SSL)** command is issued.

This parameter is valid only on [Multiplatforms](#).

### **z/OS** **CFCONLOS**

Specifies the action to be taken when the queue manager loses connectivity to the administration structure, or any CF structure with **CFCONLOS** set to `ASQMGR`.

#### **TERMINATE**

The queue manager terminates when connectivity to CF structures is lost.

## TOLERATE

The queue manager tolerates loss of connectivity to CF structures without terminating.

This parameter is valid only on z/OS.

### Multi CHAD

Specifies whether receiver and server-connection channels can be defined automatically:

#### DISABLED

Auto-definition is not used.

This is the queue manager's initial default value.

#### ENABLED

Auto-definition is used.

Cluster-sender channels can always be defined automatically, regardless of the setting of this parameter.

This parameter is valid only on [Multiplatforms](#).

### Multi CHADEV

Specifies whether channel auto-definition events are generated.

#### DISABLED

Auto-definition events are not generated.

This is the queue manager's initial default value.

#### ENABLED

Auto-definition events are generated.

This parameter is valid only on [Multiplatforms](#).

## CHADEXIT(string)

Auto-definition exit name.

If this name is nonblank, the exit is called when an inbound request for an undefined receiver, server-connection, or cluster-sender channel is received. It is also called when starting a cluster-receiver channel.

The format and maximum length of the name depends on the environment:

- ▶ **Linux** ▶ **AIX** On AIX and Linux, it is of the form *libraryname(functionname)*. The maximum length is 128 characters.
- ▶ **Windows** On Windows, it is of the form *dllname(functionname)* where *dllname* is specified without the suffix .DLL. The maximum length is 128 characters.
- ▶ **IBM i** On IBM i, it is of the form:

```
progrname libname
```

where *program name* occupies the first 10 characters and *libname* the second 10 characters (both blank-padded to the right if necessary). The maximum length of the string is 20 characters.

- ▶ **z/OS** On z/OS, it is a load module name, the maximum length is eight characters.

▶ **z/OS** On z/OS, the **CHADEXIT** parameter applies only to cluster-sender and cluster-receiver channels.

### z/OS CHIADAPS(integer)

The number of channel initiator adapter subtasks to use for processing IBM MQ calls.

Specify a value in the range 0 - 9999. Suggested settings are:

- Test system: 8
- Production system: 30

Changes to this parameter take effect when the channel initiator is restarted.

For more information about the relationship between CHIADAPS, CHIDISPS and MAXCHL, see [Tailor the channel initiator parameters](#).

This parameter is valid only on z/OS.

#### ▶ z/OS **CHIDISPS (integer)**

The number of dispatchers to use in the channel initiator.

Specify a value in the range 1 through 9999. Suggested settings are:

- Test system: 5
- Production system: 20

Changes to this parameter take effect when the channel initiator is restarted.

For more information about the relationship between CHIADAPS, CHIDISPS and MAXCHL, see [Tailor the channel initiator parameters](#).

This parameter is valid only on z/OS.

#### ▶ z/OS **CHISERVP**

This parameter is reserved for IBM use only; it is not for general use.

This parameter is valid only on z/OS.

### **CHLAUTH**

Specifies whether the rules defined by channel authentication records are used. CHLAUTH rules can still be set and displayed regardless of the value of this attribute.

Changes to this parameter take effect the next time that an inbound channel attempts to start. Channels that are currently started are unaffected by changes to this parameter.

#### **DISABLED**

Channel authentication records are not checked.

#### **ENABLED**

Channel authentication records are checked.

### **CHLEV**

Specifies whether channel events are generated.

#### **DISABLED**

Channel events are not generated. This is the queue manager's initial default value.

#### **ENABLED**

All channel events are generated.

#### **EXCEPTION**

All exception channel events are generated.

### **CLWLDATA(string)**

Cluster workload exit data. The maximum length of the string is 32 characters.

This string is passed to the cluster workload exit when it is called.

### **CLWLEXIT(string)**

Cluster workload exit name.

If this name is nonblank, the exit is called when a message is put to a cluster queue. The format and maximum length of the name depends on the environment:

- ▶ Linux ▶ AIX On AIX and Linux, it is of the form *libraryname(functionname)*. The maximum length is 128 characters.

- **Windows** On Windows, it is of the form *dllname(functionname)*, where *dllname* is specified without the suffix .DLL. The maximum length is 128 characters.
- **z/OS** On z/OS, it is a load module name. The maximum length is eight characters.
- **IBM i** On IBM i, it is of the form:

```
progrname libname
```

where *program name* occupies the first 10 characters and *libname* the second 10 characters (both blank-padded to the right if necessary). The maximum length is 20 characters.

### **CLWLLEN(integer)**

The maximum number of bytes of message data that is passed to the cluster workload exit.

Specify a value in the range:

- **ALW** 0 - 999,999,999 on AIX, Linux, and Windows.
- **IBM i** 0 - 999,999,999 on IBM i.
- **z/OS** 0 - 100 MB on z/OS systems.

### **CLWLMRUC(integer)**

The maximum number of most recently used outbound cluster channels.

Specify a value in the range 1 through 999,999,999.

See [CLWLMRUC queue manager attribute](#).

### **CLWLUSEQ**

The attribute applies to queues with the queue attribute **CLWLUSEQ** set to QMGR. It specifies the behavior of an MQPUT operation when the target queue has a local instance and at least one remote cluster instance. It does not apply if the MQPUT originates from a cluster channel.

Specify either:

#### **LOCAL**

The local queue is the only target for MQPUT operations.

This is the queue manager's initial default value.

#### **ANY**

The queue manager treats the local queue as another instance of the cluster queue for the purposes of workload distribution.

See [CLWLUSEQ queue manager attribute](#).

### **CMDEV**

Specifies whether command events are generated:

#### **DISABLED**

Command events are not generated.

This is the queue manager's initial default value.

#### **ENABLED**

Command events are generated for all successful commands.

#### **NODISPLAY**

Command events are generated for all successful commands, other than DISPLAY commands.

### **z/OS CMDSCOPE**

Specifies how the command is run when the queue manager is a member of a queue sharing group.

,

The command is run on the queue manager on which it was entered.

### **qmgr-name**

The command is run on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a different queue manager. You can do so if you are using a queue sharing group environment, and if the command server is enabled. You can then specify a different queue manager to the one on which the command is entered.

**\***

The command is run on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of entering this value is the same as entering the command on every queue manager in the queue sharing group.

This parameter is valid only on z/OS.

### **CONFIGEV**

Specifies whether configuration events are generated:

#### **ENABLED**

Configuration events are generated. After setting this value, issue **REFRESH QMGR TYPE (CONFIGEV)** commands for all objects to bring the queue manager configuration up to date.

#### **DISABLED**



Configuration events are not generated.

This is the queue manager's initial default value.

### **CONNAUTH**

The name of an authentication information object that is used to provide the location of user ID and password authentication. If **CONNAUTH** is blank, no user ID and password checking is done by the queue manager. The maximum length of the string is **MQ\_AUTH\_INFO\_NAME\_LENGTH**.

Only authentication information objects with type **IDPWOS** or **IDPWLDAP** can be specified; other types result in an error message when:

-  **Multi** The OAM reads the configuration on [Multiplatforms](#).
-  **z/OS** The security component reads the configuration on z/OS.

Changes to this configuration, or the object to which it refers, take effect when a **REFRESH SECURITY TYPE (CONNAUTH)** command is issued.

If you leave **CONNAUTH** blank, and attempt to connect to a channel (matching a **CHLAUTH** record) set in the **CHKCLNT** field, the connection fails:

-  **Multi** REQDADM
-  **z/OS** REQUIRED

### **CUSTOM(string)**

The custom attribute for new features.

This attribute is reserved for the configuration of new features before named attributes are introduced. It can contain the values of zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form **NAME (VALUE)**. Escape a single quotation mark with another single quotation mark.

No values are defined for **Custom**.

### **DEADQ(string)**

The local name of a dead-letter queue (or undelivered-message queue) on which messages that cannot be routed to their correct destination are put.

The queue named must be a local queue; see [Rules for naming IBM MQ objects](#).

## DEFCLXQ

The **DEFCLXQ** attribute controls which transmission queue is selected by default by cluster-sender channels to get messages from, to send the messages to cluster-receiver channels.

## SCTQ

All cluster-sender channels send messages from `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. The `correlID` of messages placed on the transmission queue identifies which cluster-sender channel the message is destined for.

SCTQ is set when a queue manager is defined. **DEFCLXQ** was not present.

## CHANNEL

Each cluster-sender channel sends messages from a different transmission queue. Each transmission queue is created as a permanent dynamic queue from the model queue `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`.

If the queue manager attribute, **DEFCLXQ**, is set to CHANNEL, the default configuration is changed to cluster-sender channels being associated with individual cluster transmission queues. The transmission queues are permanent-dynamic queues created from the model queue `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`. Each transmission queue is associated with one cluster-sender channel. As one cluster-sender channel services a cluster transmission queue, the transmission queue contains messages for only one queue manager in one cluster. You can configure clusters so that each queue manager in a cluster contains only one cluster queue. In this case, the message traffic from a queue manager to each cluster queue is transferred separately from messages to other queues.

## DEFXMITQ(*string*)

Local name of the default transmission queue on which messages destined for a remote queue manager are put. The default transmission queue is used if there is no other suitable transmission queue defined.

The cluster transmission queue must not be used as the default transmission queue of the queue manager.

The queue named must be a local transmission queue; see [Rules for naming IBM MQ objects](#).

## DESCR(*string*)

Plain-text comment. It provides descriptive information about the queue manager.

It contains only displayable characters. The maximum length of the string is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

If the characters in the descriptive information are in the coded character set identifier (CCSID) for this queue manager they are translated correctly. They are translated when the descriptive information is sent to another queue manager. If they are not in the CCSID for this queue manager, they might be translated incorrectly.

## **DNSGROUP(*string*)**

This parameter is no longer used. See [z/OS: WLM/DNS no longer supported](#).

## **DNSWLM**

This parameter is no longer used. See [z/OS: WLM/DNS no longer supported](#).

## **NO**

This value is the only value accepted.

## **EXPRYINT**

Specifies how often queues are scanned to discard expired messages:

## **OFF**

Queues are not scanned. No internal expiry processing is performed.

### **integer**

The approximate interval in seconds at which queues are scanned. Each time that the expiry interval is reached, the queue manager looks for candidate queues that are worth scanning to discard expired messages.

The queue manager maintains information about the expired messages on each queue, and therefore whether a scan for expired messages is worthwhile. So, only a selection of queues is scanned at any time.

The value must be in the range 1 through 99999999. The minimum scan interval used is 5 seconds, even if you specify a lower value.

You must set the same **EXPRYINT** value for all queue managers within a queue sharing group that support this attribute. Shared queues are scanned by only one queue manager in a queue sharing group. This queue manager is either the first queue manager to restart, or the first queue manager for which **EXPRYINT** is set.

Changes to **EXPRYINT** take effect when the current interval expires. Changes also take effect if the new interval is less than the unexpired portion of the current interval. In this case, a scan is scheduled and the new interval value takes immediate effect.

This parameter is valid only on z/OS.

### **z/OS GROUPUR**

This parameter controls whether CICS and XA client applications can establish transactions with a GROUP unit of recovery disposition.

The property can be enabled only when the queue manager is a member of a queue sharing group.

#### **ENABLED**

CICS and XA client applications can establish transactions with a group unit of recovery disposition by specifying a queue sharing group name when they connect.

#### **DISABLED**

CICS and XA client applications must connect using a queue manager name.

This parameter is valid only on z/OS.

### **z/OS IGQ**

Specifies whether intra-group queuing is used.

The **IGQ** parameter is valid only on z/OS when the queue manager is a member of a queue sharing group.

#### **ENABLED**

Message transfer between queue managers within a queue sharing group uses the shared transmission queue, `SYSTEM.QSG.TRANSMIT.QUEUE`.

#### **DISABLED**

Message transfer between queue managers within a queue sharing group uses non-shared transmission queues and channels. Queue managers that are not part of a queue sharing group also use this mechanism.

If intra-group queuing is enabled, but the intra-group queuing agent is stopped, use the following command to restart it:

```
ALTER QMGR IGQ(ENABLED)
```

This parameter is valid only on z/OS.

### **z/OS IGQAUT**

Specifies the type of authority checking and, therefore, the user IDs, to be used by the IGQ agent (IGQA). This parameter establishes the authority to put messages to a destination queue.

The **IGQAUT** parameter is valid only on z/OS when the queue manager is a member of a queue sharing group.



## DEF

Indicates that the default user ID is used to establish authority to put messages to a destination queue.

For a one user ID check, the default user ID is the user ID of a queue manager within the queue sharing group. The default user ID is the user ID of the queue manager that put the messages to the `SYSTEM.QSG.TRANSMIT.QUEUE`. This user ID is referred to as the QSGSEND user ID.

For two user ID checks, the default second user ID is the IGQ user ID.

## CTX

Indicates that the user ID from a *UserIdentifier* field is used to establish authority to put messages to a destination queue. The user ID is the *UserIdentifier* field in the message descriptor of a message on the `SYSTEM.QSG.TRANSMIT.QUEUE`.

For one user ID check, the QSGSEND user ID is used.

For two user ID checks, the QSGSEND user ID, the IGQ user ID and the alternate user ID are used. The alternate user ID is taken from the *UserIdentifier* field in the message descriptor of a message on the `SYSTEM.QSG.TRANSMIT.QUEUE`. The alternate user ID is referred to as ALT.

## ONLYIGQ

Indicates that only the IGQ user ID is used to establish authority to put messages to a destination queue.

For all ID checks, the IGQ user ID is used.

## ALTIGQ

Indicates that the IGQ user ID and the ALT user ID are used to establish authority to put messages to a destination queue.

For one user ID check, the IGQ user ID is used.

For two user ID checks, the IGQ user ID and the ALT user ID are used.

This parameter is valid only on z/OS.

## **IGQUSER**

Nominates a user ID to be used by the IGQ agent (IGQA) to establish authority to put messages to a destination queue. The user ID is referred to as the IGQ user ID.

This parameter is valid only on z/OS when the queue manager is a member of a queue sharing group. Possible values are:

### Blanks

Indicates that the user ID of the receiving queue manager within the queue sharing group is used.

### Specific user ID

Indicates that the user ID specified in the **IGQUSER** parameter of the receiving queue manager is used.

### Note:

1. As the receiving queue manager has authority to all queues it can access, security checking might not be performed for this user ID type.
2. As the value of blanks has a special meaning, you cannot use IGQUSER to specify a real user ID of blanks.

This parameter is valid only on z/OS.

## **IMGINTVL**

The target frequency with which the queue manager automatically writes media images, in minutes since the previous media image for the object.

Possible values are:

### **1 - 999 999 999**

The time in minutes at which the queue manager automatically writes media images.

The default value is 60 minutes.

### **OFF**

Automatic media images are not written on a time interval basis.

**V 9.3.4** The interval specified by **IMGINTVL** is honored when enough new work has been carried out on the queue manager to make it worthwhile recording a new image. Otherwise the taking of new images is delayed.

This parameter is valid only on [Multiplatforms](#).

### **Multi** **IMGLOGLN**

The target size of recovery log, written before the queue manager automatically writes media images, in number of megabytes since the previous media image for the object. This limits the amount of log to be read when recovering an object.

Possible values are:

### **1 - 999 999 999**

The target size of the recovery log in megabytes.

### **OFF**

Automatic media images are not written based on the size of log written.

OFF is the default value (except when creating a Native HA queue manager).

**V 9.3.4** Native HA queue managers are created with **IMGLOGLN** set to the value of 25% of the available space on the volume where the recovery logs are to be written. This value is calculated at queue manager creation time, but can be altered subsequently, if required.

This parameter is valid only on [Multiplatforms](#).

### **Multi** **IMGRCOVO**

Specifies whether authentication information, channel, client connection, listener, namelist, process, alias queue, remote queue, and service objects are recoverable from a media image, if linear logging is being used.

Possible values are:

### **NO**

The “[rcdmqimg \(record media image\)](#)” on page 139 and “[rcrmqobj \(re-create object\)](#)” on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

### **YES**

These objects are recoverable.

YES is the default value.

This parameter is valid only on [Multiplatforms](#).

### **Multi** **IMGRCOVQ**

Specifies the default **IMGRCOVQ** attribute for local and permanent dynamic queue objects, when used with this parameter.

Possible values are:

### **NO**

The **IMGRCOVQ** attribute for local and permanent dynamic queue objects is set to NO.

### **YES**

The **IMGRCOVQ** attribute for local and permanent dynamic queue objects is set to YES.

YES is the default value.

This parameter is valid only on [Multiplatforms](#).

### Multi **IMGSCHED**

Whether the queue manager automatically writes media images.

Possible values are:

#### **AUTO**

The queue manager attempts to automatically write a media image for an object, before **IMGINTVL** minutes have elapsed, or **IMGLOGLN** megabytes of recovery log have been written, since the previous media image for the object was taken.

The previous media image might have been taken manually or automatically, depending on the settings of **IMGINTVL** or **IMGLOGLN**.

**V 9.3.4** The interval specified by **IMGINTVL** is honored when enough new work has been carried out on the queue manager to make it worthwhile recording a new image. Otherwise the taking of new images is delayed.

#### **MANUAL**

Automatic media images are not written.

MANUAL is the default value.

This parameter is valid only on [Multiplatforms](#).

### **INHIBTEV**

Specifies whether inhibit events are generated. The events are generated for Inhibit Get and Inhibit Put.

#### **ENABLED**

Inhibit events are generated.

#### **DISABLED**

Inhibit events are not generated.

This is the queue manager's initial default value.

### Multi **V 9.3.0** **V 9.3.0** **INITKEY**

The initial key for the password protection system. IBM MQ encrypts the value of some queue manager attributes using the IBM MQ password protection system.

An initial key is used by the encryption algorithm to encrypt and decrypt these attributes. You should use this attribute to set a unique initial key for the queue manager, prior to setting a values of the attributes that are encrypted.

The maximum length of the initial key is 256 bytes.

If an initial key is not set using this attribute, a default initial key is used.



**Warning:** If you change this attribute, the values of the attributes that are encrypted are invalidated and must be reset before they can be used.

### **IPADDRV**

Specifies which IP protocol is to be used for channel connections.

#### **IPV4**

The IPv4 IP address is to be used.

This is the queue manager's initial default value.

#### **IPV6**

The IPv6 IP address is to be used.

This parameter is used only in systems running IPv4 and IPv6. It applies to channels defined only with a **TRPTYPE** of TCP when either of the following two conditions is true:

- The **CONNAME** parameter of the channel contains a host name that resolves to both an IPv4 and an IPv6 address, and the **LOCLADDR** parameter is not specified.

- The value of the **CONNNAME** and **LOCLADDR** parameters of the channel is a host name that resolves to both an IPv4 and IPv6 address.

These values have the same meanings as the values set in the **IPAddressVersion** attribute of the TCP stanza of the client configuration file and the **MQIPADDRV** environment variable.

#### V 9.3.0 ALW V 9.3.0 **KEYRPWD**

The password to access the TLS key repository.

If a value is specified for this attribute, it is used as the password to access the Secure Sockets Layer key repository. If this attribute is blank, the stash file that is associated with the key repository is used. If the stash file is not present or cannot be read, the key repository cannot be accessed and channels using TLS fail to start.

You should set **INITKEY** to a unique value for the queue manager before this attribute is set. The default value is blank.

#### **LOCALEV**

Specifies whether local error events are generated, caused by an application or the queue manager not being able to access a local queue or other local object, for example, because the object has not been defined:

##### **ENABLED**

Local error events are generated.

##### **DISABLED**

Local error events are not generated.

This is the queue manager's initial default value.

#### Multi **LOGGEREV**

Specifies whether recovery log events are generated:

##### **DISABLED**

Logger events are not generated.

This is the queue manager's initial default value.

##### **ENABLED**

Logger events are generated.

This parameter is valid only on Multiplatforms.

#### z/OS **LSTRTMR(integer)**

The time interval, in seconds, between attempts by IBM MQ to restart a listener after an APPC or TCP/IP failure. When the listener is restarted on TCP/IP, it uses the same port and IP address as it used when it first started.

Specify a value in the range 5 through 9999.

Changes to this parameter take effect for listeners that are later started. Listeners that are currently started are unaffected by changes to this parameter.

This parameter is valid only on z/OS.

#### z/OS **LUGROUP(string)**

The generic LU name to be used by the LU 6.2 listener that handles inbound transmissions for the queue sharing group. The maximum length of this parameter is eight characters.

If this name is blank, the listener cannot be used.

Changes to this parameter take effect for listeners that are later started. Listeners that are currently started are unaffected by changes to this parameter.

This parameter is valid only on z/OS.

z/OS

### LUNAME(*string*)

The name of the LU to use for outbound LU 6.2 transmissions. Set this parameter to be the same as the name of the LU to be used by the listener for inbound transmissions. The maximum length of this parameter is eight characters.

If this name is blank, the APPC/MVS default LU name is used. This name is variable, so LUNAME must always be set if you are using LU 6.2

Changes to this parameter take effect when the channel initiator is restarted.

This parameter is valid only on z/OS.

z/OS

### LU62ARM(*string*)

The suffix of the APPCPM member of SYS1.PARMLIB. This suffix nominates the LUADD for this channel initiator. When automatic restart manager (ARM) restarts the channel initiator, the z/OS command SET APPC= *xx* is issued.

If you do not provide a value for this parameter, no SET APPC= *xx* command is issued.

The maximum length of this parameter is two characters.

Changes to this parameter take effect when the channel initiator is restarted.

This parameter is valid only on z/OS.

z/OS

### LU62CHL(*integer*)

The maximum number of channels that can be current, or clients that can be connected, that use the LU 6.2 transmission protocol.

Specify a value 0- 9999 that is not greater than the value of MAXCHL. MAXCHL defines the maximum number of channels available. If you specify zero, the LU 6.2 transmission protocol is not used.

If you change this value, also review the MAXCHL, LU62CHL, and ACTCHL values. Ensure that there is no conflict of values and if necessary, raise the value of MAXCHL and ACTCHL.

If the value of this parameter is reduced, any current channels that exceed the new limit continue to run until they stop.

If the value of **LU62CHL** is non-zero when the channel initiator starts up, the value can be modified dynamically. If the value of **LU62CHL** is zero when the channel initiator starts up, a later ALTER command does not take effect. In this case, you should carry out an ALTER command, either before the channel initiator starts, or in CSQINP2 before you issue the **START CHINIT** command.

This parameter is valid only on z/OS.

### MARKINT(*integer*)

The time interval, expressed in milliseconds, for which messages marked as browsed by a call to MQGET, with the get message option MQGMO\_MARK\_BROWSE\_CO\_OP, are expected to remain marked-browsed.

If messages are marked for more than approximately **MARKINT** milliseconds, the queue manager might automatically unmark messages. It might unmark messages that are marked as browsed for the cooperating set of handles.

This parameter does not affect the state of any message marked as browse by a call to MQGET with the get message option MQGMO\_MARK\_BROWSE\_HANDLE.

Specify a value up to the maximum of 999,999,999. The default value is 5000.



**Attention:** You should not reduce the value below the default of 5000.

The special value NOLIMIT indicates that the queue manager does not automatically unmark messages by this process.

## **MAXCHL(integer)**

The maximum number of channels that can be *current* (including server-connection channels with connected clients).

Specify a value in the range 1- 9999. If you change this value, also review the **TCPCHL**, **LU62CHL**, and **ACTCHL** values to ensure that there is no conflict of values. If necessary, increase the number of active channels with the **ACTCHL** value. The values of **ACTCHL**, **LU62CHL**, and **TCPCHL** must not be greater than the maximum number of channels. Suggested settings are:

- Test system: 200
- Production system: 1000

For an explanation of which channel states are considered current; see [Channel states](#).

If the value of this parameter is reduced, any current channels that exceed the new limit continue to run until they stop.

If the value of MAXCHL is reduced to less than its value when the channel initiator was initialized, channels continue to run until they stop. When the number of running channels falls below the value of MAXCHL, more channels can be started. Increasing the value of MAXCHL to more than its value when the channel initiator was initialized does not have immediate effect. The higher value of MAXCHL takes effect at the next channel initiator restart.

Sharing conversations do not contribute to the total for this parameter.

For more information about the relationship between **CHIADAPS**, **CHIDISPS**, and **MAXCHL**, see [Tailor the channel initiator parameters](#).

This parameter is valid only on z/OS.

## **MAXHANDS(integer)**

The maximum number of open handles that any one connection can have at the same time.

This value is a value in the range 0 - 999,999,999.

## **MAXMSGL(integer)**

The maximum length of messages allowed on queues for this queue manager.

This value is in the range 32 KB through 100 MB.

Ensure that you also consider the length of any message properties when deciding the value for the MAXMSGL parameter of a channel.

If you reduce the maximum message length for the queue manager, you must also reduce the maximum message length of the SYSTEM.DEFAULT.LOCAL.QUEUE definition. You must also reduce the maximum message length for all other queues defined on the queue manager. This change ensures that the limit of the queue manager is not less than the limit of any of the queues associated with it. If you do not change these lengths, and applications inquire only the **MAXMSGL** value of the queue, they might not work correctly.

Note that by adding the digital signature and key to the message, [Advanced Message Security](#) increases the length of the message.

## **MAXPROPL ( integer )**

The maximum length of property data in bytes that can be associated with a message.

This value is in the range 0 through 100 MB (104 857 600 bytes).

The special value NOLIMIT indicates that the size of the properties is not restricted, except by the upper limit.


## **MAXUMSGS(integer)**

The maximum number of uncommitted messages within a sync point.

**MAXUMSGS** is a limit on the number of messages that can be retrieved, plus the number of messages that can be put, within any single sync point. The limit does not apply to messages that are put or retrieved outside sync point.

The number includes any trigger messages and report messages generated within the same unit of recovery.

If existing applications and queue manager processes are putting and getting a larger number of messages in sync point, reducing **MAXUMSGS** might cause problems.

 An example of queue manager processes that might be affected is clustering on z/OS.

Specify a value in the range 1 through 999,999,999. The default value is 10000.

MAXUMSGS has no effect on MQ Telemetry. MQ Telemetry tries to batch requests to subscribe, unsubscribe, send, and receive messages from multiple clients into batches of work within a transaction.

## MONACLS

Controls the collection of online monitoring data for auto-defined cluster-sender channels:

### QMGR

Collection of online monitoring data is inherited from the setting of the **MONCHL** parameter of the queue manager.

This is the queue manager's initial default value.

### OFF

Monitoring for the channel is disabled.

### LOW

Unless **MONCHL** is NONE, monitoring is enabled with a low rate of data collection with a minimal effect on system performance. The data collected is not likely to be the most current.

### MEDIUM

Unless **MONCHL** is NONE, monitoring is enabled with a moderate rate of data collection with limited effect on system performance.

### HIGH

Unless **MONCHL** is NONE, monitoring is enabled with a high rate of data collection with a likely effect on system performance. The data collected is the most current available.

A change to this parameter takes effect only on channels started after the change occurs. Any channel started before the change to the parameter continues with the value in force at the time that the channel started.

## MONCHL

Controls the collection of online monitoring data for channels. The channels defined with **MONCHL (QMGR)** are affected by changing the QMGR **MONCHL** attribute.

### OFF

Online monitoring data collection is turned off for channels specifying a value of QMGR in their **MONCHL** parameter.

This is the queue manager's initial default value.

### NONE

Online monitoring data collection is turned off for channels regardless of the setting of their **MONCHL** parameter.

### LOW

Online monitoring data collection is turned on, with a low ratio of data collection, for channels specifying a value of QMGR in their **MONCHL** parameter.

### MEDIUM

Online monitoring data collection is turned on, with a moderate ratio of data collection, for channels specifying a value of QMGR in their **MONCHL** parameter.

### HIGH

Online monitoring data collection is turned on, with a high ratio of data collection, for channels specifying a value of QMGR in their **MONCHL** parameter.

A change to this parameter takes effect only on channels started after the change occurs. Any channel started before the change to the parameter continues with the value in force at the time that the channel started.

## **MONQ**

Controls the collection of online monitoring data for queues.

### **OFF**

Online monitoring data collection is turned off for queues specifying a value of QMGR in their **MONQ** parameter.

This is the queue manager's initial default value.

### **NONE**

Online monitoring data collection is turned off for queues regardless of the setting of their **MONQ** parameter.

### **LOW**

Online monitoring data collection is turned on for queues specifying a value of QMGR in their **MONQ** parameter.

### **MEDIUM**

Online monitoring data collection is turned on for queues specifying a value of QMGR in their **MONQ** parameter.

### **HIGH**

Online monitoring data collection is turned on for queues specifying a value of QMGR in their **MONQ** parameter.

In contrast to **MONCHL**, there is no distinction between the values LOW, MEDIUM, and HIGH. These values all turn data collection on, but do not affect the rate of collection.

Changes to this parameter are effective only for queues opened after the parameter is changed.

## **OPORTMAX(integer)**

The maximum value in the range of port numbers to be used when binding outgoing channels. When all the port numbers in the specified range are used, outgoing channels bind to any available port number.

Specify a value in the range 0 - 65535. A value of zero means that all outgoing channels bind to any available port number.

Specify a corresponding value for **OPORTMIN** to define a range of port numbers. Ensure that the value you specify for **OPORTMAX** is greater than or equal to the value you specify for **OPORTMIN**.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

This parameter is valid only on z/OS.

## **OPORTMIN(integer)**

The minimum value in the range of port numbers to be used when binding outgoing channels. When all the port numbers in the specified range are used, outgoing channels bind to any available port number.

Specify a value in the range 0 - 65535.

Specify a corresponding value for **OPORTMAX** to define a range of port numbers. Ensure that the value you specify for **OPORTMIN** is less than or equal to the value you specify for **OPORTMAX**.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

This parameter is valid only on z/OS.



## **PARENT(parentname)**

The name of the parent queue manager to which the local queue manager is to connect as its child in a hierarchy.

A blank value indicates that the queue manager has no parent queue manager.

If there is an existing parent queue manager it is disconnected.

IBM MQ hierarchical connections require that the queue manager attribute **PSMODE** is set to **ENABLED**.

The value of **PARENT** can be set to a blank value if **PSMODE** is set to **DISABLED**.

Before a queue manager can connect to a queue manager as its child in a hierarchy, channels must exist in both directions. The channels must exist between the parent queue manager and the child queue manager.

If a parent is already defined, the **ALTER QMGR PARENT** command disconnects from the original parent and sends a connection flow to the new parent queue manager.

Successful completion of the command does not mean that the action completed. To check that this command has completed, see the [ALTER QMGR](#) step in [Checking that async commands for distributed networks have finished](#).

## **PERFMEV**

Specifies whether performance-related events are generated:


### **ENABLED**

Performance-related events are generated.

### **DISABLED**

Performance-related events are not generated.

This is the queue manager's initial default value.

 On IBM MQ for z/OS, all the queue managers in a queue sharing group must have the same setting.

## **PSCLUS**

Controls whether this queue manager participates in publish/subscribe activity across any clusters in which it is a member. No clustered topic objects can exist in any cluster when modifying from **ENABLED** to **DISABLED**.

For more information about **PSCLUS**, see [Inhibiting clustered publish/subscribe](#).

**Note:** To change a **PSCLUS** parameter status, the CHIN address space needs to be running.

### **ENABLED**

This queue manager can define clustered topic objects, publish to subscribers on other queue managers, and register subscriptions that receive publications from other queue managers. All queue managers in the cluster running a version of IBM MQ that supports this option must specify **PSCLUS(ENABLED)** for the publish/subscribe activity to function as expected. **ENABLED** is the default value when a queue manager is created.

### **DISABLED**

This queue manager cannot define clustered topic objects and ignores their definition on any other queue manager in the cluster.

Publications are not forwarded to subscribers elsewhere in the cluster, and subscriptions are not registered other than on the local queue manager.

To ensure that no publish/subscribe activity occurs in the cluster, all queue managers must specify **PSCLUS(DISABLED)**. As a minimum, full repositories must be consistent in enabling or disabling publish/subscribe participation.

## **PSMODE**

Controls whether the publish/subscribe engine and the queued publish/subscribe interface are running. It controls whether applications can publish or subscribe by using the application

programming interface. It also controls whether the queues that are monitored by the queued publish/subscribe interface, are monitored.

Changing the **PSMODE** attribute can change the **PSMODE** status. Use one of the following commands to determine the current state of the publish/subscribe engine and the queued publish/subscribe interface:

- **DISPLAY PUBSUB**
-  **DSPMQM** (on IBM i only)

### **COMPAT**

The publish/subscribe engine is running. It is therefore possible to publish or subscribe by using the application programming interface.

The queued publish/subscribe interface is not running. Any publish/subscribe messages put to the queues that are monitored by the queued publish/subscribe interfaces are not acted upon.

Use this setting for compatibility with IBM Integration Bus (formerly known as WebSphere Message Broker) V6 or earlier versions that use this queue manager.

### **DISABLED**

The publish/subscribe engine and the queued publish/subscribe interface are not running. It is therefore not possible to publish or subscribe by using the application programming interface. Any publish/subscribe messages put to the queues that are monitored by the queued publish/subscribe interfaces are not acted upon.

If a queue manager is in a publish/subscribe cluster or hierarchy, it might receive publish/subscribe messages from other queue managers in the cluster or hierarchy. Examples of such messages are publication messages or proxy subscriptions. While **PSMODE** is set to **DISABLED** those messages are not processed. For this reason, disable any queue manager in a publish/subscribe cluster or hierarchy only for as long as there is little build-up of messages.

### **ENABLED**

The publish/subscribe engine and the queued publish/subscribe interface are running. It is therefore possible to publish or subscribe by using the application programming interface and the queues that are being monitored by the queued publish/subscribe interface.

This is the queue manager's initial default value.

**Note:** If a queue manager is in a publish/subscribe cluster or hierarchy, and you change **PSMODE** to **ENABLED**, you might have to run the command **REFRESH QMGR TYPE (PROXY)**. The command ensures that non-durable subscriptions are known across the cluster or hierarchy when **PSMODE** is set back to **ENABLED**. The circumstance in which you must run the command is as follows. If **PSMODE** is changed from **ENABLED** to **DISABLED** and back to **ENABLED**, and one or more non-durable subscriptions exist across all three stages.

### **PSNPMSG**

If the queued publish/subscribe interface cannot process a non-persistent input message it might attempt to write the input message to the dead-letter queue. Whether it attempts to do so depends on the report options of the input message. The attempt to write the input message to the dead-letter queue might fail. In this case, the queued publish/subscribe interface might discard the input message. If **MQRO\_DISCARD\_MSG** is specified on the input message, the input message is discarded. If **MQRO\_DISCARD\_MSG** is not set, setting **PSNPMSG** to **KEEP** prevents the input message from being discarded. The default is to discard the input message.

**Note:** If you specify a value of **IFPER** for **PSSYNCP**, you must not specify a value of **KEEP** for **PSNPMSG**.

### **DISCARD**

Non-persistent input messages might be discarded if they cannot be processed.


## **KEEP**

Non-persistent input messages are not discarded if they cannot be processed. In this situation, the queued publish/subscribe interface continues to try to process this message again at appropriate intervals and does not continue processing subsequent messages.

## **PSNPRES**

The **PSNPRES** attribute controls whether the queued publish/subscribe interface writes an undeliverable reply message to the dead-letter queue, or discards the message. The choice is necessary if the queued publish/subscribe interface cannot deliver a reply message to the reply-to queue.

For new queue managers, the initial value is **NORMAL**. If you specify a value of **IFPER** for **PSSYNCPT**, you must not specify a value of **KEEP** or **SAFE** for **PSNPRES**.

 For migrated queue managers on [Multiplatforms](#), the value depends on `DLQNonPersistentResponse` and `DiscardNonPersistentResponse`.

## **NORMAL**

Non-persistent responses which cannot be placed on the reply queue are put on the dead-letter queue. If they cannot be placed on the dead-letter queue then they are discarded.

## **SAFE**

Non-persistent responses which cannot be placed on the reply queue are put on the dead-letter queue. If the response cannot be sent and cannot be placed on the dead-letter queue, the queued publish/subscribe interface backs out of the current operation. It tries again at appropriate intervals, and does not continue processing subsequent messages.

## **DISCARD**

Non-persistent responses which cannot be placed on the reply queue are discarded

## **KEEP**

Non-persistent responses are not placed on the dead-letter queue or discarded. Instead the queued publish/subscribe interface backs out the current operation and then tries it again at appropriate intervals and does not continue processing subsequent messages.

## **PSRTCNT**

If the queued publish/subscribe interface fails to process a command message under sync point, the unit of work is backed out. The command tries to process the message a number of times again, before the publish/subscribe broker processes the command message according to its report options instead. This situation can arise for a number of reasons. For example, if a publish message cannot be delivered to a subscriber, and it is not possible to put the publication on the dead letter queue.

The initial value for this parameter on a new queue manager is 5.

Range is 0 - 999,999,999.

## **PSSYNCPT**

Controls whether the queued publish/subscribe interface processes command messages (publishes or delete publication messages) under sync point.

### **YES**

All messages are processed under sync point.

### **IFPER**

Only persistent messages are part of the sync point

The initial value of the queue manager is **IFPER**.

## **RCVTIME (integer)**

The approximate length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state.

This parameter applies only to message channels and to MQI server-connection and client-connection channels where **SHARECNV** is greater than zero, when the channel receive timeout is set based on the

negotiated heartbeat interval in the same way as for message channels. This number can be qualified as follows:

- To specify that this number is a multiplier to apply to the negotiated **HBINT** value to determine how long a channel is to wait, set **RCVTTYE** to MULTIPLY. Specify a **RCVTIME** value of zero or in the range 2 through 99. If you specify zero, the channel continues to wait indefinitely to receive data from its partner.
- To specify that **RCVTIME** is the number of seconds to add to the negotiated **HBINT** value to determine how long a channel is to wait, set **RCVTTYE** to ADD. Specify an **RCVTIME** value in the range 1 through 999999.
- To specify that **RCVTIME** is a value, in seconds, that the channel is to wait, set **RCVTTYE** to EQUAL. Specify an **RCVTIME** value in the range 0 - 999,999. If you specify zero, the channel continues to wait indefinitely to receive data from its partner.

**Note:** For MQI channels that use sharing conversations, the heartbeat interval used by **ReceiveTimeout**, **ReceiveTimeMin**, or **ReceiveTimeoutType** is five seconds greater than the negotiated heartbeat interval.

For channels with **SHARECNV** equal to zero, **RCVTMIN** does not apply.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

For more information, see [Checking that the other end of the channel is still available](#).

This parameter is valid only on z/OS.

#### **RCVTMIN(integer)**

The minimum length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to an inactive state.

This parameter applies only to message channels and to MQI server-connection and client-connection channels where **SHARECNV** is greater than zero, when the channel receive timeout is set based on the negotiated heartbeat interval in the same way as for message channels.

**Note:** For MQI channels that use sharing conversations, the heartbeat interval used by **ReceiveTimeout**, **ReceiveTimeMin**, or **ReceiveTimeoutType** is five seconds greater than the negotiated heartbeat interval.

For channels with **SHARECNV** equal to zero, **RCVTMIN** does not apply.

The TCP/IP channel wait time can be configured relative to the negotiated value of **HBINT**. If **RCVTTYE** is MULTIPLY or ADD, the resultant value might be less than the value set in **RCVTMIN**. In this case, the TCP/IP channel wait time is set to **RCVTMIN**. If **RCVTTYE** is EQUAL then **RCVTMIN** does not apply.

Specify a value, in seconds, between zero and 999999.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

For more information, see [Checking that the other end of the channel is still available](#).

This parameter is valid only on z/OS.

#### **RCVTTYE**

The qualifier to apply to the value in **RCVTIME**.

##### **MULTIPLY**

Specifies that **RCVTIME** is a multiplier to be applied to the negotiated **HBINT** value to determine how long a channel waits.

##### **ADD**

Specifies that **RCVTIME** is a value, in seconds, to be added to the negotiated **HBINT** value to determine how long a channel waits.

## EQUAL

Specifies that **RCVTIME** is a value, in seconds, representing how long the channel waits.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

For more information, see [Checking that the other end of the channel is still available](#).

This parameter is valid only on z/OS.

## REMOTEEV

Specifies whether remote error events are generated, caused by an application or the queue manager not being able to access a remote queue on another queue manager, for example, the transmission queue not being correctly defined:


### DISABLED

Remote error events are not generated.

This is the queue manager's initial default value.

### ENABLED

Remote error events are generated.

 If you are using the reduced function form of IBM MQ for z/OS supplied with WebSphere Application Server, only DISABLED is valid.

## REPOS(*clustname*)

The name of a cluster for which this queue manager provides a repository manager service. The maximum length is 48 characters conforming to the rules for naming IBM MQ objects.

You can specify either the **REPOS** or the **REPOSNL** parameter, but not both. Both **REPOS** and **REPOSNL** might be blank, or **REPOS** might be blank and the namelist specified by **REPOSNL** might be empty. In these cases, this queue manager does not have a full repository. It might be a client of other repository services defined in the cluster.

Use a cluster-sender channel to connect this queue manager to at least one other full repository queue manager in the cluster (if specifying **REPOS**) or in each cluster named in the namelist (if specifying **REPOSNL**). See the information in [Components of a cluster](#) for details about using cluster-sender channels with full repository queue managers.

Successful completion of the command does not mean that the action completed. To check for true completion, see the **ALTER QMGR** step in [Checking that async commands for distributed networks have finished](#).

## REPOSNL(*nlname*)

The name of a namelist of clusters for which this queue manager provides a repository manager service. The maximum length is 48 characters conforming to the rules for naming an IBM MQ namelist object.

See the description of **REPOS** for information on specifying either **REPOS** or **REPOSNL**.

## REVDNS

Controls whether reverse lookup of the host name from a Domain Name Server (DNS) is done for the IP address from which a channel has connected. This attribute has an effect only on channels using a transport type (TRPTYPE) of TCP:

### ENABLED

DNS host names are reverse looked-up for the IP addresses of inbound channels when this information is required. This setting is required for matching against CHLAUTH rules that contain host names, and to include the host name in error messages. The IP address is still included in messages that provide a connection identifier.

This is the initial default value for the queue manager.

**DISABLED**

DNS host names are not reverse looked-up for the IP addresses of inbound channels. With this setting any CHLAUTH rules using host names are not matched.

**ROUTEREC**

Specifies whether trace-route information is recorded if requested in the message. If this parameter is not set to **DISABLED**, it controls whether any reply generated is sent to `SYSTEM.ADMIN.TRACE.ROUTE.QUEUE`, or to the destination specified by the message itself. If **ROUTEREC** is not **DISABLED**, messages not yet at the final destination might have information added to them.

**DISABLED**

Trace-route information is not recorded.

**MSG**

Trace-route information is recorded and sent to the destination specified by the originator of the message causing the trace route record.

This is the queue manager's initial default value.

**QUEUE**

Trace-route information is recorded and sent to `SYSTEM.ADMIN.TRACE.ROUTE.QUEUE`.

Multi **SCHINIT**

Specifies whether the channel initiator starts automatically when the queue manager starts.

**QMGR**

The channel initiator starts automatically when the queue manager starts.

**MANUAL**

The channel initiator does not start automatically.

This parameter is valid only on [Multiplatforms](#).

Multi **SCMDSERV**

Specifies whether the command server starts automatically when the queue manager starts.

**QMGR**

The command server starts automatically when the queue manager starts.

**MANUAL**

The command server does not start automatically.

This parameter is valid only on [Multiplatforms](#).

z/OS **SCYCASE**

Specifies whether the security profiles are uppercase or mixed case.

**UPPER**

The security profiles are uppercase only. However, `MXTOPIC` and `GMXTOPIC` are used for topic security, and can contain mixed-case profiles.

**MIXED**

The security profiles are mixed case. `MQCMD` and `MQCONN` are used for command and connection security but they can contain only uppercase profiles.

Changes to **SCYCASE** become effective after you run the following command:

```
REFRESH SECURITY(*) TYPE(CLASSES)
```

This parameter is valid only on [z/OS](#).

z/OS **SQQMNAME**

The **SQQMNAME** attribute specifies whether a queue manager in a queue sharing group opens a shared queue in the same group directly. The processing queue manager calls `MQOPEN` for a shared queue and sets the *ObjectQmgrName* parameter for the queue. If the shared queue is in the same queue

sharing group as the processing queue manager, the queue can be opened directly by the processing queue manager. Set the **SQQMNAME** attribute to control if the queue is opened directly, or by the *ObjectQmgrName* queue manager. The attribute will also be honored when opening a QALIAS with copy disposition, if the target queue is a shared queue in the same queue sharing group as the processing queue manager. In this situation it is important that the QALIAS copy object on each queue manager in the queue sharing group has the same target queue.

#### USE

The *ObjectQmgrName* is used, and the appropriate transmission queue is opened.

#### IGNORE

The processing queue manager opens the shared queue directly. Setting the parameter to this value can reduce the traffic in your queue manager network.

This parameter is valid only on z/OS.

### SSLCRLNL ( *nlname* )

The name of a namelist of authentication information objects which are used to provide certificate revocation locations to allow enhanced TLS certificate checking.






**Attention:** The namelist can reference a maximum of one OCSP type AUTHINFO object only.

From IBM MQ 9.2.0, if more than one OCSP type AUTHINFO objects is referenced in the NAMELIST, only the first entry is used.

If SSLCRLNL is blank, certificate revocation checking is not invoked unless one of the TLS certificates used contains an AuthorityInfoAccess or CrlDistributionPoint X.509 certificate extension.

Changes to SSLCRLNL, or to the names in a previously specified namelist, or to previously referenced authentication information objects become effective as follows:

- When a **REFRESH SECURITY TYPE(SSL)** command is issued.
-  On AIX, Linux, and Windows:
  - When a new channel process is started
  - For channels that run as threads of the channel initiator, when the channel initiator is restarted
  - For channels that run as threads of the listener, when the listener is restarted
-  On IBM i:
  - When a new channel process is started
  - For channels that run as threads of the channel initiator, when the channel initiator is restarted
  - For channels that run as threads of the listener, when the listener is restarted

On IBM i queue managers, this parameter is ignored. However, it is used to determine which authentication information objects are written to the AMQCLCHL . TAB file.
-  On z/OS, when the channel initiator is restarted.

Only authentication information objects with types of CRLLDAP or OCSP are allowed in the namelist referred to by **SSLCRLNL**. Any other type results in an error message when the list is processed and is subsequently ignored.

### SSLCRYP(*string*)

Sets the name of the parameter string required to configure the cryptographic hardware present on the system.

All supported cryptographic hardware supports the PKCS #11 interface. Specify a string of the following format:

```
GSK_PKCS11= the PKCS #11 driver path and file name>  
; the PKCS #11 token label> ;
```

```
the PKCS #11 token password> ; symmetric cipher setting>
;
```

The PKCS #11 driver path is an absolute path to the shared library providing support for the PKCS #11 card. The PKCS #11 driver file name is the name of the shared library. An example of the value required for the PKCS #11 driver path and file name is `/usr/lib/pkcs11/PKCS11_API.so`

To access symmetric cipher operations through GSKit, specify the symmetric cipher setting parameter. The value of this parameter is either:

#### **SYMMETRIC\_CIPHER\_OFF**

Do not access symmetric cipher operations.

#### **SYMMETRIC\_CIPHER\_ON**

Access symmetric cipher operations.

If the symmetric cipher setting parameter is not specified, it has the same effect as specifying `SYMMETRIC_CIPHER_OFF`.

The maximum length of the string is 256 characters.

If you specify a string that is not in the format listed, you get an error.

When the **SSLCRYP** value is changed, the cryptographic hardware parameters specified become the ones used for new TLS connection environments. The new information becomes effective:

- When a new channel process is started.
- For channels that run as threads of the channel initiator, when the channel initiator is restarted.
- For channels that run as threads of the listener, when the listener is restarted.
- When a **REFRESH SECURITY TYPE(SSL)** command is issued.

#### **SSLEV**

Specifies whether TLS events are generated.

##### **DISABLED**

TLS events are not generated.

This is the queue manager's initial default value.

##### **ENABLED**

All TLS events are generated.

#### **z/OS ALW SSLFIPS**

**SSLFIPS** specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM MQ, rather than in cryptographic hardware. If cryptographic hardware is configured, the cryptographic modules used are those modules provided by the hardware product. These might, or might not, be FIPS-certified to a particular level. Whether the modules are FIPS-certified depends on the hardware product in use. For more information about FIPS, see the [Federal Information Processing Standards \(FIPS\) manual](#).

##### **NO**

If you set **SSLFIPS** to NO, you can use either FIPS certified or non-FIPS certified CipherSpecs.

If the queue manager runs without using cryptographic hardware, refer to the CipherSpecs listed in [Specifying CipherSpecs](#).

This is the queue manager's initial default value.

##### **YES**

Specifies that only FIPS-certified algorithms are to be used in the CipherSpecs allowed on all TLS connections from and to this queue manager.

For a listing of appropriate FIPS 140-2 certified CipherSpecs; see [Specifying CipherSpecs](#).

Changes to **SSLFIPS** become effective as follows:

-  On AIX, Linux, and Windows:



- when a **REFRESH SECURITY TYPE(SSL)** command is issued
- when a new channel process is started
- for channels that run as threads of the channel initiator, when the channel initiator is restarted
- for channels that run as threads of the listener, when the listener is restarted
- for channels that run as threads of a process pooling process, when the process pooling process is started or restarted and first runs a TLS channel. If the process pooling process has already run a TLS channel, and you want the change to become effective immediately, run the MQSC command **REFRESH SECURITY TYPE(SSL)**. The process pooling process is **amqzmpa**

- **z/OS** On z/OS, when the channel initiator is restarted.

This parameter is valid on z/OS, AIX, Linux, and Windows.

### SSLKEYR(string)

The name of the Secure Sockets Layer key repository. The maximum length of the string is 256 characters. The format of the name depends on the environment.

**z/OS** On z/OS, the name is the name of a key ring.

**Multi** **V 9.3.0** **V 9.3.0** On Multiplatforms, the name is the full path and the file name of the key repository:

- **IBM i** **V 9.3.0** **V 9.3.0** On IBM i, the name is of the form *keyfile.kdb*, where *keyfile* identifies a GSKit CMS key database file. If the file suffix is not specified, it is assumed to be *.kdb*.
  - If you specify **\*SYSTEM**, IBM MQ uses the system certificate store as the key repository for the queue manager. The queue manager is registered as a server application in the Digital Certificate Manager (DCM). You can assign any server/client certificate in the system store to the queue manager, because you registered it as a server application.
  - If you change the SSLKEYR parameter to a value other than **\*SYSTEM**, IBM MQ unregisters the queue manager as an application with DCM.
- **Linux** **V 9.3.0** **AIX** **V 9.3.0** On AIX and Linux, the name is of the form *pathname/keyfile* where *keyfile* identifies a GSKit CMS or PKCS#12 key database file.
- **Windows** **V 9.3.0** **V 9.3.0** On Windows, the name is of the form *pathname\keyfile* where *keyfile* identifies a GSKit CMS or PKCS#12 key database file.

**V 9.3.0** On Windows and Linux if TLS AMQP channels are used, the suffix of the key repository file must be one of the following:

- *.kdb*, for a CMS key repository
- *.p12* or *.pkcs12*, for a PKCS #12 key repository.

**Multi** On Multiplatforms, the syntax of this parameter is validated to ensure that it contains a valid and absolute directory path.

If **SSLKEYR** is blank, channels using TLS do not start. If **SSLKEYR** is set to a value that does not correspond to a key ring or key database file, channels using TLS also do not start.

Changes to **SSLKEYR** become effective as follows:

- When a **REFRESH SECURITY TYPE(SSL)** command is issued.
- **Multi** On Multiplatforms:
  - When a new channel process is started.
  - For channels that run as threads of the channel initiator, when the channel initiator is restarted.
  - For channels that run as threads of the listener, when the listener is restarted.

- For channels that run as threads of a process pooling process, **amqrmppa**, when the process pooling process is started or restarted and first runs a TLS channel. If the process pooling process has already run a TLS channel, and you want the change to become effective immediately, run the MQSC command **REFRESH SECURITY TYPE(SSL)**.

- **z/OS** On z/OS, when the channel initiator is restarted.

### **SSLRKEYC(integer)**

The number of bytes to be sent and received within a TLS conversation before the secret key is renegotiated. The number of bytes includes control information.

SSLRKEYC is used only by TLS channels which initiate communication from the queue manager. For example, the sender channel initiates communication in a sender and receiver channel pairing.

If a value greater than zero is specified, the secret key is also renegotiated before message data is sent or received following a channel heartbeat. The count of bytes until the next secret key renegotiation is reset after each successful renegotiation.

Specify a value in the range 0 - 999,999,999. A value of zero means that the secret key is never renegotiated. If you specify a TLS secret key reset count in the range 1 - 32767 bytes (32 KB), TLS channels use a secret key reset count of 32 KB. The larger reset count value avoids the cost of excessive key resets which would occur for small TLS secret key reset values.



**Attention:** If your enterprise has applied APAR *PH30305*, the following statement no longer applies:

- Non-zero values less than 4096 (4 KB) might cause channels to fail to start, or might cause inconsistencies in the values of **SSLKEYDA**, **SSLKEYTI**, and **SSLRKEYS**.

**z/OS**

### **SSLTASKS(integer)**

The number of server subtasks to use for processing TLS calls. To use TLS channels, you must have at least two of these tasks running.

This value is in the range 0 - 9999. To avoid problems with storage allocation, do not set the **SSLTASKS** parameter to a value greater than 50.

Changes to this parameter are effective when the channel initiator is restarted.

This parameter is valid only on z/OS.

### **STATACLS**

Specifies whether statistics data is to be collected for auto-defined cluster-sender channels:

#### **QMGR**

Collection of statistics data is inherited from the setting of the **STATCHL** parameter of the queue manager.

This is the queue manager's initial default value.

#### **OFF**

Statistics data collection for the channel is disabled.

#### **LOW**

Unless **STATCHL** is **NONE**, statistics data collection is switched on with a low ratio of data collection with a minimal effect on system performance.

#### **MEDIUM**

Unless **STATCHL** is **NONE**, statistics data collection is switched on with a moderate ratio of data collection.

#### **HIGH**

Unless **STATCHL** is **NONE**, statistics data collection is switched on with a high ratio of data collection.

A change to this parameter takes effect only on channels started after the change occurs. Any channel started before the change to the parameter continues with the value in force at the time that the channel started.

**z/OS** On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

## **STATCHL**

Specifies whether statistics data is to be collected for channels:

### **NONE**

Statistics data collection is turned off for channels regardless of the setting of their **STATCHL** parameter.

### **OFF**

Statistics data collection is turned off for channels specifying a value of QMGR in their **STATCHL** parameter.

This is the queue manager's initial default value.

### **LOW**

Statistics data collection is turned on, with a low ratio of data collection, for channels specifying a value of QMGR in their **STATCHL** parameter.

### **MEDIUM**

Statistics data collection is turned on, with a moderate ratio of data collection, for channels specifying a value of QMGR in their **STATCHL** parameter.

### **HIGH**

Statistics data collection is turned on, with a high ratio of data collection, for channels specifying a value of QMGR in their **STATCHL** parameter.

A change to this parameter takes effect only on channels started after the change occurs. Any channel started before the change to the parameter continues with the value in force at the time that the channel started.

**z/OS** On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

## **Multi STATINT(integer)**

The time interval, in seconds, at which statistics monitoring data is written to the monitoring queue.

Specify a value in the range 1 through 604800.

Changes to this parameter take immediate effect on the collection of monitoring and statistics data.

This parameter is valid only on [Multiplatforms](#).

## **Multi STATMQI**

Specifies whether statistics monitoring data is to be collected for the queue manager:

### **OFF**

Data collection for MQI statistics is disabled.

This is the queue manager's initial default value.

### **ON**

Data collection for MQI statistics is enabled.

Changes to this parameter take immediate effect on the collection of monitoring and statistics data.

This parameter is valid only on [Multiplatforms](#).

## **V9.3.0 V9.3.0 STATQ**

Specifies whether statistics data is to be collected for queues:

**Multi** **NONE**

Statistics data collection is turned off for queues regardless of the setting of their **STATQ** parameter.

**OFF**

Statistics data collection is turned off for queues specifying a value of QMGR or OFF in their **STATQ** parameter. OFF is the default value.

**ON**

Statistics data collection is turned on for queues specifying a value of QMGR or ON in their **STATQ** parameter.

**z/OS** On z/OS, you must switch on class 5 statistics by the START TRACE command.

Statistics messages are generated only for queues which are opened after statistics collection is enabled. You do not need to restart the queue manager for the new value of STATQ to take effect.

**STRSTPEV**

Specifies whether start and stop events are generated:

**ENABLED**

Start and stop events are generated.

This is the queue manager's initial default value.

**DISABLED**

Start and stop events are not generated.

**SUITEB**

Specifies whether Suite B-compliant cryptography is used and what strength is required.

**NONE**

Suite B is not used. NONE is the default

**128\_BIT**

Suite B 128-bit level security is used.

**192\_BIT**

Suite B 192-bit level security is used

**128\_BIT,192\_BIT**

Both Suite B 128-bit and 192-bit level security is used

**z/OS** **TCPCHL(integer)**

The maximum number of channels that can be current, or clients that can be connected, that use the TCP/IP transmission protocol.

The maximum number of sockets used is the sum of the values in **TCPCHL** and **CHDISPS**. The z/OS UNIX System Services **MAXFILEPROC** parameter (specified in the BPXPRM<sub>xx</sub> member of SYS1.PARMLIB) controls how many sockets each task is allowed, and thus how many channels each dispatcher is allowed. In this case, the number of channels using TCP/IP is limited to the value of **MAXFILEPROC** multiplied by the value of **CHDISPS**.

Specify a value 0-9999. The value must not be greater than the value of **MAXCHL**. **MAXCHL** defines the maximum number of channels available. TCP/IP might not support as many as 9999 channels. If so, the value you can specify is limited by the number of channels TCP/IP can support. If you specify zero, the TCP/IP transmission protocol is not used.

If you change this value, also review the **MAXCHL**, **LU62CHL**, and **ACTCHL** values to ensure that there is no conflict of values. If necessary, raise the value of **MAXCHL** and **ACTCHL**.

If the value of this parameter is reduced, any current channels that exceed the new limit continue to run until they stop.

Sharing conversations do not contribute to the total for this parameter.

If the value of **TCPCHL** is non-zero when the channel initiator starts up, the value can be modified dynamically. If the value of **TCPCHL** is zero when the channel initiator starts up, a later **ALTER**

command does not take effect. In this case, you should carry out an **ALTER** command, either before the channel initiator starts, or in CSQINP2 before you issue the **START CHINIT** command.

This parameter is valid only on z/OS.

#### **z/OS TCPKEEP**

Specifies whether the **KEEPALIVE** facility is to be used to check that the other end of the connection is still available. If it is unavailable, the channel is closed.

##### **NO**

The TCP **KEEPALIVE** facility is not to be used.

This is the queue manager's initial default value.

##### **YES**

The TCP **KEEPALIVE** facility is to be used as specified in the TCP profile configuration data set. The interval is specified in the **KAINT** channel attribute.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

This parameter is valid only on z/OS.

Using the **TCPKEEP** parameter is no longer required for 'modern' queue managers. The replacement is a combination of:

- using 'modern' client channels (**SHARECNV** <> 0)
- using receive timeout for message channels **RCVTIME**.

For more information, see the technote *Setting the TCP/IP KeepAlive interval to be used by IBM MQ*, at: <https://www.ibm.com/support/pages/node/342737>

#### **z/OS TCPNAME(string)**

The name of either the only, or preferred, TCP/IP stack to be used, depending on the value of **TCPSTACK**. This name is the name of the z/OS UNIX System Services stack for TCP/IP, as specified in the **SUBFILESYSTYPE** NAME parameter in the BPXPRMxxx member of SYS1.PARMLIB. **TCPNAME** is only applicable in CINET multiple stack environments. The queue manager's initial default value is TCPIP.

In INET single stack environments the channel initiator uses the only available TCP/IP stack.

The maximum length of this parameter is eight characters.

Changes to this parameter take effect when the channel initiator is restarted.

This parameter is valid only on z/OS.

#### **z/OS TCPSTACK**

Specifies whether the channel initiator can use only the TCP/IP stack specified in **TCPNAME**, or optionally bind to any selected TCP/IP stack defined. This parameter is only applicable in CINET multiple stack environments.

##### **SINGLE**

The channel initiator can use only the TCP/IP address space specified in **TCPNAME**.

##### **MULTIPLE**

The channel initiator can use any TCP/IP address space available to it.

Changes to this parameter take effect when the channel initiator is restarted.

This parameter is valid only on z/OS.

#### **z/OS TRAXSTR**

Specifies whether the channel initiator trace starts automatically:

##### **YES**

Channel initiator trace is to start automatically.

## NO

Channel initiator trace is not to start automatically.

Changes to this parameter take effect when the channel initiator is restarted. If you want to start or stop channel initiator trace without restarting the channel initiator, use the **START TRACE** or **STOP TRACE** commands after starting the channel initiator.

This parameter is valid only on z/OS.

## **TRAXTBL(integer)**

The size, in megabytes, of the trace data space of the channel initiator.

Specify a value in the range 2 through 2048.

### Note:

1. Changes to this parameter take effect immediately; any existing trace table contents are lost.
2. The **CHINIT** trace is stored in a dataspace called qmidCHIN.CSQXTRDS. When you use large z/OS data spaces, ensure that sufficient auxiliary storage is available on your system to support any related z/OS paging activity. You might also need to increase the size of your SYS1.DUMP data sets.

This parameter is valid only on z/OS.

## **TREELIFE (integer)**

The lifetime, in seconds of non-administrative topics.

Non-administrative topics are those topics created when an application publishes to, or subscribes on, a topic string that does not exist as an administrative node. When this non-administrative node no longer has any active subscriptions, this parameter determines how long the queue manager waits before removing that node. Only non-administrative topics that are in use by a durable subscription remain after the queue manager is recycled.

Specify a value in the range 0 through 604000. A value of 0 means that non-administrative topics are not removed by the queue manager.

## **TRIGINT(integer)**

A time interval expressed in milliseconds.

The **TRIGINT** parameter is relevant only if the trigger type (**TRIGTYPE**) is set to **FIRST** (see [“DEFINE QLOCAL \(define a new local queue\)”](#) on page 574 for details). In this case trigger messages are normally generated only when a suitable message arrives on the queue, and the queue was previously empty. Under certain circumstances, however, an additional trigger message can be generated with **FIRST** triggering even if the queue was not empty. These additional trigger messages are not generated more often than every **TRIGINT** milliseconds; see [Special case of trigger type FIRST](#).

Specify a value in the range 0 - 999,999,999.

### Related concepts

[Working with dead-letter queues](#)

 [Working with TLS on z/OS](#)

### Related tasks

[Displaying and altering queue manager attributes](#)

## **ALTER queues (alter queue settings)**

Use the MQSC **ALTER** command to alter the parameters of a queue. A queue might be a local queue (**ALTER QLOCAL**), alias queue (**ALTER QALIAS**), model queue (**ALTER QMODEL**), a remote queue, a queue manager alias, or a reply-to queue alias (**ALTER QREMOTE**).


## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

This section contains the following commands:

- [“ALTER QALIAS” on page 406](#)
- [“ALTER QLOCAL” on page 408](#)
- [“ALTER QMODEL” on page 410](#)
- [“ALTER QREMOTE” on page 412](#)

Parameters not specified in the **ALTER** queue commands result in the existing values for those parameters being left unchanged.

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

## Usage notes for ALTER queues

- Successful completion of the command does not mean that the action completed. To check for true completion, see the [ALTER queues](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for ALTER QUEUE

The parameters that are relevant for each type of queue are tabulated in [Table 139 on page 383](#). Each parameter is described after the table.






Parameter	Local queue	Model queue	Alias queue	Remote queue
<a href="#">ACCTQ</a>	✓	✓		
<a href="#">BOQNAME</a>	✓	✓		
<a href="#">BOTHRESH</a>	✓	✓		
  <a href="#">CAPEXPY</a>	✓	✓	✓	✓
 <a href="#">CFSTRUCT</a>	✓	✓		
<a href="#">CLCHNAME</a>	✓			
<a href="#">CLUSNL</a>	✓		✓	✓
<a href="#">CLUSTER</a>	✓		✓	✓
<a href="#">CLWLPRTY</a>	✓		✓	✓
<a href="#">CLWLANK</a>	✓		✓	✓
<a href="#">CLWLUSEQ</a>	✓			
 <a href="#">CMDSCOPE</a>	✓	✓	✓	✓
<a href="#">CUSTOM</a>	✓	✓	✓	✓

Table 139. DEFINE and ALTER QUEUE parameters (continued)

Parameter	Local queue	Model queue	Alias queue	Remote queue
<u>DEFBIND</u>	✓		✓	✓
<u>DEFPRESP</u>	✓	✓	✓	✓
<u>DEFPRTY</u>	✓	✓	✓	✓
<u>DEFPSIST</u>	✓	✓	✓	✓
<u>DEFREADA</u>	✓	✓	✓	
<u>DEFSOPT</u>	✓	✓		
<u>DEFTYPE</u>	✓	✓		
<u>DESCR</u>	✓	✓	✓	✓
<u>DISTL</u>	✓	✓		
<u>FORCE</u>	✓		✓	✓
<u>GET</u>	✓	✓	✓	
<u>HARDENBO</u> or <u>NOHARDENBO</u>	✓	✓		
<u>IMGRCOVQ</u>	✓	✓		
<u>INDXTYPE</u>	✓	✓		
<u>INITQ</u>	✓	✓		
<u>LIKE</u>	✓	✓	✓	✓
<u>MAXDEPTH</u>	✓	✓		
<u>MAXFSIZE</u>	✓	✓		
<u>MAXMSGL</u>	✓	✓		
<u>MONQ</u>	✓	✓		
<u>MSGDLVSQ</u>	✓	✓		
<u>NPMCLASS</u>	✓	✓		
<u>PROCESS</u>	✓	✓		
<u>PROPCTL</u>	✓	✓	✓	
<u>PUT</u>	✓	✓	✓	✓
<i>queue-name</i>	✓	✓	✓	✓
<u>QDEPTHHI</u>	✓	✓		
<u>QDEPTHLO</u>	✓	✓		



Table 139. DEFINE and ALTER QUEUE parameters (continued)

Parameter	Local queue	Model queue	Alias queue	Remote queue
<u>QDPHIEV</u>	✓	✓		
<u>QDPLOEV</u>	✓	✓		
<u>QDPMAXEV</u>	✓	✓		
 <u>QSGDISP</u>	✓	✓	✓	✓
<u>QSVCI EV</u>	✓	✓		
<u>QSVCI NT</u>	✓	✓		
<u>RETINTVL</u>	✓	✓		
<u>RNAME</u>				✓
<u>RQMNAME</u>				✓
<u>SCOPE</u>	✓		✓	✓
<u>SHARE</u> or <u>NOSHARE</u>	✓	✓		
<u>STATQ</u>	✓	✓		
 <u>STGCLASS</u>	✓	✓		
 <u>STREAMQ</u>	✓	✓		
 <u>STRMQOS</u>	✓	✓		
<u>TARGET</u>			✓	
<u>TARGQ</u>			✓	
<u>TARGETTYPE</u>			✓	
<u>TRIGDATA</u>	✓	✓		
<u>TRIGDPTH</u>	✓	✓		
<u>TRIGGER</u> or <u>NOTRIGGER</u>	✓	✓		
<u>TRIGMPRI</u>	✓	✓		
<u>TRIGTYPE</u>	✓	✓		
<u>USAGE</u>	✓	✓		
<u>XMITQ</u>				✓

### **queue-name**

Local name of the queue, except the remote queue where it is the local definition of the remote queue.

See [Rules for naming IBM MQ objects](#).

### **ACCTQ**


Specifies whether accounting data collection is to be enabled for the queue. On z/OS, the data collected is class 3 accounting data (thread-level and queue-level accounting). In order for accounting data to be collected for this queue, accounting data for this connection must also be enabled. Turn on accounting data collection by setting either the **ACCTQ** queue manager attribute, or the options field in the MQCNO structure on the MQCONN call.

### **QMGR**

The collection of accounting data is based on the setting of the **ACCTQ** parameter on the queue manager definition.

### **ON**

Accounting data collection is enabled for the queue unless the **ACCTQ** queue manager parameter has a value of NONE.

 On z/OS systems, you must enable class 3 accounting using the **START TRACE** command.

### **OFF**

Accounting data collection is disabled for the queue.

### **BOQNAME(queue-name)**

The excessive backout requeue name.

This parameter is supported only on local and model queues.

Use this parameter to set or change the back out queue name attribute of a local or model queue. Apart from allowing its value to be queried, the queue manager does nothing based on the value of this attribute. IBM MQ classes for JMS transfers a message that is backed out the maximum number of times to this queue. The maximum is specified by the **BOTHRESH** attribute.

### **BOTHRESH(integer)**

The backout threshold.

This parameter is supported only on local and model queues.

Use this parameter to set or change the value of the back out threshold attribute of a local or model queue. Apart from allowing its value to be queried, the queue manager does nothing based on the value of this attribute. IBM MQ classes for JMS use the attribute to determine how many times to allow a message to be backed out. When the value is exceeded, the message is transferred to the queue named by the **BOQNAME** attribute.

Specify a value in the range 0 - 999,999,999.

### **CAEXPRY(integer)**

The maximum time, expressed in tenths of a second, until a message put using an object handle with this object in the resolution path, becomes eligible for expiry processing.

**Important:**  You cannot use the **CAEXPRY** attribute introduced in IBM MQ 9.3.1 with a cluster if the full repository is on z/OS.

For more information on message expiry processing, and what happens if you migrate from an earlier version of the product, see [Enforcing lower expiration times](#).

### **integer**

The value must be in the range one through to 999 999 999.

### **NOLIMIT**

There is no limit on the expiry time of messages put using this object. This is the default value.

If you set **CAEXPRY** to NOLIMIT, you can set the **CUSTOM CAEXPRY** attribute as well.

Note that existing messages in the queue, prior to a change in **CAPEXPY**, are not affected by the change (that is, their expiry time remains intact). Only new messages that are put into the queue after the change in **CAPEXPY** have the new expiry time.

z/OS

### **CFSTRUCT**(*structure-name*)

Specifies the name of the coupling facility structure where you want messages stored when you use shared queues.

This parameter is supported only on z/OS for local and model queues.

The name:

- Cannot have more than 12 characters
- Must start with an uppercase letter (A - Z)
- Can include only the characters A - Z and 0 - 9

The name of the queue sharing group to which the queue manager is connected is prefixed to the name you supply. The name of the queue sharing group is always four characters, padded with @ symbols if necessary. For example, if you use a queue sharing group named NY03 and you supply the name PRODUCT7, the resultant coupling facility structure name is NY03PRODUCT7. The administrative structure for the queue sharing group (in this case NY03CSQ\_ADMIN) cannot be used for storing messages.

For **ALTER QLOCAL**, **ALTER QMODEL**, **DEFINE QLOCAL** with **REPLACE**, and **DEFINE QMODEL** with **REPLACE** the following rules apply:

- On a local queue with **QSGDISP**(SHARED), **CFSTRUCT** cannot change.
- If you change either the **CFSTRUCT** or **QSGDISP** value you must delete and redefine the queue. To preserve any of the messages on the queue you must offload the messages before you delete the queue. Reload the messages after you redefine the queue, or move the messages to another queue.
- On a model queue with **DEFTYPE**(SHAREDYN), **CFSTRUCT** cannot be blank.
- On a local queue with a **QSGDISP** other than SHARED, or a model queue with a **DEFTYPE** other than SHAREDYN, the value of **CFSTRUCT** does not matter.

For **DEFINE QLOCAL** with **NOREPLACE** and **DEFINE QMODEL** with **NOREPLACE**, the coupling facility structure:

- On a local queue with **QSGDISP**(SHARED) or a model queue with a **DEFTYPE**(SHAREDYN), **CFSTRUCT** cannot be blank.
- On a local queue with a **QSGDISP** other than SHARED, or a model queue with a **DEFTYPE** other than SHAREDYN, the value of **CFSTRUCT** does not matter.

**Note:** Before you can use the queue, the structure must be defined in the coupling facility Resource Management (CFRM) policy data set.

### **CLCHNAME**(*channel name*)

This parameter is supported only on transmission queues.

**CLCHNAME** is the generic name of the cluster-sender channels that use this queue as a transmission queue. The attribute specifies which cluster-sender channels send messages to a cluster-receiver channel from this cluster transmission queue.

You can also set the transmission queue attribute **CLCHNAME** attribute to a cluster-sender channel manually. Messages that are destined for the queue manager connected by the cluster-sender channel are stored in the transmission queue that identifies the cluster-sender channel. They are not stored in the default cluster transmission queue. If you set the **CLCHNAME** attribute to blanks, the channel switches to the default cluster transmission queue when the channel restarts. The default queue is either **SYSTEM.CLUSTER.TRANSMIT.ChannelName** or **SYSTEM.CLUSTER.TRANSMIT.QUEUE**, depending on the value of the queue manager **DEFCLXQ** attribute.

By specifying asterisks, "" \* "" , in **CLCHNAME**, you can associate a transmission queue with a set of cluster-sender channels. The asterisks can be at the beginning, end, or any number of places in the middle of the channel name string. **CLCHNAME** is limited to a length of 48 characters, MQ\_OBJECT\_NAME\_LENGTH. A channel name is limited to 20 characters: MQ\_CHANNEL\_NAME\_LENGTH. If you specify an asterisk you must also set the SHARE attribute so that multiple channels can concurrently access the transmission queue.

**z/OS** If you specify a "\*" in **CLCHNAME**, to obtain a channel profile name, you must specify the channel profile name within quotation marks. If you do not specify the generic channel name within quotation marks you receive message CSQ9030E.

The default queue manager configuration is for all cluster-sender channels to send messages from a single transmission queue, SYSTEM . CLUSTER . TRANSMIT . QUEUE. The default configuration can be modified by changing the queue manager attribute, **DEFCLXQ**. The default value of the attribute is SCTQ. You can change the value to CHANNEL. If you set the **DEFCLXQ** attribute to CHANNEL, each cluster-sender channel defaults to using a specific cluster transmission queue, SYSTEM . CLUSTER . TRANSMIT . *ChannelName*.

**z/OS** On z/OS, if this parameter is set, the queue:

- Must be shareable, by specifying the queue attribute SHARE.
- Must be indexed on the correlation ID by specifying INDXTYPE(CORRELID).
- Must not be a dynamic or a shared queue.

#### **z/OS** **ALW** **CLUSNL(namelist name)**

The name of the namelist that specifies a list of clusters to which the queue belongs.

This parameter is supported only on alias, local, and remote queues.

Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of **CLUSNL** or **CLUSTER** can be non-blank; you cannot specify a value for both.

On local queues, this parameter cannot be set for the following queues:

- Transmission queues
- SYSTEM . CHANNEL . *xx* queues
- SYSTEM . CLUSTER . *xx* queues
- SYSTEM . COMMAND . *xx* queues
- **z/OS** On z/OS only, SYSTEM . QSG . *xx* queues

This parameter is valid only on the following platforms:

- AIX, Linux, and Windows
- z/OS

#### **z/OS** **ALW** **CLUSTER(cluster name)**

The name of the cluster to which the queue belongs.


This parameter is supported only on alias, local, and remote queues.

The maximum length is 48 characters conforming to the rules for naming IBM MQ objects. Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of **CLUSNL** or **CLUSTER** can be non-blank; you cannot specify a value for both.

On local queues, this parameter cannot be set for the following queues:

- Transmission queues
- SYSTEM . CHANNEL . *xx* queues
- SYSTEM . CLUSTER . *xx* queues

- SYSTEM.COMMAND.*xx* queues
-  On z/OS only, SYSTEM.QSG.*xx* queues

This parameter is valid only on the following platforms:

- AIX, Linux, and Windows
- z/OS

### **CLWLPRTY**(*integer*)

Specifies the priority of the queue for the purposes of cluster workload distribution. This parameter is valid only for local, remote, and alias queues. The value must be in the range zero through 9 where zero is the lowest priority and 9 is the highest. For more information about this attribute, see [CLWLPRTY queue attribute](#).

### **CLWLRANK** (*integer*)

Specifies the rank of the queue for the purposes of cluster workload distribution. This parameter is valid only for local, remote, and alias queues. The value must be in the range zero through 9 where zero is the lowest rank and 9 is the highest. For more information about this attribute, see [CLWLRANK queue attribute](#).

### **CLWLUSEQ**

Specifies the behavior of an MQPUT operation when the target queue has a local instance and at least one remote cluster instance. The parameter has no effect when the MQPUT originates from a cluster channel. This parameter is valid only for local queues.

#### **QMGR**

The behavior is as specified by the **CLWLUSEQ** parameter of the queue manager definition.

#### **ANY**

The queue manager is to treat the local queue as another instance of the cluster queue for the purposes of workload distribution.

#### **LOCAL**

The local queue is the only target of the MQPUT operation.

### **CMDSCOPE**

This parameter applies to z/OS only. It specifies where the command is run when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP or SHARED.

..

The command runs on the queue manager on which it was entered.

#### **QmgrName**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered. You can specify another name, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

### **CUSTOM**(*string*)

The custom attribute for new features.

This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE).

The maximum length is defined by the IBM MQ constant MQ\_CUSTOM\_LENGTH and is currently set to 128 on all platforms.

The **CUSTOM** attribute is intended to be used with the following IBM MQ attribute.

**Note:** **V9.3.1** The **CAEXPRY** queue attribute introduced at IBM MQ 9.3.1 for IBM MQ for Multiplatforms replaces the use of the **CAEXPRY** option in the **CUSTOM** field.

It is not possible to set the **CAEXPRY** attribute if the **CUSTOM** field has a **CAEXPRY** attribute defined in it already.

You should alter existing queues to set the new **CAEXPRY** field and unset the **CAEXPRY** attribute from the **CUSTOM** field. For example:

```
ALTER QL(Q1) CAEXPRY(1000) CUSTOM(' ')
```

On IBM MQ for z/OS you set **CAEXPRY** using the **CUSTOM** attribute.

For example:

```
ALTER QL(Q1) CUSTOM(' CAEXPRY(1000) ')
```

To unset the **CAEXPRY** attribute on IBM MQ for z/OS enter, for example, the following command:

```
ALTER QL(Q1) CUSTOM(' ')
```

Note, that on z/OS you must leave a space between the single quotation marks, otherwise the command fails.

The maximum time, expressed in tenths of a second, until a message put using an object handle with this object in the resolution path, becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

### **integer**

The value must be in the range one through to 999 999 999.

### **NOLIMIT**

There is no limit on the expiry time of messages put using this object. This is the default value.

Specifying a value for **CAEXPRY** that is not valid, does not cause the command to fail. Instead the default value is used.

Note that existing messages in the queue, prior to a change in **CAEXPRY**, are not affected by the change (that is, their expiry time remains intact). Only new messages that are put into the queue after the change in **CAEXPRY** have the new expiry time.

### **DEFBIND**

Specifies the binding to be used when the application specifies MQ00\_BIND\_AS\_Q\_DEF on the MQOPEN call, and the queue is a cluster queue.

### **OPEN**

The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

### **NOTFIXED**

The queue handle is not bound to any instance of the cluster queue. The queue manager selects a specific queue instance when the message is put using MQPUT. It changes that selection later, if the need arises.

### **GROUP**

Allows an application to request that a group of messages is allocated to the same destination instance.

Multiple queues with the same name can be advertised in a queue manager cluster. An application can send all messages to a single instance, MQ00\_BIND\_ON\_OPEN. It can allow a workload management algorithm to select the most suitable destination on a per message basis, MQ00\_BIND\_NOT\_FIXED. It can allow an application to request that a group of messages be all allocated to the same destination instance. The workload balancing reselects a destination between groups of messages, without requiring an MQCLOSE and MQOPEN of the queue.

The MQPUT1 call always behaves as if NOTFIXED is specified.

This parameter is valid on all platforms.

#### **DEFPRESP**

Specifies the behavior to be used by applications when the put response type, within the MQPMO options, is set to MQPMO\_RESPONSE\_AS\_Q\_DEF.

##### **SYNC**

Put operations to the queue specifying MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE is specified instead.

##### **ASYN**

Put operations to the queue specifying MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_ASYNC\_RESPONSE is specified instead; see [MQPMO options \(MQLONG\)](#).

#### **DEFPRTY(*integer*)**

The default priority of messages put on the queue. The value must be in the range 0 - 9. Zero is the lowest priority, through to the **MAXPRTY** queue manager parameter. The default value of **MAXPRTY** is 9.

#### **DEFPSIST**


Specifies the message persistence to be used when applications specify the MQPER\_PERSISTENCE\_AS\_Q\_DEF option.

##### **NO**

Messages on this queue are lost across a restart of the queue manager.

##### **YES**

Messages on this queue survive a restart of the queue manager.

 On z/OS, N and Y are accepted as synonyms of NO and YES.

#### **DEFREADA**

Specifies the default read ahead behavior for non-persistent messages delivered to the client. Enabling read ahead can improve the performance of client applications consuming non-persistent messages.

##### **NO**

Non-persistent messages are not read ahead unless the client application is configured to request read ahead.

##### **YES**

Non-persistent messages are sent to the client before an application requests them. Non-persistent messages can be lost if the client ends abnormally or if the client does not delete all the messages it is sent.

##### **DISABLED**


Read ahead of non-persistent messages is not enabled for this queue. Messages are not sent ahead to the client regardless of whether read ahead is requested by the client application.

#### **DEFSOPT**

The default share option for applications opening this queue for input:

##### **EXCL**

The open request is for exclusive input from the queue.

 On z/OS, EXCL is the default value.

##### **SHARED**

The open request is for shared input from the queue.

 On Multiplatforms, SHARED is the default value.

#### **DEFTYPE**

Queue definition type.

This parameter is supported only on model queues.

## PERMDYN

A permanent dynamic queue is created when an application issues an MQOPEN MQI call with the name of this model queue specified in the object descriptor (MQOD).

▶ **z/OS** On z/OS, the dynamic queue has a disposition of QMGR.

## ▶ **z/OS** SHAREDYN

This option is available on z/OS only.

A permanent dynamic queue is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor (MQOD).

The dynamic queue has a disposition of SHARED.

## TEMPDYN

A temporary dynamic queue is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor (MQOD).

▶ **z/OS** On z/OS, the dynamic queue has a disposition of QMGR.

▶ **z/OS** Do not specify this value for a model queue definition with a **DEFPSIST** parameter of YES.

▶ **z/OS** If you specify this option, do not specify **INDXTYPE**(MSGTOKEN).

## DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the **DISPLAY QUEUE** command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** Use characters that are in the coded character set identifier (CCSID) of this queue manager. If you do not do so and if the information is sent to another queue manager, they might be translated incorrectly.

## ▶ **ALW** DISTL

Sets whether distribution lists are supported by the partner queue manager.

### YES

Distribution lists are supported by the partner queue manager.

### NO

Distribution lists are not supported by the partner queue manager.

**Note:** You do not normally change this parameter, because it is set by the MCA. However you can set this parameter when defining a transmission queue if the distribution list capability of the destination queue manager is known.

This parameter is valid only on AIX, Linux, and Windows.

## FORCE

This parameter applies only to the **ALTER** command on alias, local and remote queues.

Specify this parameter to force completion of the command in the following circumstances.

For an alias queue, if both of the following statements are true:

- The **TARGET** parameter specifies a queue
- An application has this alias queue open

For a local queue, if both of the following statements are true:

- The **NOSHARE** parameter is specified
- More than one application has the queue open for input

**FORCE** is also needed if both of the following statements are true:



- The **USAGE** parameter is changed
- Either one or more messages are on the queue, or one or more applications have the queue open

Do not change the **USAGE** parameter while there are messages on the queue; the format of messages changes when they are put on a transmission queue.

For a remote queue, if both of the following statements are true:

- The **XMITQ** parameter is changed
- One or more applications has this queue open as a remote queue

**FORCE** is also needed if both of the following statements are true:

- Any of the **RNAME**, **RQNAME**, or **XMITQ** parameters are changed
- One or more applications has a queue open that resolved through this definition as a queue manager alias

**Note:** **FORCE** is not required if this definition is in use as a reply-to queue alias only.

If **FORCE** is not specified in the circumstances described, the command is unsuccessful.

## GET

Specifies whether applications are to be permitted to get messages from this queue:

### ENABLED

Messages can be retrieved from the queue, by suitably authorized applications.

### DISABLED

Applications cannot retrieve messages from the queue.

This parameter can also be changed using the MQSET API call.

## HARDENBO & NOHARDENBO

Specifies whether the count of the number of times that a message was backed out is hardened.

When the count is hardened, the value of the **BackoutCount** field of the message descriptor is written to the log before the message is returned by an MQGET operation. Writing the value to the log ensures that the value is accurate across restarts of the queue manager.

This parameter is supported only on local and model queues.


When the backout count is hardened, the performance of MQGET operations for persistent messages on this queue is impacted.

### HARDENBO

The message backout count for messages on this queue is hardened to ensure that the count is accurate.

### NOHARDENBO

The message backout count for messages on this queue is not hardened and might not be accurate over queue manager restarts.

**Note:**  This parameter affects only z/OS. You can set this parameter on Multiplatforms but it is ineffective.

## IMGRCOVQ

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used. Possible values are:

### YES

These queue objects are recoverable.

### NO

The “[rcdmqimg \(record media image\)](#)” on page 139 and “[rcrmqobj \(re-create object\)](#)” on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

## QMGR

If you specify QMGR, and the **IMGRCOVQ** attribute for the queue manager specifies YES, these queue objects are recoverable.

If you specify QMGR and the **IMGRCOVQ** attribute for the queue manager specifies NO, the “[rcdmqimg \(record media image\)](#)” on page 139 and “[rcrmqobj \(re-create object\)](#)” on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

QMGR is the default value.

This parameter is not valid on z/OS.

z/OS

## INDXTYPE

The type of index maintained by the queue manager to expedite MQGET operations on the queue. For shared queues, the type of index determines the type of MQGET operations that can be used.

This parameter is supported only on z/OS.

This parameter is supported only on local and model queues.

Messages can be retrieved using a selection criterion only if an appropriate index type is maintained, as the following table shows:

Retrieval selection criterion	Index type required	
	Shared queue	Other queue
None (sequential retrieval)	Any	Any
Message identifier	MSGID or NONE	Any
Correlation identifier	CORRELID	Any
Message and correlation identifiers	MSGID or CORRELID	Any
Group identifier	GROUPID	Any
Grouping	GROUPID	GROUPID
Message token	Not allowed	MSGTOKEN

where the value of **INDXTYPE** parameter has the following values:

### NONE

No index is maintained. Use NONE when messages are typically retrieved sequentially or use both the message identifier and the correlation identifier as a selection criterion on the MQGET call.

### MSGID

An index of message identifiers is maintained. Use MSGID when messages are typically retrieved using the message identifier as a selection criterion on the MQGET call with the correlation identifier set to NULL.

### CORRELID

An index of correlation identifiers is maintained. Use CORRELID when messages are typically retrieved using the correlation identifier as a selection criterion on the MQGET call with the message identifier set to NULL.

### GROUPID

An index of group identifiers is maintained. Use GROUPID when messages are retrieved using message grouping selection criteria.

#### Note:

1. You cannot set **INDXTYPE** to GROUPID if the queue is a transmission queue.
2. The queue must use a CF structure at CFLEVEL (3), to specify a shared queue with **INDXTYPE(GROUPID)**.

**z/OS MSGTOKEN**

An index of message tokens is maintained. Use MSGTOKEN when the queue is a WLM-managed queue that you are using with the Workload Manager functions of z/OS.

**Note:** You cannot set **INDXTYPE** to MSGTOKEN if:

- The queue is a model queue with a definition type of SHAREDYN
- The queue is a temporary dynamic queue
- The queue is a transmission queue
- You specify **QSGDISP**(SHARED)

For queues that are not shared and do not use grouping or message tokens, the index type does not restrict the type of retrieval selection. However, the index is used to expedite **GET** operations on the queue, so choose the type that corresponds to the most common retrieval selection.

If you are altering or replacing an existing local queue, you can change the **INDXTYPE** parameter only in the cases indicated in the following table:

*Table 141. Index type change permitted depending upon queue-sharing and presence of messages in the queue*

Queue type		NON-SHARED			SHARED	
Queue state		Uncommitted activity	No uncommitted activity, messages present	No uncommitted activity, and empty	Open or messages present	Not open, and empty
Change <b>INDXTYPE</b> from:	To:	Change allowed?				
NONE	MSGID	No	Yes	Yes	No	Yes
NONE	CORRELID	No	Yes	Yes	No	Yes
NONE	MSGTOKEN	No	No	Yes	-	-
NONE	GROUPLD	No	No	Yes	No	Yes
MSGID	NONE	No	Yes	Yes	No	Yes
MSGID	CORRELID	No	Yes	Yes	No	Yes
MSGID	MSGTOKEN	No	No	Yes	-	-
MSGID	GROUPLD	No	No	Yes	No	Yes
CORRELID	NONE	No	Yes	Yes	No	Yes
CORRELID	MSGID	No	Yes	Yes	No	Yes
CORRELID	MSGTOKEN	No	No	Yes	-	-
CORRELID	GROUPLD	No	No	Yes	No	Yes
MSGTOKEN	NONE	No	Yes	Yes	-	-
MSGTOKEN	MSGID	No	Yes	Yes	-	-
MSGTOKEN	CORRELID	No	Yes	Yes	-	-
MSGTOKEN	GROUPLD	No	No	Yes	-	-
GROUPLD	NONE	No	No	Yes	No	Yes
GROUPLD	MSGID	No	No	Yes	No	Yes

Table 141. Index type change permitted depending upon queue-sharing and presence of messages in the queue (continued)

Queue type		NON-SHARED			SHARED	
GROUPID	CORRELID	No	No	Yes	No	Yes
GROUPID	MSGTOKEN	No	No	Yes	-	-

### INITQ(string)

The local name of the initiation queue on this queue manager, to which trigger messages relating to this queue are written. See [Rules for naming IBM MQ objects](#).

This parameter is supported only on local and model queues.

### LIKE(qtype-name)

The name of a queue, with parameters that are used to model this definition.

If this field is not completed, the values of undefined parameter fields are taken from one of the following definitions. The choice depends on the queue type:

Table 142. Queue types and their corresponding definitions

Queue type	Definition
Alias queue	SYSTEM.DEFAULT.ALIAS.QUEUE
Local queue	SYSTEM.DEFAULT.LOCAL.QUEUE
Model queue	SYSTEM.DEFAULT.MODEL.QUEUE
Remote queue	SYSTEM.DEFAULT.REMOTE.QUEUE

For example, not completing this parameter is equivalent to defining the following value of **LIKE** for an alias queue:

```
LIKE(SYSTEM.DEFAULT.ALIAS.QUEUE)
```

If you require different default definitions for all queues, alter the default queue definitions instead of using the **LIKE** parameter.

**z/OS** On z/OS, the queue manager searches for an object with the name and queue type you specify with a disposition of QMGR, COPY, or SHARED. The disposition of the **LIKE** object is not copied to the object you are defining.

### Note:

1. **QSGDISP**(GROUP) objects are not searched.
2. **LIKE** is ignored if **QSGDISP**(COPY) is specified.

### **z/OS** **ALW** **MAXDEPTH(integer)**

The maximum number of messages allowed on the queue.

This parameter is supported only on local and model queues.

On the following platforms, specify a value in the range zero through 999999999:

- **ALW** AIX, Linux, and Windows
- **z/OS** z/OS

On any other IBM MQ platform, specify a value in the range zero through 640000.

Other factors can still cause the queue to be treated as full, for example, if there is no further hard disk space available.

If this value is reduced, any messages that are already on the queue that exceed the new maximum remain intact.

### **Multi** **MAXFSIZE**

The maximum size, in megabytes, that a queue file can grow to. It is possible for a queue file to exceed this size if you have configured the value to be lower than the current queue file size.

If that happens the queue file no longer accepts new messages, but allows existing messages to be consumed. When the queue file size has dropped below the configured value, new messages can be put to the queue.

**Note:** This figure can differ from the value of the attribute configured on the queue, because internally the queue manager might need to use a larger block size to reach the chosen size. See [Modifying IBM MQ queue files](#) for more information on changing the size of queue files and block size and granularity.

When the granularity needs changing because this attribute has been increased, warning message AMQ7493W Granularity changed is written to the AMQERR logs. This gives you an indication that you need to plan for the queue to be emptied, in order for IBM MQ to adopt the new granularity.

Specify a value greater than or equal to 20, and less than or equal to 267,386,880.

The default value for this attribute is *DEFAULT*, which equates to a hard-coded value of 2,088,960 MB, the maximum for a queue in versions of IBM MQ prior to IBM MQ 9.1.5.

### **MAXMSGL(integer)**

The maximum length (in bytes) of messages on this queue.

This parameter is supported only on local and model queues.

**ALW** On AIX, Linux, and Windows, specify a value in the range zero to the maximum message length for the queue manager. See the **MAXMSGL** parameter of the ALTER QMGR command, [ALTER QMGR MAXMSGL](#).

**z/OS** On z/OS, specify a value in the range zero through 100 MB (104 857 600 bytes).

Message length includes the length of user data and the length of headers. For messages put on the transmission queue, there are additional transmission headers. Allow an additional 4000 bytes for all the message headers.

If this value is reduced, any messages that are already on the queue with length that exceeds the new maximum are not affected.

Applications can use this parameter to determine the size of buffer for retrieving messages from the queue. Therefore, the value can be reduced only if it is known that this reduction does not cause an application to operate incorrectly.

Note that by adding the digital signature and key to the message, [Advanced Message Security](#) increases the length of the message.

### **MONQ**

Controls the collection of online monitoring data for queues.

This parameter is supported only on local and model queues.

#### **QMGR**

Collect monitoring data according to the setting of the queue manager parameter **MONQ**.

#### **OFF**

Online monitoring data collection is turned off for this queue.

#### **LOW**

If the value of the **MONQ** parameter of the queue manager is not NONE, online monitoring data collection is turned on for this queue.

#### **MEDIUM**

If the value of the **MONQ** parameter of the queue manager is not NONE, online monitoring data collection is turned on for this queue.

## HIGH

If the value of the **MONQ** parameter of the queue manager is not NONE, online monitoring data collection is turned on for this queue.

There is no distinction between the values LOW, MEDIUM, and HIGH. These values all turn data collection on, but do not affect the rate of collection.

When this parameter is used in an **ALTER** queue command, the change is effective only when the queue is next opened.

## MSGDLVSQ

Message delivery sequence.

This parameter is supported only on local and model queues.

## PRIORITY


Messages are delivered (in response to MQGET API calls) in first-in-first-out (FIFO) order within priority.

## FIFO

Messages are delivered (in response to MQGET API calls) in FIFO order. Priority is ignored for messages on this queue.

The message delivery sequence parameter can be changed from PRIORITY to FIFO while there are messages on the queue. The order of the messages already on the queue is not changed. Messages added to the queue later take the default priority of the queue, and so might be processed before some of the existing messages.

If the message delivery sequence is changed from FIFO to PRIORITY, the messages put on the queue while the queue was set to FIFO take the default priority.

**Note:**  If **INDXTYPE**(GROUPID) is specified with **MSGDLVSQ**(PRIORITY), the priority in which groups are retrieved is based on the priority of the first message within each group. The priorities 0 and 1 are used by the queue manager to optimize the retrieval of messages in logical order. The first message in each group must not use these priorities. If it does, the message is stored as if it was priority two.

## **NPMCLASS**


The level of reliability to be assigned to non-persistent messages that are put to the queue:

### NORMAL

Non-persistent messages are lost after a failure, or queue manager shutdown. These messages are discarded on a queue manager restart.

### HIGH

The queue manager attempts to retain non-persistent messages on this queue over a queue manager restart or switch over.

 You cannot set this parameter on z/OS.

## **PROCESS(string)**



The local name of the IBM MQ process.

This parameter is supported only on local and model queues.

This parameter is the name of a process instance that identifies the application started by the queue manager when a trigger event occurs; see [Rules for naming IBM MQ objects](#).

The process definition is not checked when the local queue is defined, but it must be available for a trigger event to occur.

If the queue is a transmission queue, the process definition contains the name of the channel to be started. This parameter is optional for transmission queues on the following platforms:

-  IBM i
-  AIX, Linux, and Windows

-  z/OS

If you do not specify it, the channel name is taken from the value specified for the **TRIGDATA** parameter.

### **PROPCTL**

Property control attribute. The attribute is optional. It is applicable to local, alias, and model queues.

**Note:** If your application is opening an alias queue you must set this value on both the alias and target queues.

**PROPCTL** options are as follows. The options do not affect message properties in the MQMD or MQMD extension.

#### **ALL**

Set ALL so that an application can read all the properties of the message either in MQRFH2 headers, or as properties of the message handle.

The ALL option enables applications that cannot be changed to access all the message properties from MQRFH2 headers. Applications that can be changed, can access all the properties of the message as properties of the message handle.

In some cases, the format of data in MQRFH2 headers in the received message might be different to the format in the message when it was sent.

#### **COMPAT**

Set COMPAT so that unmodified applications that expect JMS-related properties to be in an MQRFH2 header in the message data continue to work as before. Applications that can be changed, can access all the properties of the message as properties of the message handle.

If the message contains a property with a prefix of `mcd.`, `jms.`, `usr.`, or `mqext.`, all message properties are delivered to the application. If no message handle is supplied, properties are returned in an MQRFH2 header. If a message handle is supplied, all properties are returned in the message handle.

If the message does not contain a property with one of those prefixes, and the application does not provide a message handle, no message properties are returned to the application. If a message handle is supplied, all properties are returned in the message handle.

In some cases, the format of data in MQRFH2 headers in the received message might be different to the format in the message when it was sent.

#### **FORCE**

Force all applications to read message properties from MQRFH2 headers.

Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

A valid message handle supplied in the `MsgHandle` field of the `MQGMO` structure on the `MQGET` call is ignored. Properties of the message are not accessible using the message handle.

In some cases, the format of data in MQRFH2 headers in the received message might be different to the format in the message when it was sent.

#### **NONE**

If a message handle is supplied, all the properties are returned in the message handle.

All message properties are removed from the message body before it is delivered to the application.

### **PUT**

Specifies whether messages can be put on the queue.

#### **ENABLED**

Messages can be added to the queue (by suitably authorized applications).


#### **DISABLED**

Messages cannot be added to the queue.

This parameter can also be changed using the MQSET API call.

### **QDEPTHHI(integer)**

The threshold against which the queue depth is compared to generate a Queue Depth High event. This parameter is supported only on local and model queues.


 For more information about the effect that shared queues on z/OS have on this event; see [Shared queues and queue depth events on z/OS](#).

This event indicates that an application put a message on a queue resulting in the number of messages on the queue becoming greater than or equal to the queue depth high threshold. See the **QDPHIEV** parameter.

The value is expressed as a percentage of the maximum queue depth (**MAXDEPTH** parameter), and must be in the range zero through 100 and no less than **QDEPTHLO**.

### **QDEPTHLO(integer)**

The threshold against which the queue depth is compared to generate a Queue Depth Low event. This parameter is supported only on local and model queues.

 For more information about the effect that shared queues on z/OS have on this event; see [Shared queues and queue depth events on z/OS](#).

This event indicates that an application retrieved a message from a queue resulting in the number of messages on the queue becoming less than or equal to the queue depth low threshold. See the **QDPLOEV** parameter.

The value is expressed as a percentage of the maximum queue depth (**MAXDEPTH** parameter), and must be in the range zero through 100 and no greater than **QDEPTHHI**.

### **QDPHIEV**

Controls whether Queue Depth High events are generated.

This parameter is supported only on local and model queues.

A Queue Depth High event indicates that an application put a message on a queue resulting in the number of messages on the queue becoming greater than or equal to the queue depth high threshold. See the **QDEPTHHI** parameter.


#### **ENABLED**

Queue Depth High events are generated.

#### **DISABLED**

Queue Depth High events are not generated.

**Note:** The value of this parameter can change implicitly.

 On z/OS, shared queues affect the event.

For more information about this event, see [Queue Depth High](#).

### **QDPLOEV**

Controls whether Queue Depth Low events are generated.

This parameter is supported only on local and model queues.

A Queue Depth Low event indicates that an application retrieved a message from a queue resulting in the number of messages on the queue becoming less than or equal to the queue depth low threshold. See the **QDEPTHLO** parameter.

#### **ENABLED**

Queue Depth Low events are generated.

#### **DISABLED**

Queue Depth Low events are not generated.

**Note:** The value of this parameter can change implicitly.

 On z/OS, shared queues affect the event.



For more information about this event, see [Queue Depth Low](#).

### QDPMAXEV

Controls whether Queue Full events are generated.

This parameter is supported only on local and model queues.

A Queue Full event indicates that a put to a queue was rejected because the queue is full. The queue depth reached its maximum value.


#### ENABLED

Queue Full events are generated.

#### DISABLED

Queue Full events are not generated.

**Note:** The value of this parameter can change implicitly.

 On z/OS, shared queues affect the event.

For more information about this event, see [Queue Full](#).

### QSGDISP

This parameter applies to z/OS only.

Specifies the disposition of the object within the group.

<i>Table 143. Action of <b>ALTER</b> depending on different values of <b>QSGDISP</b>.</i>	
<b>QSGDISP</b>	<b>ALTER</b>
COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP</b> (COPY). Any object residing in the shared repository, or any object defined using a command that had the parameters <b>QSGDISP</b> (QMGR), is not affected by this command.
GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameters <b>QSGDISP</b>(GROUP). Any object residing on the page set of the queue manager that executes the command (except a local copy of the object), or any object defined using a command that had the parameters <b>QSGDISP</b>(SHARED), is not affected by this command. If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE QUEUE(QNAME) REPLACE QSGDISP(COPY)</pre> <p>The ALTER for the group object takes effect regardless of whether the generated command with <b>QSGDISP</b>(COPY) fails.</p>
PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with <b>QSGDISP</b> (QMGR) or <b>QSGDISP</b> (COPY). Any object residing in the shared repository is unaffected.
QMGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP</b> (QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

Table 143. Action of **ALTER** depending on different values of **QSGDISP**. (continued)

QSGDISP	ALTER
SHARED	This value applies only to local queues. The object definition resides in the shared repository. The object was defined using a command that had the parameters <b>QSGDISP</b> (SHARED). Any object residing on the page set of the queue manager that executes the command, or any object defined using a command that had the parameters <b>QSGDISP</b> (GROUP), is not affected by this command. If the queue is clustered, a command is generated and sent to all active queue managers in the queue sharing group to notify them of this clustered, shared queue.

### **QSVCI EV**

Controls whether Service Interval High or Service Interval OK events are generated.

This parameter is supported only on local and model queues and is ineffective if it is specified on a shared queue.

A Service Interval High event is generated when a check indicates that no messages were retrieved from the queue for at least the time indicated by the **QSVCI NT** parameter.

A Service Interval OK event is generated when a check indicates that messages were retrieved from the queue within the time indicated by the **QSVCI NT** parameter.

**Note:** The value of this parameter can change implicitly. For more information, see the description of the Service Interval High and Service Interval OK events in [Queue Service Interval High](#) and [Queue Service Interval OK](#).

#### **HIGH**

Service Interval High events are generated

#### **OK**

Service Interval OK events are generated

#### **NONE**

No service interval events are generated

### **QSVCI NT(integer)**

The service interval used for comparison to generate Service Interval High and Service Interval OK events.

This parameter is supported only on local and model queues and is ineffective if it is specified on a shared queue.

See the **QSVCI EV** parameter.

The value is in units of milliseconds, and must be in the range zero through 999999999.

### **RETINTVL(integer)**

The number of hours from when the queue was defined, after which the queue is no longer needed. The value must be in the range 0 - 999,999,999.

This parameter is supported only on local and model queues.

The **CRDATE** and **CRTIME** can be displayed using the [DISPLAY QUEUE](#) command.

This information is available for use by an operator or a housekeeping application to delete queues that are no longer required.

**Note:** The queue manager does not delete queues based on this value, nor does it prevent queues from being deleted if their retention interval is not expired. It is the responsibility of the user to take any required action.

### **RNAME(string)**

Name of remote queue. This parameter is the local name of the queue as defined on the queue manager specified by **RQMNAME**.

This parameter is supported only on remote queues.

- If this definition is used for a local definition of a remote queue, **RNAME** must not be blank when the open occurs.
- If this definition is used for a queue manager alias definition, **RNAME** must be blank when the open occurs.

In a queue manager cluster, this definition applies only to the queue manager that made it. To advertise the alias to the whole cluster, add the **CLUSTER** attribute to the remote queue definition.

- If this definition is used for a reply-to queue alias, this name is the name of the queue that is to be the reply-to queue.

The name is not checked to ensure that it contains only those characters normally allowed for queue names; see [Rules for naming IBM MQ objects](#).

### **RQMNAME(string)**

The name of the remote queue manager on which the queue **RNAME** is defined.

This parameter is supported only on remote queues.

- If an application opens the local definition of a remote queue, **RQMNAME** must not be blank or the name of the local queue manager. When the open occurs, if **XMITQ** is blank there must be a local queue of this name, which is to be used as the transmission queue.
- If this definition is used for a queue manager alias, **RQMNAME** is the name of the queue manager that is being aliased. It can be the name of the local queue manager. Otherwise, if **XMITQ** is blank, when the open occurs there must be a local queue of this name, which is to be used as the transmission queue.
- If **RQMNAME** is used for a reply-to queue alias, **RQMNAME** is the name of the queue manager that is to be the reply-to queue manager.

The name is not checked to ensure that it contains only those characters normally allowed for IBM MQ object names; see [Rules for naming IBM MQ objects](#).

### **ALW SCOPE**

Specifies the scope of the queue definition.

This parameter is supported only on alias, local, and remote queues.

### **QMGR**

The queue definition has queue manager scope. This means that the definition of the queue does not extend beyond the queue manager that owns it. You can open a queue for output that is owned by another queue manager in either of two ways:

1. Specify the name of the owning queue manager.
2. Open a local definition of the queue on the other queue manager.

### **CELL**

The queue definition has cell scope. Cell scope means that the queue is known to all the queue managers in the cell. A queue with cell scope can be opened for output merely by specifying the name of the queue. The name of the queue manager that owns the queue need not be specified.

If there is already a queue with the same name in the cell directory, the command fails. The **REPLACE** option does not affect this situation.

This value is valid only if a name service supporting a cell directory is configured.

**Restriction:** The DCE name service is no longer supported.

This parameter is valid only on AIX, Linux, and Windows.

### **SHARE and NOSHARE**

Specifies whether multiple applications can get messages from this queue.

This parameter is supported only on local and model queues.

### **SHARE**

More than one application instance can get messages from the queue.

## NOSHARE

Only a single application instance can get messages from the queue.

## V 9.3.0 V 9.3.0 STATQ


Specifies whether statistics data collection is enabled:

### QMGR

Statistics data collection is based on the setting of the **STATQ** parameter of the queue manager.

### ON

If the value of the **STATQ** parameter of the queue manager is not NONE, statistics data collection for the queue is enabled.

 On z/OS systems, you must enable class 5 statistics using the START TRACE command.

### OFF

Statistics data collection for the queue is disabled.

If this parameter is used in an **ALTER** queue command, the change is effective only for connections to the queue manager made after the change to the parameter.

## STGCLASS(string)

The name of the storage class.

This parameter is supported only on local and model queues.

**Note:** You can change this parameter only if the queue is empty and closed.

This parameter is an installation-defined name. The first character of the name must be uppercase A through Z, and subsequent characters either uppercase A through Z or numeric 0 through 9.

This parameter is valid only on z/OS; see [Storage classes](#).



## V 9.3.0 STREAMQ

The name of a secondary queue where a copy of each message is put.



**Attention:** If the user setting the **STREAMQ** attribute does not have change authority on the chosen stream queue, the command fails with error message AMQ8135E, or the equivalent message CSQ9016E on z/OS.

In addition, if the stream queue does not exist, error message AMQ8135E (CSQ9016E on z/OS) is returned instead of AMQ8147E, or the equivalent message CSQM125I on z/OS.

  See [Streaming queue restrictions](#) for information on when you can set STREAMQ.

## V 9.3.0 STRMQOS

The quality of service to use when delivering messages to the streaming queue.

The value can be one of:

### BESTEF

If the original message can be delivered, but the streamed message cannot, the original message is still delivered to its queue.

This is the default value.

### MUSTDUP

The queue manager ensures that both the original message and the streamed message are successfully delivered to their queues.

If, for some reason, the streamed message cannot be delivered to its queue, then the original message is not delivered to its queue either. The putting application receives an error reason code and must try to put the message again.

**TARGET(string)**

The name of the queue or topic object being aliased; See [Rules for naming IBM MQ objects](#). The object can be a queue or a topic as defined by **TARGETYPE**. The maximum length is 48 characters.

This parameter is supported only on alias queues.

This object needs to be defined only when an application process opens the alias queue.

The TARGQ parameter, defined in IBM WebSphere MQ 6.0, is renamed to TARGET from version 7.0 and generalized to allow you to specify the name of either a queue or a topic. The default value for TARGET is a queue, therefore TARGET(my\_queue\_name) is the same as TARGQ(my\_queue\_name). The TARGQ attribute is retained for compatibility with your existing programs. If you specify **TARGET**, you cannot also specify **TARGQ**.

**TARGETYPE(string)**

The type of object to which the alias resolves.

**QUEUE (default)**

The alias resolves to a queue.

**TOPIC**

The alias resolves to a topic.

**TRIGDATA(string)**

The data that is inserted in the trigger message. The maximum length of the string is 64 bytes.

This parameter is supported only on local and model queues.

For a transmission queue, you can use this parameter to specify the name of the channel to be started.

This parameter can also be changed using the MQSET API call.

**TRIGDPATH(integer)**

The number of messages that have to be on the queue before a trigger message is written, if **TRIGTYPE** is DEPTH. The value must be in the range 1 - 999,999,999. The default value is 1.

This parameter is supported only on local and model queues.

This parameter can also be changed using the MQSET API call.

**TRIGGER & NOTRIGGER**

Specifies whether trigger messages are written to the initiation queue, named by the **INITQ** parameter, to trigger the application, named by the **PROCESS** parameter:

**TRIGGER**

Triggering is active, and trigger messages are written to the initiation queue.

**NOTRIGGER**

Triggering is not active, and trigger messages are not written to the initiation queue. This is the default value.

This parameter is supported only on local and model queues.

This parameter can also be changed using the MQSET API call.

**TRIGMPRI(integer)**

The message priority number that triggers this queue. The value must be in the range zero through to the **MAXPRTY** queue manager parameter; see [“DISPLAY QMGR \(display queue manager settings\)” on page 765](#) for details. The default value is zero.

This parameter can also be changed using the MQSET API call.

**TRIGTYPE**

Specifies whether and under what conditions a trigger message is written to the initiation queue. The initiation queue is (named by the **INITQ** parameter).

This parameter is supported only on local and model queues.

**FIRST**

Whenever the first message of priority equal to or greater than the priority specified by the **TRIGMPRI** parameter of the queue arrives on the queue. This is the default value.

**EVERY**

Every time a message arrives on the queue with priority equal to or greater than the priority specified by the **TRIGMPRI** parameter of the queue.

**DEPTH**

When the number of messages with priority equal to or greater than the priority specified by **TRIGMPRI** is equal to the number indicated by the **TRIGDPH** parameter.

**NONE**

No trigger messages are written.

This parameter can also be changed using the MQSET API call.

**USAGE**

Queue usage.

This parameter is supported only on local and model queues.

**NORMAL**

The queue is not a transmission queue.

**XMITQ**

The queue is a transmission queue, which is used to hold messages that are destined for a remote queue manager. When an application puts a message to a remote queue, the message is stored on the appropriate transmission queue. It stays there, awaiting transmission to the remote queue manager.

If you specify this option, do not specify values for **CLUSTER** and **CLUSNL**.

**z/OS** Additionally, on z/OS, do not specify **INDXTYPE**(MSGTOKEN) or **INDXTYPE**(GROUPID).

**XMITQ(string)**

The name of the transmission queue to be used for forwarding messages to the remote queue. **XMITQ** is used with either remote queue or queue manager alias definitions.

This parameter is supported only on remote queues.

If **XMITQ** is blank, a queue with the same name as **RQMNAME** is used as the transmission queue.

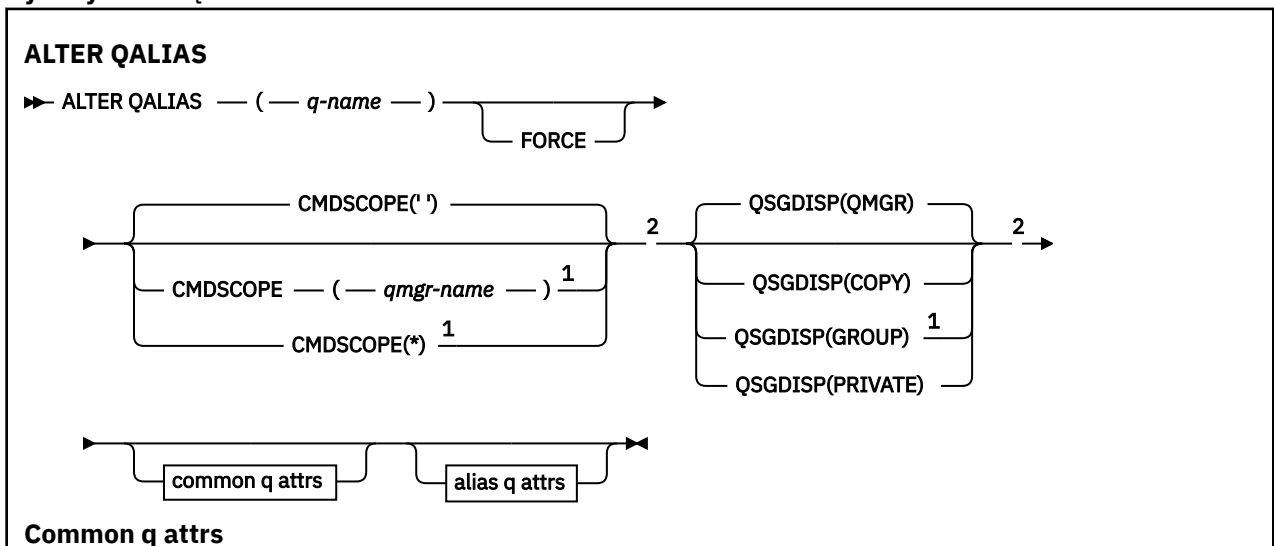
This parameter is ignored if the definition is being used as a queue manager alias and **RQMNAME** is the name of the local queue manager.

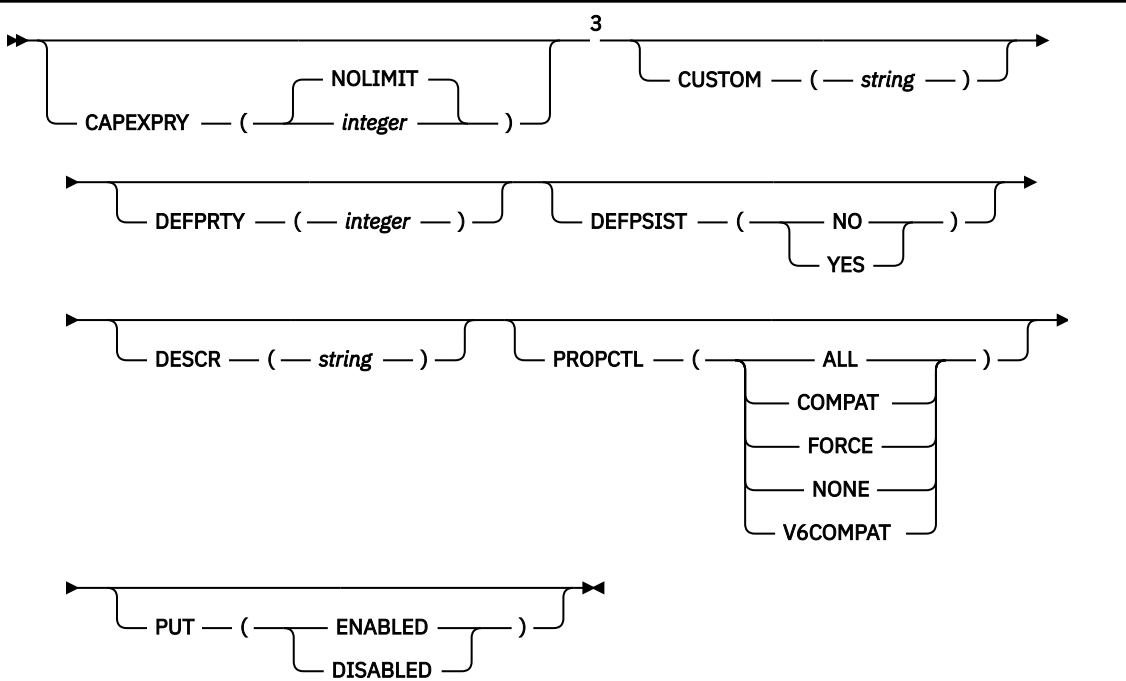
It is also ignored if the definition is used as a reply-to queue alias definition.

**ALTER QALIAS**

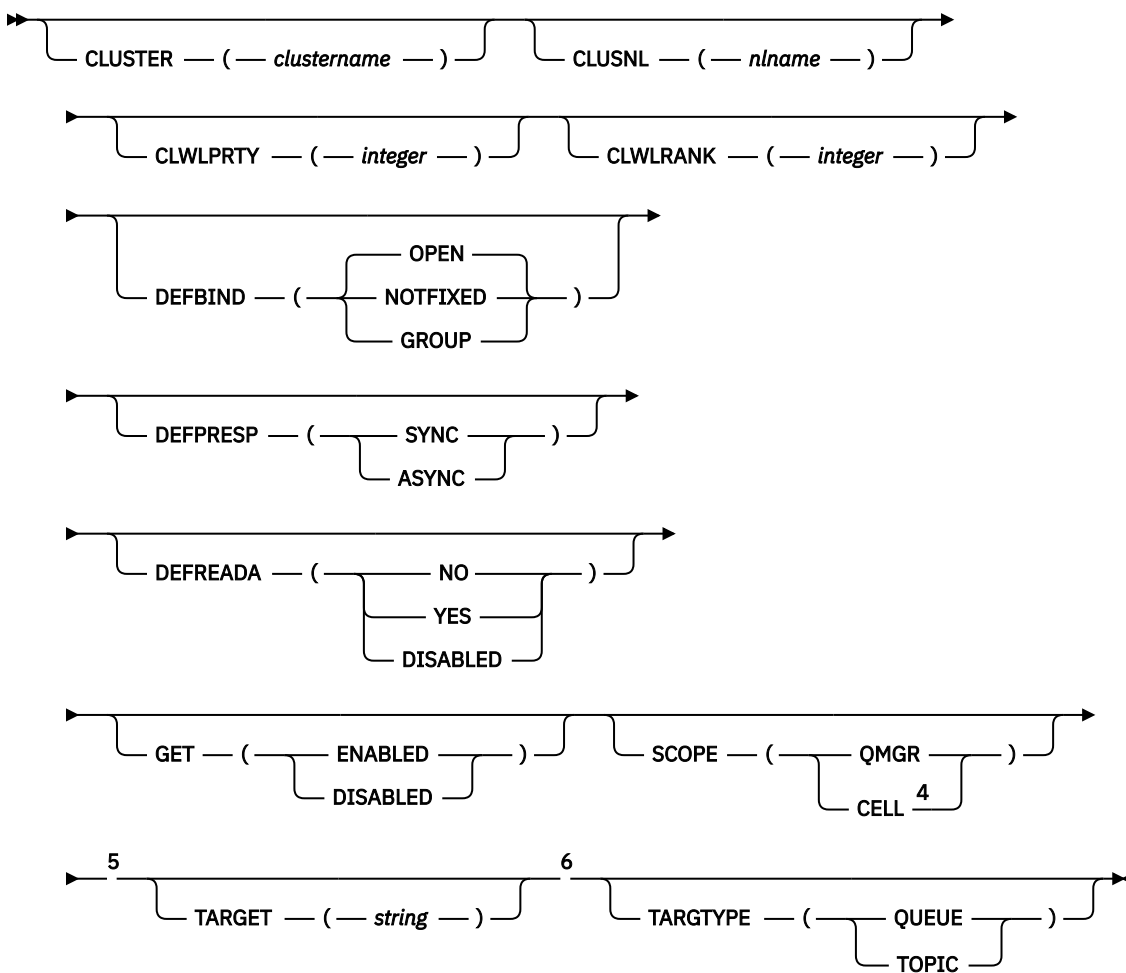
Use the MQSC command **ALTER QALIAS** to alter the parameters of an alias queue.

**Synonym:** ALT QA





**Alias q attr**



Notes:

<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

- <sup>2</sup> Valid only on z/OS.
- <sup>3</sup> Not valid on z/OS.
- <sup>4</sup> Valid only on AIX, Linux, and Windows systems.
- <sup>5</sup> Not valid on z/OS.
- <sup>6</sup> The TARGTYPE parameter is available for compatibility with previous releases. It is a synonym of TARGET; you cannot specify both parameters.

The parameters are described in [“ALTER queues \(alter queue settings\)”](#) on page 382.

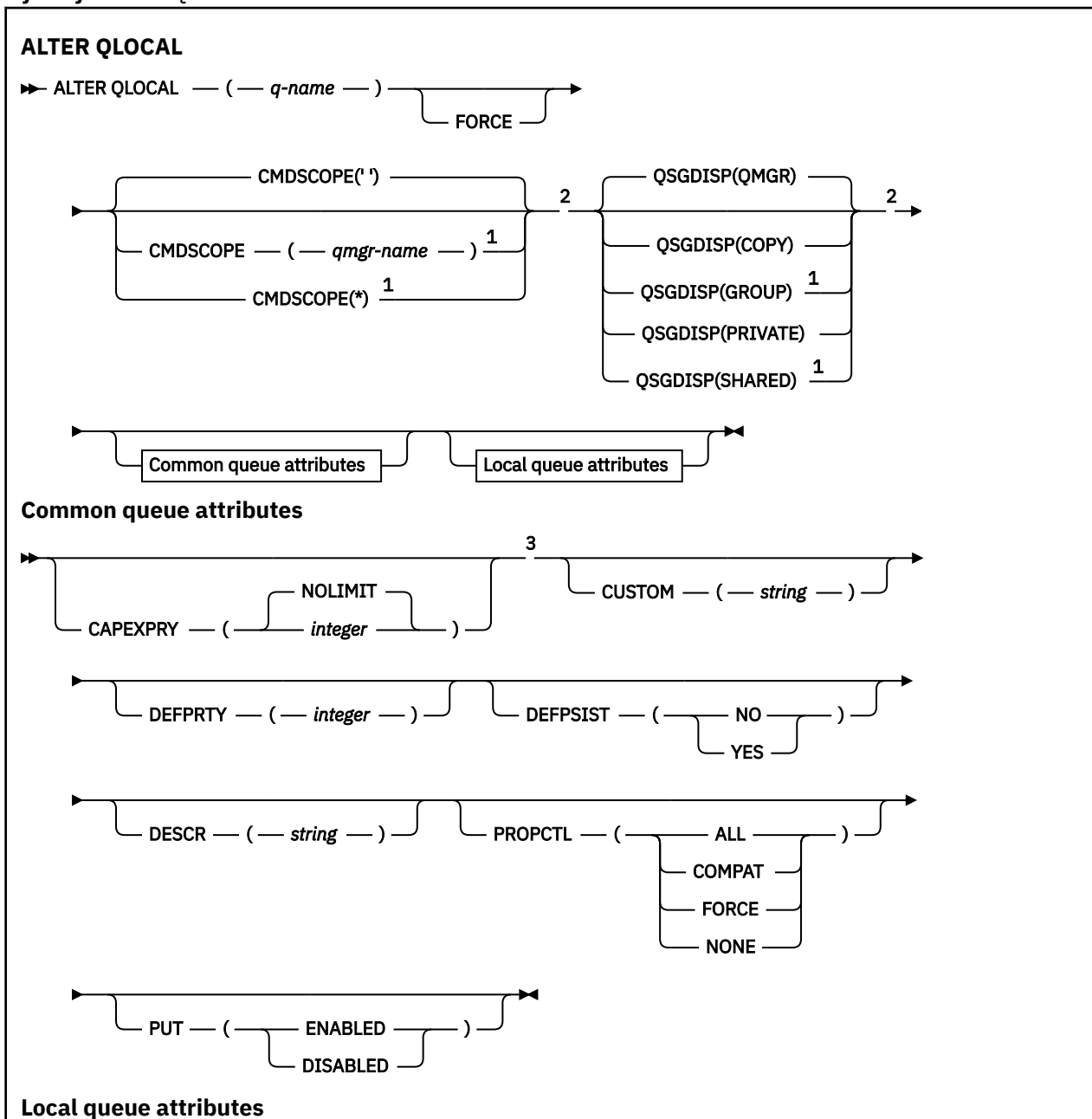
### Related concepts

[Working with alias queues](#)

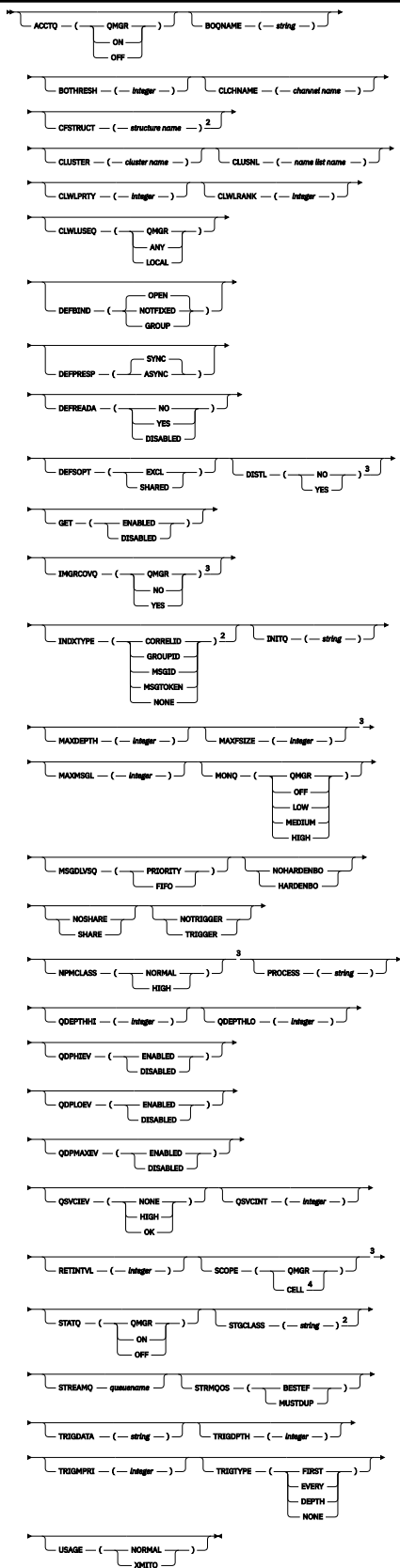
## ALTER QLOCAL

Use the MQSC command **ALTER QLOCAL** to alter the parameters of a local queue.

**Synonym:** ALT QL







Notes:

- 1 Valid only on z/OS when the queue manager is a member of a queue sharing group.
- 2 Valid only on z/OS.

<sup>3</sup> Not valid on z/OS.

<sup>4</sup> Valid only on AIX, Linux, and Windows systems.

The parameters are described in “ALTER queues (alter queue settings)” on page 382.

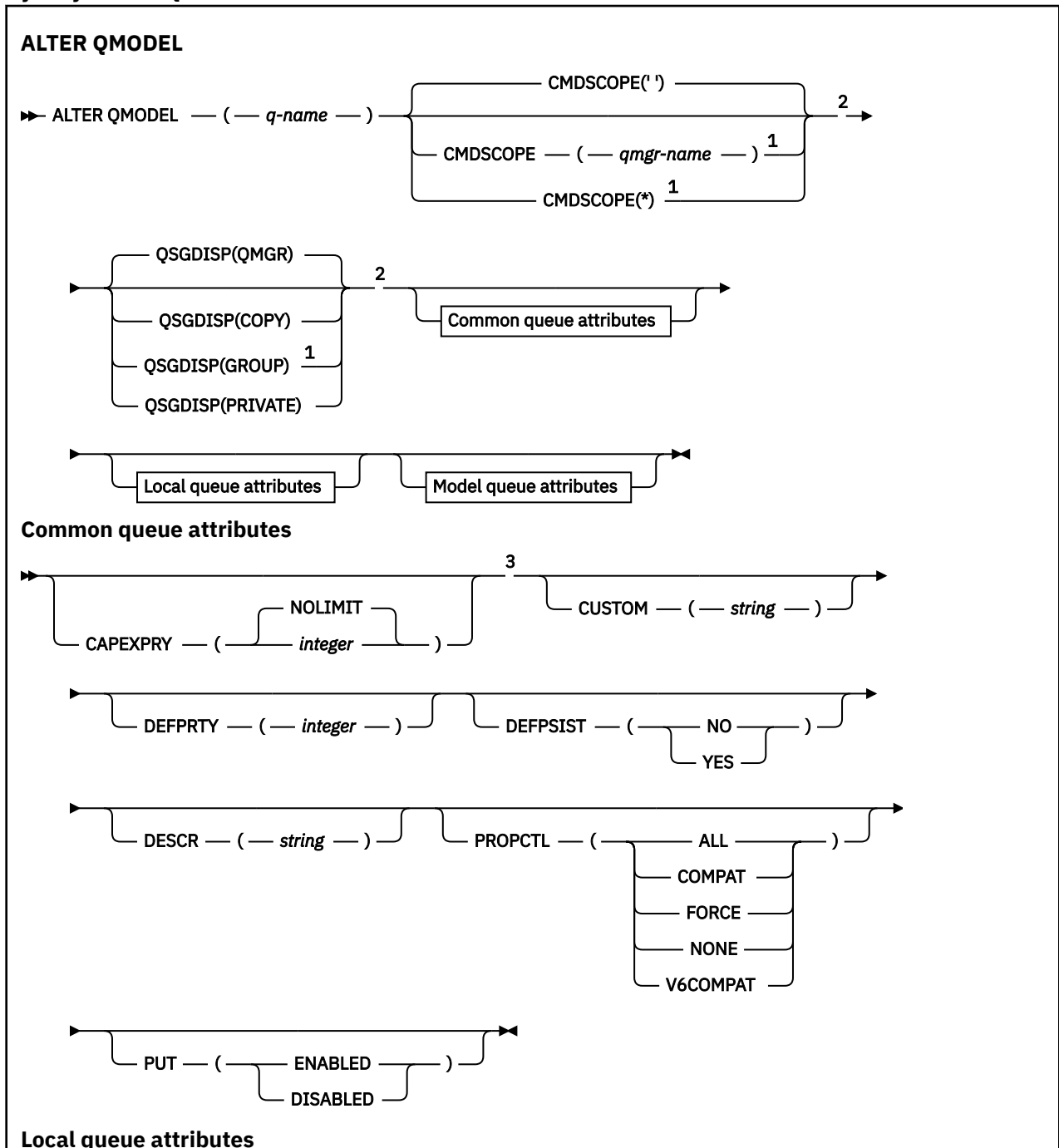
### Related tasks

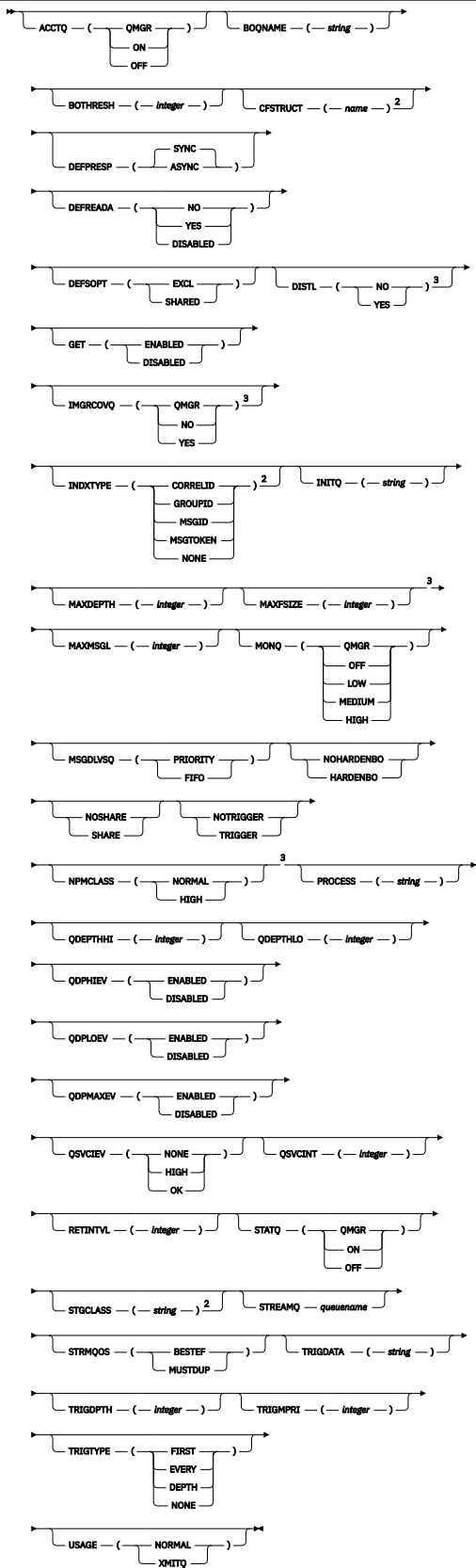
[Changing local queue attributes](#)

## ALTER QMODEL

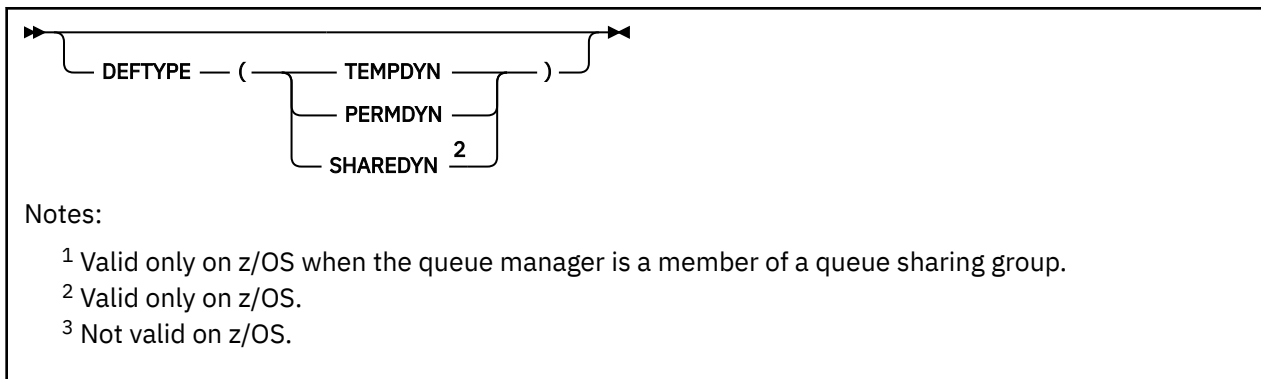
Use the MQSC command **ALTER QMODEL** to alter the parameters of a model queue.

**Synonym:** ALT QM





**Model queue attributes**



The parameters are described in “ALTER queues (alter queue settings)” on page 382.

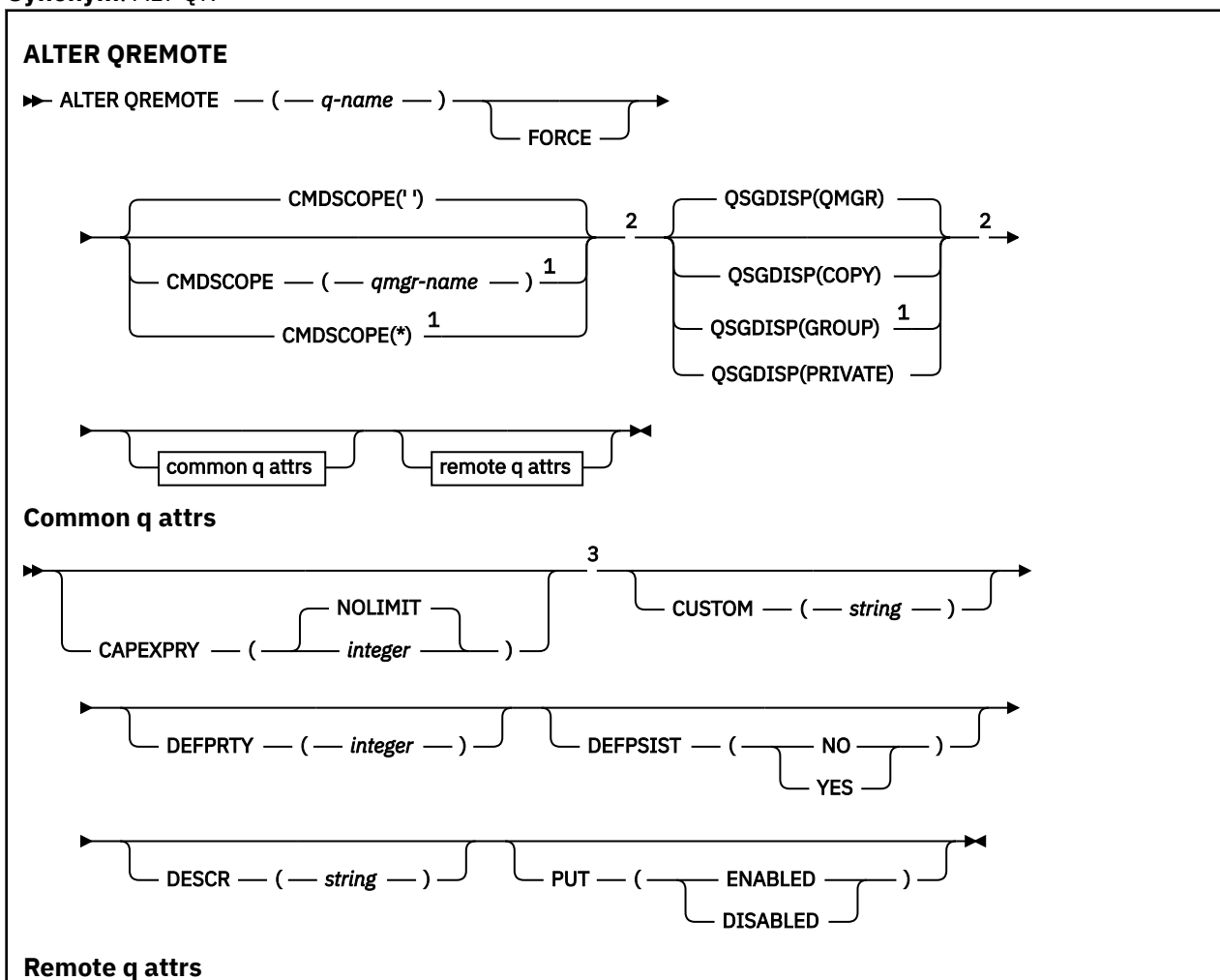
**Related concepts**

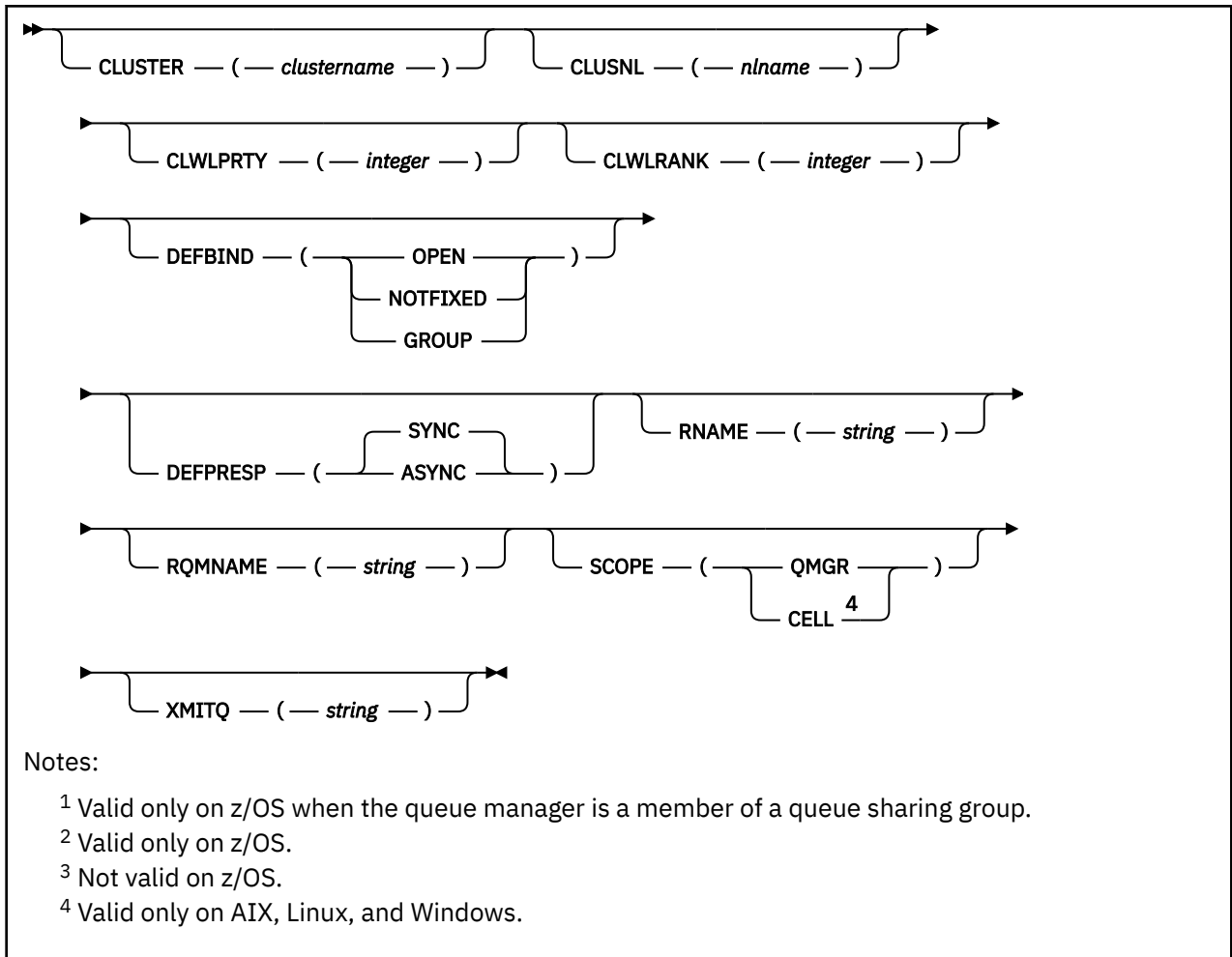
Working with model queues

**ALTER QREMOTE**

Use the MQSC command **ALTER QREMOTE** to alter the parameters of a local definition of a remote queue, a queue manager alias, or a reply-to queue alias.

**Synonym:** ALT QR





The parameters are described in [“ALTER queues \(alter queue settings\)”](#) on page 382.

## **ALTER SECURITY (alter security options) on z/OS**

Use the MQSC command **ALTER SECURITY** to define system-wide security options.

### Using MQSC commands on z/OS

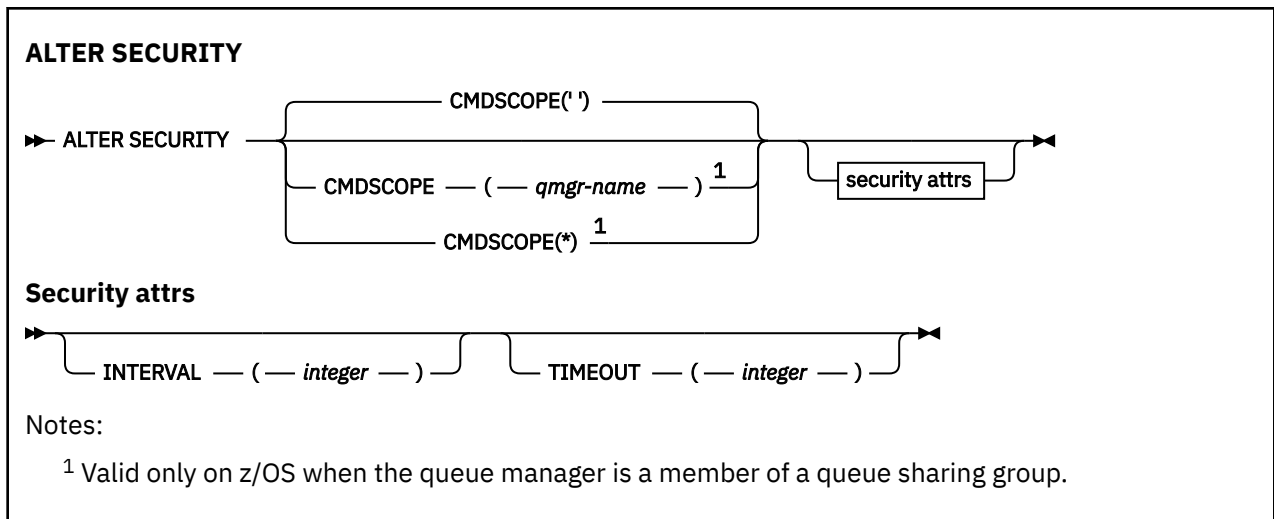
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Parameters not specified in the **ALTER SECURITY** command result in the existing values for those parameters being left unchanged.

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for ALTER SECURITY”](#) on page 414

**Synonym:** ALT SEC



## Parameter descriptions for ALTER SECURITY

The parameters you specify override the current parameter values. Attributes that you do not specify are unchanged.

**Note:** If you do not specify any parameters, the command completes successfully, but no security options are changed.

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

### **INTERVAL**(*integer*)

The interval between checks for user IDs and their associated resources to determine whether the **TIMEOUT** has expired. The value is in minutes, in the range zero through 10080 (one week). If **INTERVAL** is specified as zero, no user timeouts occur.

### **TIMEOUT**(*integer*)

How long security information about an unused user ID and associated resources is retained by IBM MQ. The value specifies a number of minutes in the range zero through 10080 (one week). If **TIMEOUT** is specified as zero, and **INTERVAL** is nonzero, all such information is discarded by the queue manager every **INTERVAL** number of minutes.

The length of time that an unused user ID and associated resources are retained by IBM MQ depends on the value of **INTERVAL**. The user ID times out at a time between **TIMEOUT** and **TIMEOUT** plus **INTERVAL**.

When the **TIMEOUT** and **INTERVAL** parameters are changed, the previous timer request is canceled and a new timer request is scheduled immediately, using the new **TIMEOUT** value. When the timer request is actioned, a new value for **INTERVAL** is set.

## Related concepts

[User ID timeouts](#)

## **Multi** ALTER SERVICE (alter a service definition) on Multiplatforms

Use the MQSC command **ALTER SERVICE** to alter the parameters of an existing IBM MQ service definition.

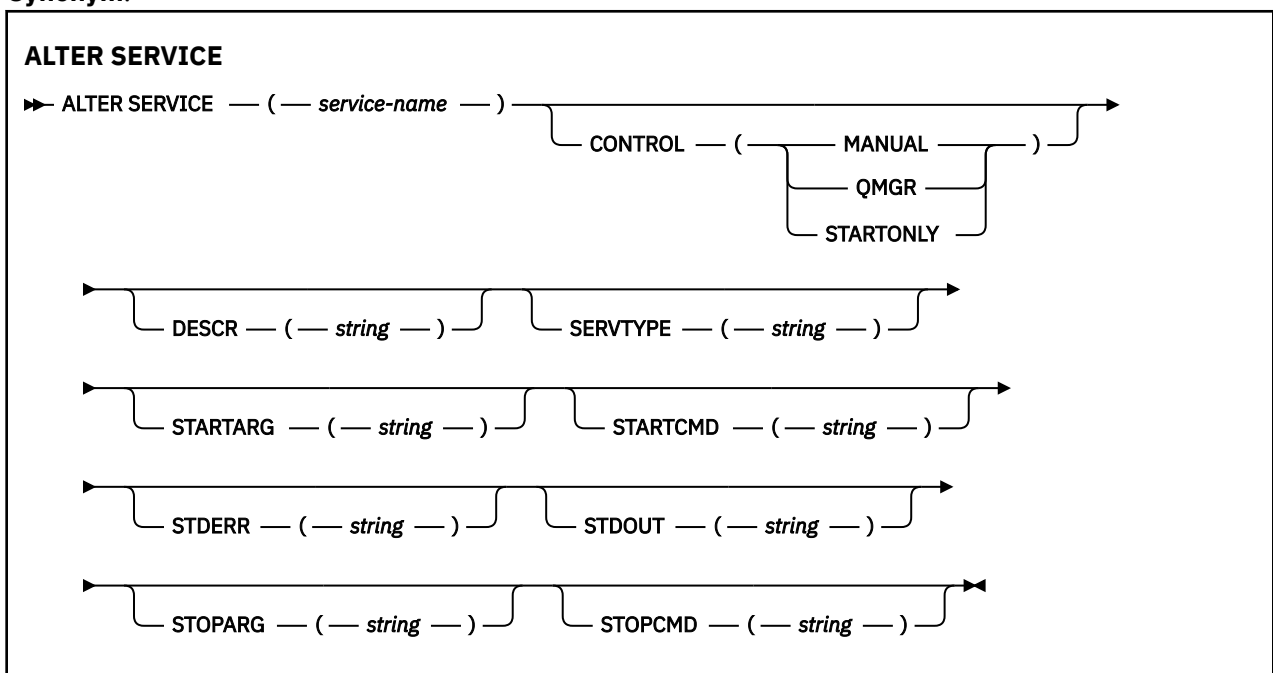
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

Parameters not specified in the **ALTER SERVICE** command result in the existing values for those parameters being left unchanged.

- [Syntax diagram](#)
- [“Parameter descriptions for ALTER SERVICE” on page 415](#)

#### Synonym:



### Parameter descriptions for ALTER SERVICE

The parameter descriptions apply to the **ALTER SERVICE** and **DEFINE SERVICE** commands, with the following exceptions:

- The **LIKE** parameter applies only to the **DEFINE SERVICE** command.
- The **NOREPLACE** and **REPLACE** parameter applies only to the **DEFINE SERVICE** command.

#### (*service-name*)

Name of the IBM MQ service definition (see [Rules for naming IBM MQ objects](#)).

The name must not be the same as any other service definition currently defined on this queue manager (unless **REPLACE** is specified).

#### **CONTROL**(*string*)

Specifies how the service is to be started and stopped:

##### **MANUAL**

The service is not to be started automatically or stopped automatically. It is to be controlled by use of the **START SERVICE** and **STOP SERVICE** commands.

**QMGR**

The service being defined is to be started and stopped at the same time as the queue manager is started and stopped.

**STARTONLY**

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**DESCR(*string*)**

Plain-text comment. It provides descriptive information about the service when an operator issues the **DISPLAY SERVICE** command (see [“DISPLAY SERVICE \(display service information\) on Multiplatforms”](#) on page 823).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

**LIKE(*service-name*)**

The name of a service the parameters of which are used to model this definition.

This parameter applies only to the **DEFINE SERVICE** command.

If this field is not completed, and you do not complete the parameter fields related to the command, the values are taken from the default definition for services on this queue manager. Not completing this parameter is equivalent to specifying:

```
LIKE (SYSTEM.DEFAULT.SERVICE)
```

A default service is provided but it can be altered by the installation of the default values required. See [Rules for naming IBM MQ objects](#).

**REPLACE and NOREPLACE**

Whether the existing definition is to be replaced with this one.

This parameter applies only to the **DEFINE SERVICE** command.

**REPLACE**

The definition must replace any existing definition of the same name. If a definition does not exist, one is created.

**NOREPLACE**

The definition should not replace any existing definition of the same name.

**SERVTYPE**

Specifies the mode in which the service is to run:

**COMMAND**

A command service object. Multiple instances of a command service object can be executed concurrently. You cannot monitor the status of command service objects.

**SERVER**

A server service object. Only one instance of a server service object can be executed at a time. The status of server service objects can be monitored using the **DISPLAY SVSTATUS** command.

**STARTARG(*string*)**

Specifies the arguments to be passed to the user program at queue manager startup.

**STARTCMD(*string*)**

Specifies the name of the program which is to run. You must specify a fully qualified path name to the executable program.

**STDERR(*string*)**

Specifies the path to a file to which the standard error (stderr) of the service program is redirected. If the file does not exist when the service program is started, the file is created. If this value is blank then any data written to stderr by the service program is discarded.



**STDOUT(string)**

Specifies the path to a file to which the standard output (stdout) of the service program is redirected. If the file does not exist when the service program is started, the file is created. If this value is blank then any data written to stdout by the service program is discarded.

**STOPARG(string)**

Specifies the arguments to be passed to the stop program when instructed to stop the service.

**STOPCMD(string)**

Specifies the name of the executable program to run when the service is requested to stop. You must specify a fully qualified path name to the executable program.

Replaceable inserts can be used for any of the **STARTCMD**, **STARTARG**, **STOPCMD**, **STOPARG**, **STDOUT** or **STDERR** strings, for more information, see [Replaceable inserts on service definitions](#).

**Related concepts**

[Working with services](#)

**Related tasks**

[Using a server service object](#)

[Using a command service object](#)

**Related reference**

[“DEFINE SERVICE \(create a new service definition\) on Multiplatforms” on page 581](#)

Use the MQSC command **DEFINE SERVICE** to define a new IBM MQ service definition, and set its parameters.

[“DISPLAY SVSTATUS \(display services status\) on Multiplatforms” on page 843](#)

Use the MQSC command **DISPLAY SVSTATUS** to display status information for one or more services. Only services with a **SERVTYPE** of SERVER are displayed.

[“START SERVICE \(start a service\) on Multiplatforms” on page 955](#)

Use the MQSC command **START SERVICE** to start a service. The identified service definition is started within the queue manager and inherits the environment and security variables of the queue manager.

[“STOP SERVICE \(stop a service\) on Multiplatforms” on page 975](#)

Use the MQSC command **STOP SERVICE** to stop a service.

 **ALTER SMDS (alter shared message data sets) on z/OS**

Use the MQSC command **ALTER SMDS** to alter the parameters of existing IBM MQ definitions relating to one or more shared message data sets associated with a specific application structure. It is only supported when the CFSTRUCT definition is using the option OFFLOAD (SMDS).

**Using MQSC commands on z/OS**

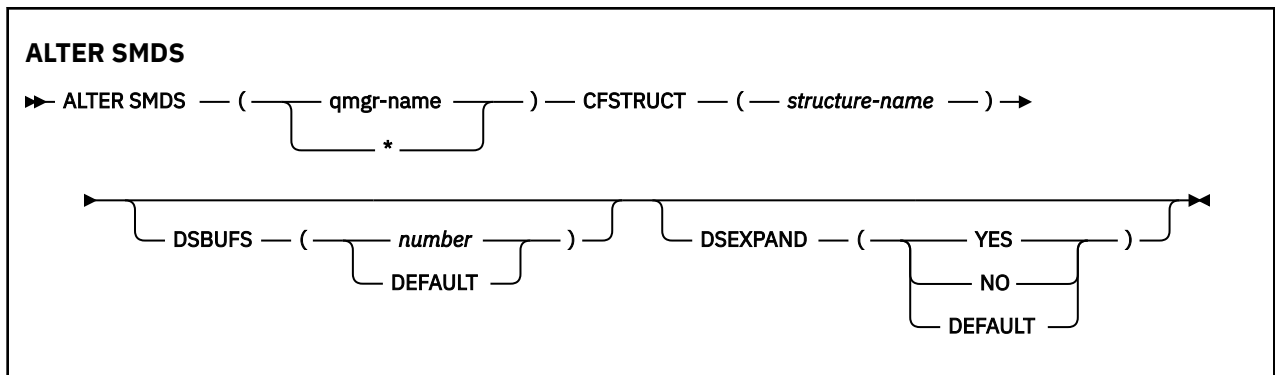
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Parameters not specified in the **ALTER SMDS** command result in the existing values for those parameters being left unchanged.

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for ALTER SMDS” on page 418](#)

**Synonym:**



## Parameter descriptions for ALTER SMDS

### SMDS(*qmgr-name*|\*)

Specify the queue manager for which the shared message data set properties are to be modified, or an asterisk to modify the properties for all data sets associated with the specified CFSTRUCT.

### CFSTRUCT(*structure-name*)

Specify the coupling facility application structure for which the properties of one or more shared message data sets are to be modified.

### DSBUFS(*number*|DEFAULT)

Specify an override value for the number of buffers to be allocated in the specified queue manager or queue managers for accessing shared message data sets for this structure, as a number in the range 1 to 9999, or specify DEFAULT to cancel a previous override and resume using the **DSBUFS** value from the CFSTRUCT definition. The size of each buffer is equal to the logical block size. SMDS buffers are allocated in memory objects residing in z/OS 64-bit storage (above the bar).

When this parameter is altered, any affected queue managers which are already connected to the structure dynamically increase or decrease the number of data set buffers being used for this structure to match the new value. If the specified target value cannot be reached, the affected queue manager replaces the specified **DSBUFS** parameter with the actual new number of buffers. If the queue manager is not active, the change will come into effect when the queue manager is restarted.

### DSEXPAND(YES|NO|DEFAULT)

Specify an override value to be used by the specified queue manager or queue managers to control expansion of shared message data sets for this structure.

This parameter controls whether the queue manager should expand a shared message data set when it becomes nearly full, and further blocks are required in the data set.

#### YES

Expansion is supported.

Each time expansion is required, the data set is expanded by the secondary allocation specified when the data set was defined. If no secondary allocation was specified, or it was specified as zero, then a secondary allocation amount of approximately 10% of the existing size is used.

#### NO

No automatic data set expansion is to take place.

#### DEFAULT

Cancels a previous override.

If you used DEFAULT to cancel a previous override it resumes using the **DSEXPAND** value from the CFSTRUCT definition.

If an expansion attempt fails, the **DSEXPAND** override for the affected queue manager is automatically changed to NO to prevent further expansion attempts, but it can be changed back to YES using the **ALTER SMDS** command to enable further expansion attempts.

When this parameter is altered, any affected queue managers which are already connected to the structure immediately start using the new parameter value.

## **z/OS** ALTER STGCLASS (alter storage class settings) on z/OS

Use the MQSC command **ALTER STGCLASS** to alter the characteristics of a storage class.

### Using MQSC commands on z/OS

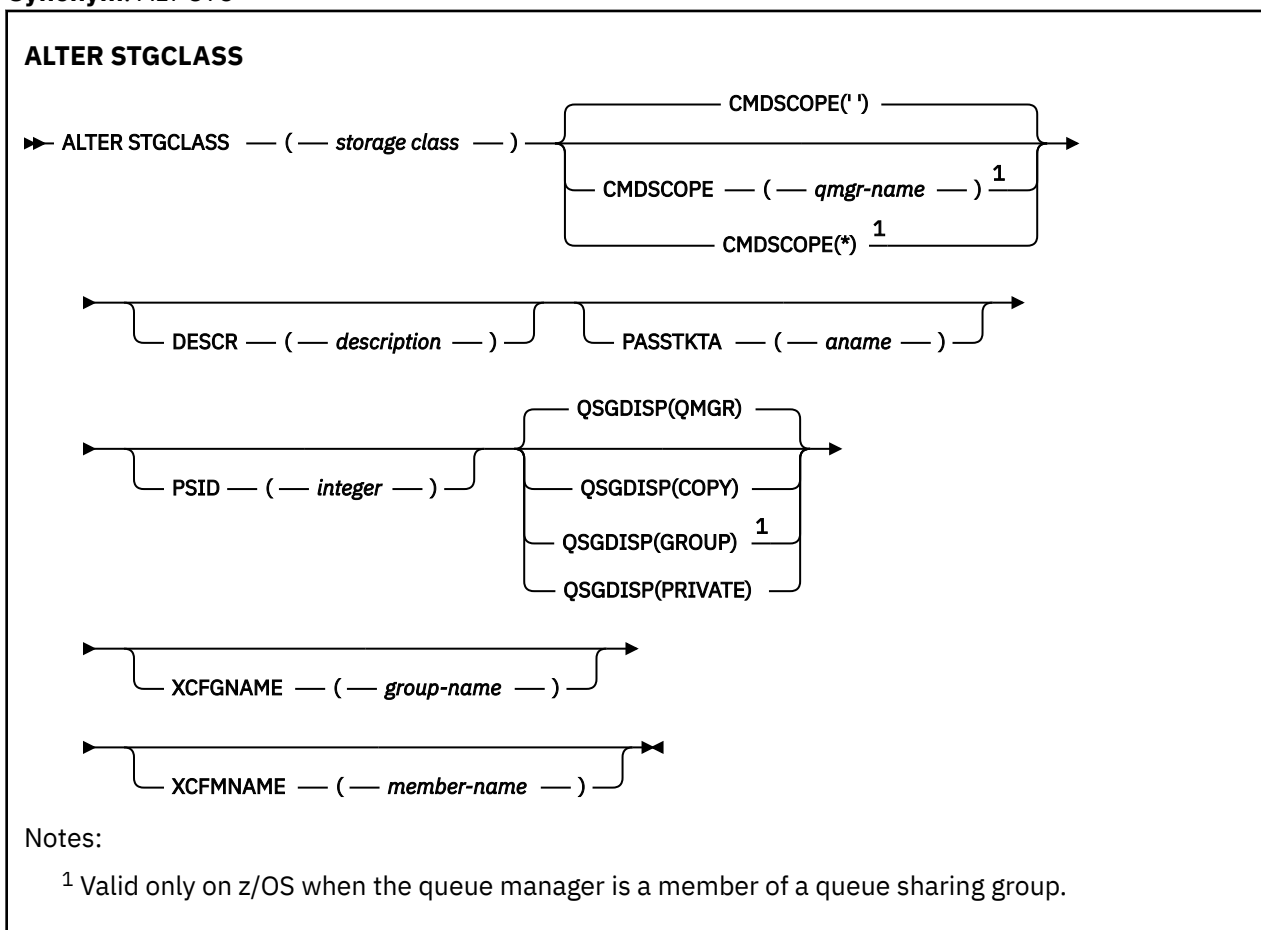
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Parameters not specified in the **ALTER STGCLASS** command result in the existing values for those parameters being left unchanged.

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for ALTER STGCLASS” on page 419](#)

**Synonym:** ALT STC



### Parameter descriptions for ALTER STGCLASS

#### (storage-class)

Name of the storage class.

This name is one to 8 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 through 9.

**Note:** Exceptionally, certain all numeric storage class names are allowed, but are reserved for the use of IBM service personnel.

The storage class must not be the same as any other storage class currently defined on this queue manager.

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered.

#### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

### DESCR(*description*)

Plain-text comment. It provides descriptive information about the object when an operator issues the **DISPLAY STGCLASS** command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager

### PASSTKTA(*application name*)

The application name that is passed to RACF when authenticating the PassTicket specified in the MQIIH header.

### PSID(*integer*)

The page set identifier that this storage class is to be associated with.

**Note:** No check is made that the page set has been defined; an error is raised only when you try to put a message to a queue that specifies this storage class (MQRC\_PAGESET\_ERROR).

The string consists of two numeric characters, in the range 00 through 99. See “[DEFINE PSID \(define page set and buffer pool\) on z/OS](#)” on page 545.

### QSGDISP

Specifies the disposition of the object in the group.

<i>Table 144. Behavior for each of the QSGDISP values</i>	
<b>QSGDISP</b>	<b>ALTER</b>
COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (COPY)</b> . Any object residing in the shared repository, or any object defined using a command that had the parameters <b>QSGDISP (QMGR)</b> , is not affected by this command.

Table 144. Behavior for each of the QSGDISP values (continued)

QSGDISP	ALTER
GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameters <b>QSGDISP (GROUP)</b>. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command. If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre data-bbox="561 474 1466 569">DEFINE STGCLASS(storage-class) REPLACE QSGDISP(COPY)</pre> <p>The ALTER for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
PRIVATE	<p>The object resides on the page set of the queue manager that executes the command, and was defined with <b>QSGDISP (QMGR)</b> or <b>QSGDISP (COPY)</b>. Any object residing in the shared repository is unaffected.</p>
QMGR	<p>The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP (QMGR)</b>. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.</p>

#### **XCFGNAME(group name)**

If you are using the IMS bridge, this name is the name of the XCF group to which the IMS system belongs. (This name is the group name specified in the IMS parameter list.)

This name is 1 - 8 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 - 9.

#### **XCFMNAME(member name)**

If you are using the IMS bridge, this name is the XCF member name of the IMS system within the XCF group specified in XCFGNAME. (This name is the member name specified in the IMS parameter list.)

This name is 1 - 16 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 - 9.


### **ALTER SUB (alter subscription settings)**

Use the MQSC command **ALTER SUB** to alter the characteristics of an existing subscription.

#### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

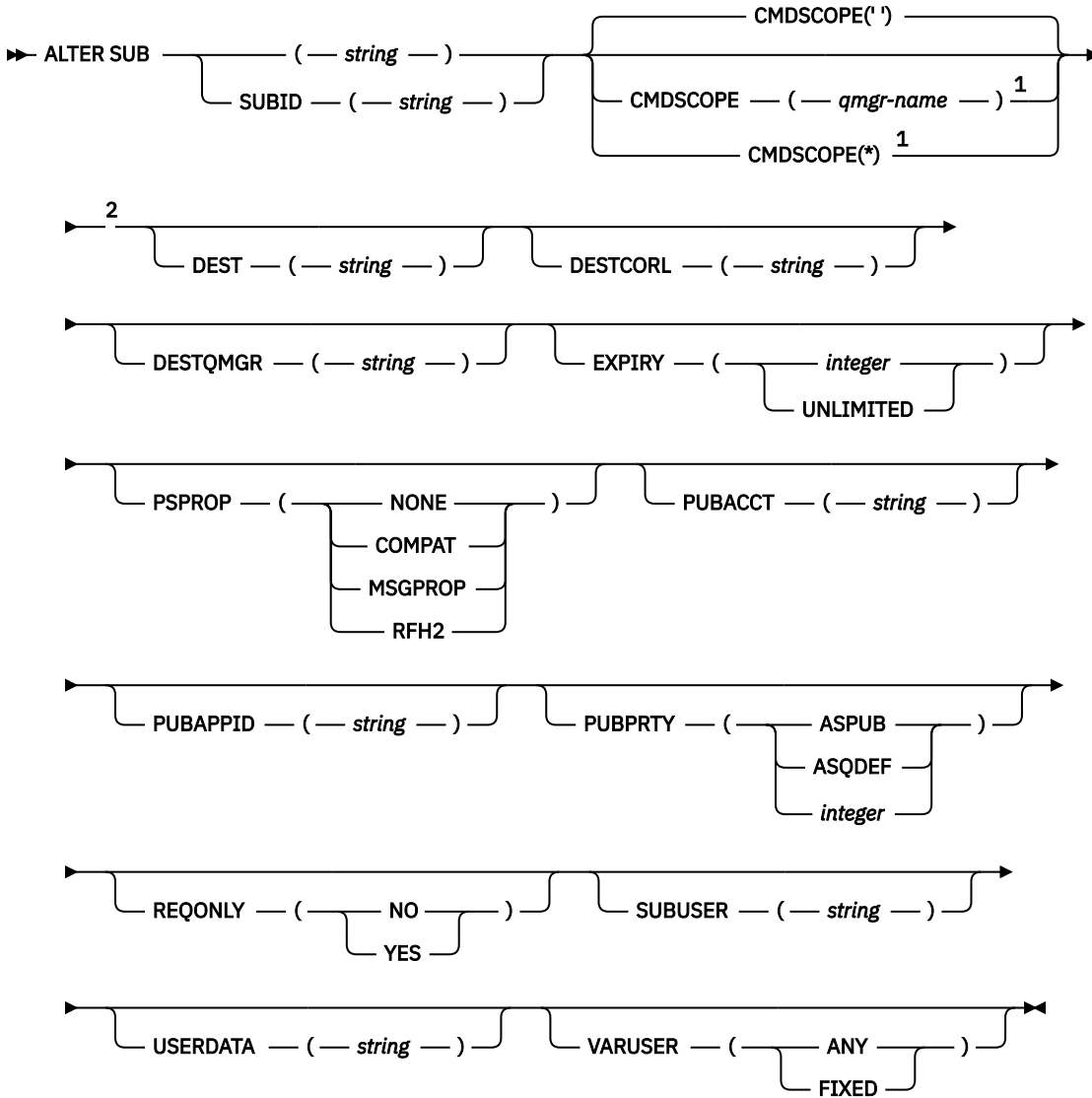
Parameters not specified in the **ALTER SUB** command result in the existing values for those parameters being left unchanged.

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for ALTER SUB” on page 422](#)
- [“Parameter descriptions for ALTER SUB” on page 423](#)

**Synonym: ALT SUB**

## ALTER SUB



### Notes:

<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

<sup>2</sup> Valid only on z/OS.

## Usage notes for ALTER SUB

- The following forms are valid for the command:

```

ALT SUB(xyz)
ALT SUB SUBID(123)
ALT SUB(xyz) SUBID(123)
  
```

- Although permitted on the **DEFINE** command, you cannot alter the following fields using **ALTER SUB**:

- **TOPICOBJ**
- **TOPICSTR**
- **WSHEMA**
- **SELECTOR**



**(integer)**

The time to expiry, in tenths of a second, from the creation date and time.

**UNLIMITED**

There is no expiry time. This is the default option supplied with the product.

**PSPROP**

The manner in which publish subscribe related message properties are added to messages sent to this subscription.

**NONE**

Do not add publish subscribe properties to the message.

**COMPAT**

Publish/subscribe properties are added within an MQRFH version 1 header unless the message was published in PCF format.

**MSGPROP**

Publish/subscribe properties are added as message properties.

**RFH2**

Publish/subscribe properties are added within an MQRFH version 2 header.

**PUBACCT(string)**

Accounting token passed by the subscriber, for propagation into messages published to this subscription in the AccountingToken field of the MQMD.

If this byte string is enclosed in quotation marks, characters in the range A-F must be specified in uppercase.

**PUBAPPID(string)**

Identity data passed by the subscriber, for propagation into messages published to this subscription in the ApplIdentityData field of the MQMD.

**PUBPRTY**

The priority of the message sent to this subscription.

**AS PUB**

Priority of the message sent to this subscription is taken from the priority supplied in the published message.

**AS QDEF**

Priority of the message sent to this subscription is taken from the default priority of the queue defined as a destination.

**(integer)**

An integer providing an explicit priority for messages published to this subscription.

**REQONLY**

Indicates whether the subscriber polls for updates using the MQSUBRQ API call, or whether all publications are delivered to this subscription.

**NO**

All publications on the topic are delivered to this subscription. This is the default value.

**YES**

Publications are only delivered to this subscription in response to an MQSUBRQ API call.

This parameter is equivalent to the subscribe option MQSO\_PUBLICATIONS\_ON\_REQUEST.

**SUBUSER(string)**

Specifies the user ID that is used for security checks that are performed to ensure that publications can be put to the destination queue associated with the subscription. This ID is either the user ID associated with the creator of the subscription or, if subscription takeover is permitted, the user ID that last took over the subscription. The length of this parameter must not exceed 12 characters.

**USERDATA(string)**

Specifies the user data associated with the subscription. The string is a variable length value that can be retrieved by the application on an MQSUB API call and passed in a message sent to this



subscription as a message property. The **USERDATA** is stored in the RFH2 header in the mqps folder with the key Sud.

An IBM MQ classes for JMS application can retrieve the subscription user data from the message by using the constant JMS\_IBM\_SUBSCRIPTION\_USER\_DATA. For more information, see [Retrieval of user subscription data](#).

#### **VARUSER**

Specifies whether a user other than the subscription creator can connect to and take over ownership of the subscription.

#### **ANY**

Any user can connect to and takeover ownership of the subscription.

#### **FIXED**

Takeover by another USERID is not permitted.

#### **Related tasks**

[Changing local subscription attributes](#)

### **ALTER TOPIC (alter topic settings)**


Use the MQSC **ALTER TOPIC** command to alter the parameters of an existing IBM MQ topic object.

#### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

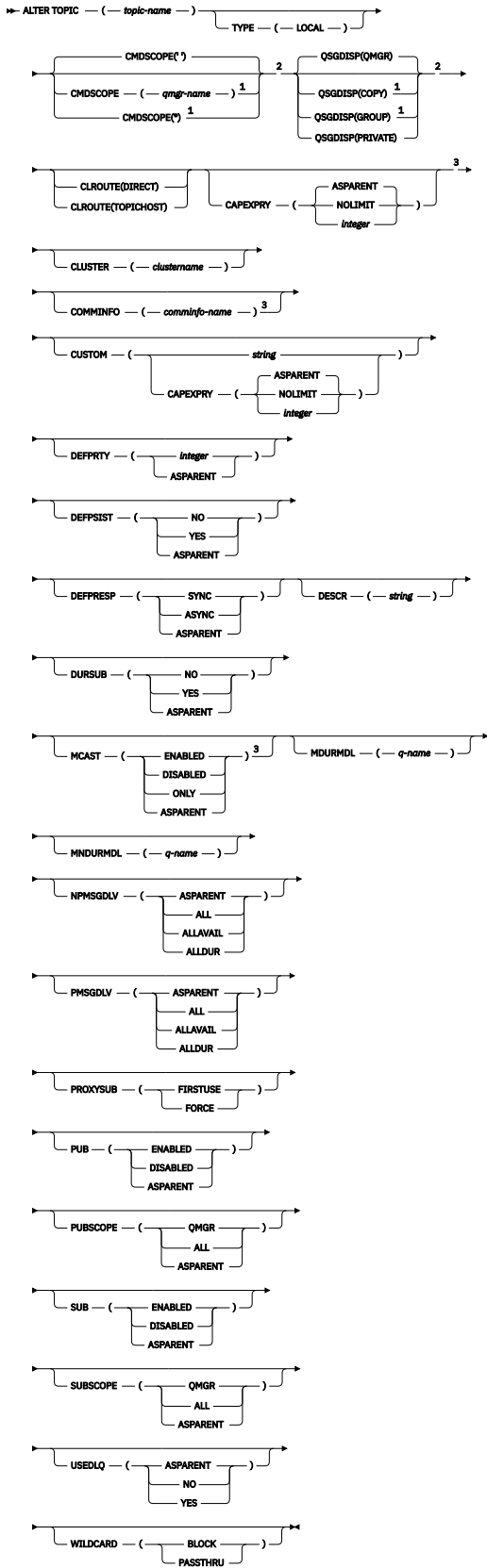
Parameters not specified in the **ALTER TOPIC** command result in the existing values for those parameters being left unchanged.

- [Syntax diagram](#)
- [“Usage notes for ALTER TOPIC” on page 427](#)
- [“Parameter descriptions for ALTER TOPIC” on page 427](#)

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

**Synonym:** ALT TOPIC

## ALTER TOPIC



### Notes:

<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

<sup>2</sup> Valid only on z/OS.

<sup>3</sup> Not valid on z/OS.

## Usage notes for ALTER TOPIC

- Successful completion of the command does not mean that the action completed. To check for true completion, see the [ALTER TOPIC](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for ALTER TOPIC

### *(topic-name)*

Name of the IBM MQ topic definition (see [Rules for naming IBM MQ objects](#)). The maximum length is 48 characters.

The name must not be the same as any other topic definition currently defined on this queue manager (unless REPLACE is specified).

**V 9.3.1**

**Multi**

### **CAEXPRY(*integer*)**

The maximum time, expressed in tenths of a second, until a message published to a topic which inherits properties from this object, remains in the system until it becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

### *integer*

The value must be in the range one through to 999 999 999.

### **NOLIMIT**

There is no limit on the expiry time of messages put to this topic.

### **ASPARENT**

The maximum message expiry time is based on the setting of the closest parent administrative topic object in the topic tree. This is the default value.

## **CLROUTE**

The routing behavior to use for topics in the cluster defined by the **CLUSTER** parameter.

### **DIRECT**

When you configure a direct routed clustered topic on a queue manager, all queue managers in the cluster become aware of all other queue managers in the cluster. When performing publish and subscribe operations, each queue manager can connect direct to any other queue manager in the cluster.

### **TOPICHOST**

When you use topic host routing, all queue managers in the cluster become aware of the cluster queue managers that host the routed topic definition (that is, the queue managers on which you have defined the topic object). When performing publish and subscribe operations, queue managers in the cluster connect only to these topic host queue managers, and not directly to each other. The topic host queue managers are responsible for routing publications from queue managers on which publications are published to queue managers with matching subscriptions.

After a topic object has been clustered (through setting the **CLUSTER** property) you cannot change the value of the **CLROUTE** property. The object must be un-clustered (**CLUSTER** set to ' ') before you can change the value. Un-clustering a topic converts the topic definition to a local topic, which results in a period during which publications are not delivered to subscriptions on remote queue managers; this should be considered when performing this change. See [The effect of defining a non-cluster topic with the same name as a cluster topic from another queue manager](#). If you try to change the value of the **CLROUTE** property while it is clustered, the system generates an MQRCCF\_CLROUTE\_NOT\_ALTERABLE exception.

See also [Routing for publish/subscribe clusters: Notes® on behavior](#) and [Designing publish/subscribe clusters](#).

## CLUSTER

The name of the cluster to which this topic belongs. Setting this parameter to a cluster that this queue manager is a member of makes all queue managers in the cluster aware of this topic. Any publication to this topic or a topic string below it put to any queue manager in the cluster is propagated to subscriptions on any other queue manager in the cluster. For more details, see [Distributed publish/subscribe networks](#).

..

If no topic object above this topic in the topic tree has set this parameter to a cluster name, then this topic does not belong to a cluster. Publications and subscriptions for this topic are not propagated to publish/subscribe cluster-connected queue managers. If a topic node higher in the topic tree has a cluster name set, publications and subscriptions to this topic are also propagated throughout the cluster.

### *string*

The topic belongs to this cluster. It is not recommended that this is set to a different cluster from a topic object above this topic object in the topic tree. Other queue managers in the cluster will honor this object's definition unless a local definition of the same name exists on those queue managers.

To prevent all subscriptions and publications being propagated throughout a cluster, leave this parameter blank on the system topics SYSTEM.BASE.TOPIC and SYSTEM.DEFAULT.TOPIC, except in special circumstances, for example to support migration.

## z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

## COMMINFO(*comminfo-name*)

The name of the communication information object associated with this topic object.

## CUSTOM(*string*)

The custom attribute for new features.

This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE).

## CAPEXPY(*integer*)

**Note:** **V9.3.1** The CAPEXPY queue attribute introduced at IBM MQ 9.3.1 replaces the use of the CAPEXPY option in the CUSTOM field. It is not possible to set the CAPEXPY attribute if the CUSTOM field has a CAPEXPY attribute defined in it already. You should alter existing topics

to set the new CAPEXPY field and unset the CAPEXPY attribute from the CUSTOM field. For example:

```
ALTER TOPIC(Q1) CAPEXPY(1000) CAPEXPY('')
```

The maximum time, expressed in tenths of a second, until a message published to a topic which inherits properties from this object, remains in the system until it becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

***integer***

The value must be in the range one through to 999 999 999.

**NOLIMIT**

There is no limit on the expiry time of messages put to this topic.

**ASPARENT**

The maximum message expiry time is based on the setting of the closest parent administrative topic object in the topic tree. This is the default value.

Specifying a value for CAPEXPY that is not valid, does not cause the command to fail. Instead, the default value is used.

**DEFPRTY(*integer*)**

The default priority of messages published to the topic.

**(*integer*)**

The value must be in the range zero (the lowest priority), through to the **MAXPRTY** queue manager parameter (**MAXPRTY** is 9).

**ASPARENT**

The default priority is based on the setting of the closest parent administrative topic object in the topic tree.

**DEFPSIST**

Specifies the message persistence to be used when applications specify the MQPER\_PERSISTENCE\_AS\_TOPIC\_DEF option.

**ASPARENT**

The default persistence is based on the setting of the closest parent administrative topic object in the topic tree.

**NO**

Messages on this queue are lost during a restart of the queue manager.

**YES**

Messages on this queue survive a restart of the queue manager.

On z/OS, N and Y are accepted as synonyms of NO and YES.

**DEFRESP**

Specifies the put response to be used when applications specify the MQPMO\_RESPONSE\_AS\_DEF option.

**ASPARENT**

The default put response is based on the setting of the closest parent administrative topic object in the topic tree.

**SYNC**

Put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application.

**ASYN**

Put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead. Some fields in the MQMD and

MQPMO are not returned by the queue manager to the application. However, an improvement in performance might be seen for messages put in a transaction and any non-persistent messages

#### **DESCR(string)**

Plain-text comment. It provides descriptive information about the object when an operator issues the **DISPLAY TOPIC** command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

#### **DURSUB**

Specifies whether applications are permitted to make durable subscriptions on this topic.

##### **ASPARENT**

Whether durable subscriptions can be made on this topic is based on the setting of the closest parent administrative topic object in the topic tree.

##### **NO**

Durable subscriptions cannot be made on this topic.

##### **YES**

Durable subscriptions can be made on this topic.

#### **MCAST**

Specifies whether multicast is allowable in the topic tree. The values are:

##### **ASPARENT**

The multicast attribute of the topic is inherited from the parent.

##### **DISABLED**

No multicast traffic is allowed at this node.

##### **ENABLED**

Multicast traffic is allowed at this node.

##### **ONLY**

Only subscriptions from a multicast capable client are allowed.

#### **MDURMDL(string)**

The name of the model queue to be used for durable subscriptions that request that the queue manager manages the destination of its publications (see [Rules for naming IBM MQ objects](#)). The maximum length is 48 characters.

If **MDURMDL** is blank, it operates in the same way as **ASPARENT** values on other attributes. The name of the model queue to be used is based on the closest parent administrative topic object in the topic tree with a value set for **MDURMDL**.

If you use **MDURMDL** to specify a model queue for a clustered topic, you must ensure that the queue is defined on every queue manager in the cluster where a durable subscription using this topic can be made.

The dynamic queue created from this model has a prefix of SYSTEM.MANAGED.DURABLE

#### **MNDURMDL(string)**

The name of the model queue to be used for non-durable subscriptions that request that the queue manager manages the destination of its publications (see [Rules for naming IBM MQ objects](#)). The maximum length is 48 characters.

If **MNDURMDL** is blank, it operates in the same way as **ASPARENT** values on other attributes. The name of the model queue to be used is based on the closest parent administrative topic object in the topic tree with a value set for **MNDURMDL**.

If you use **MNDURMDL** to specify a model queue for a clustered topic, you must ensure that the queue is defined on every queue manager in the cluster where a non-durable subscription using this topic can be made.

The dynamic queue created from this model has a prefix of SYSTEM.MANAGED.NDURABLE.

## **NPMSGDLV**

The delivery mechanism for non-persistent messages published to this topic:

### **ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **ALL**

Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

### **ALLAVAIL**

Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

### **ALLDUR**

Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a non-persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT calls fails.

## **PMSGDLV**

The delivery mechanism for persistent messages published to this topic:

### **ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **ALL**

Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

### **ALLAVAIL**

Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

### **ALLDUR**

Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT calls fails.

## **PROXYSUB**

Controls when a proxy subscription is sent for this topic, or topic strings below this topic, to neighboring queue managers when in a publish/subscribe cluster or hierarchy. For more details, see [Subscription performance in publish/subscribe networks](#).

### **FIRSTUSE**

For each unique topic string at or below this topic object, a proxy subscription is asynchronously sent to all neighboring queue managers when a local subscription is created or a proxy subscription is received that is propagated to further directly connected queue managers in a hierarchy.

### **FORCE**

A wildcard proxy subscription that matches all topic strings at and below this point in the topic tree is sent to neighboring queue managers even if no local subscriptions exist.

**Note:** The proxy subscription is sent when this value is set on **DEFINE** or **ALTER**. When set on a clustered topic, all queue managers in the cluster issue the wildcard proxy subscription to all other queue managers in the cluster.

**PUB**

Controls whether messages can be published to this topic.

**ASPARENT**

Whether messages can be published to the topic is based on the setting of the closest parent administrative topic object in the topic tree.

**ENABLED**

Messages can be published to the topic (by suitably authorized applications).

**DISABLED**

Messages cannot be published to the topic.

See also [Special handling for the \*\*PUB\*\* parameter.](#)

**PUBSCOPE**

Determines whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster.

**Note:** You can restrict the behavior on a publication-by-publication basis, using MQPMO\_SCOPE\_QMGR on the Put Message options.

**ASPARENT**

Whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster is based on the setting of the first parent administrative node found in the topic tree that relates to this topic.

**QMGR**

Publications for this topic are not propagated to connected queue managers.

**ALL**

Publications for this topic are propagated to hierarchically connected queue managers and to publish/subscribe cluster-connected queue managers.

 **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object within the group.

<i>Table 145. Behavior for each of the QSGDISP values</i>	
<b>QSGDISP</b>	<b>ALTER</b>
COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP(COPY)</b> . Any object residing in the shared repository, or any object defined using a command that had the parameters <b>QSGDISP(QMGR)</b> , is not affected by this command.
GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameters <b>QSGDISP(GROUP)</b>. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command. If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE TOPIC(name) REPLACE QSGDISP(COPY)</pre> <p>The ALTER for the group object takes effect regardless of whether the generated command with <b>QSGDISP(COPY)</b> fails.</p>



Table 145. Behavior for each of the QSGDISP values (continued)

QSGDISP	ALTER
PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with <b>QSGDISP(QMGR)</b> or <b>QSGDISP(COPY)</b> . Any object residing in the shared repository is unaffected.
QMGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters <b>QSGDISP(QMGR)</b> . Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

## SUB

Controls whether applications are to be permitted to subscribe to this topic.

### ASPARENT

Whether applications can subscribe to the topic is based on the setting of the closest parent administrative topic object in the topic tree.

### ENABLED

Subscriptions can be made to the topic (by suitably authorized applications).

### DISABLED

Applications cannot subscribe to the topic.

## SUBSCOPE

Determines whether this queue manager subscribes to publications in this queue manager or in the network of connected queue managers. If subscribing to all queue managers, the queue manager propagates subscriptions to them as part of a hierarchy or as part of a publish/subscribe cluster.

**Note:** You can restrict the behavior on a subscription-by-subscription basis, using **MQPMO\_SCOPE\_QMGR** on the Subscription Descriptor or **SUBSCOPE(QMGR)** on **DEFINE SUB**. Individual subscribers can override the **SUBSCOPE** setting of ALL by specifying the **MQSO\_SCOPE\_QMGR** subscription option when creating a subscription.

### ASPARENT

Whether this queue manager subscribes to publications in the same way as the setting of the first parent administrative node found in the topic tree relating to this topic.

### QMGR

Only publications that are published on this queue manager reach the subscriber.

### ALL

A publication made on this queue manager or on another queue manager reaches the subscriber. Subscriptions for this topic are propagated to hierarchically connected queue managers and to publish/subscribe cluster-connected queue managers.


## TOPICSTR( *string* )

The topic string represented by this topic object definition. This parameter is required and cannot contain the empty string.

The topic string must not be the same as any other topic string already represented by a topic object definition.

The maximum length of the string is 10,240 characters.

## TYPE (topic-type)

If this parameter is used it must follow immediately after the *topic-name* parameter on all platforms  except z/OS.

### LOCAL

A local topic object.

**USEDLQ**

Determines whether the dead-letter queue is used when publication messages cannot be delivered to their correct subscriber queue.

**ASPARENT**

Determines whether to use the dead-letter queue using the setting of the closest administrative topic object in the topic tree.

**NO**

Publication messages that cannot be delivered to their correct subscriber queue are treated as a failure to put the message. The MQPUT of an application to a topic fails in accordance with the settings of NPMGDLV and PMSGDLV.

**YES**

When the DEADQ queue manager attribute provides the name of a dead-letter queue, then it is used. If the queue manager does not provide the name of a dead-letter queue, then the behavior is as for NO.

**WILDCARD**

The behavior of wildcard subscriptions with respect to this topic.

**PASSTHRU**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object receive publications made to this topic and to topic strings more specific than this topic.

**BLOCK**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object do not receive publications made to this topic or to topic strings more specific than this topic.

The value of this attribute is used when subscriptions are defined. If you alter this attribute, the set of topics covered by existing subscriptions is not affected by the modification. This scenario applies also if the topology is changed when topic objects are created or deleted; the set of topics matching subscriptions created following the modification of the WILDCARD attribute is created using the modified topology. If you want to force the matching set of topics to be re-evaluated for existing subscriptions, you must restart the queue manager.

**Related tasks**

[Changing administrative topic attributes](#)

 **ALTER TRACE (alter trace event settings) on z/OS**

Use the MQSC command ALTER TRACE to change the trace events being traced for a particular active queue manager trace. ALTER TRACE stops the specified trace, and restarts it with the altered parameters.

**Using MQSC commands on z/OS**

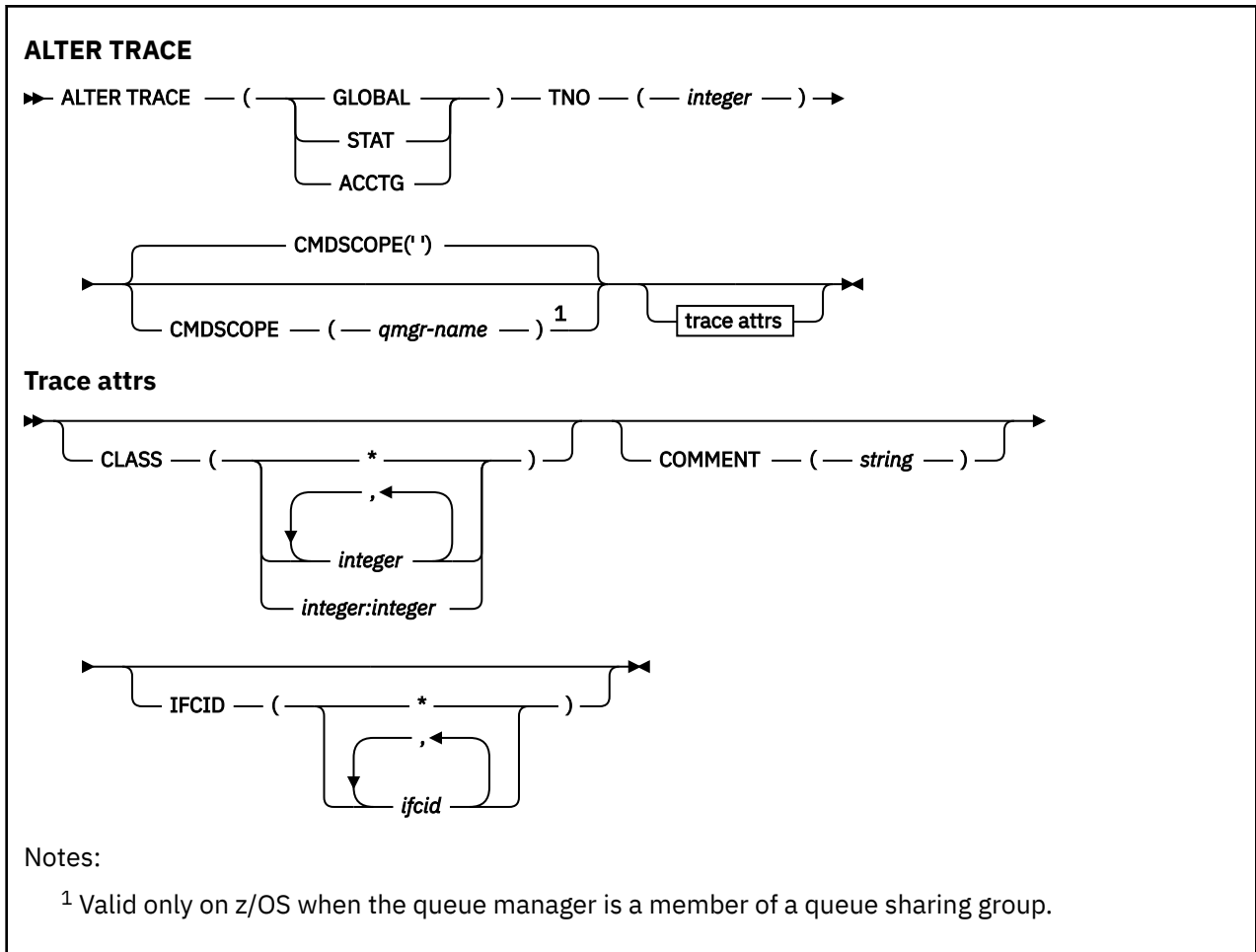
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Parameters not specified in the ALTER TRACE command result in the existing values for those parameters being left unchanged.

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 435](#)
- [“Parameter descriptions for ALTER TRACE” on page 435](#)
- [“Trace parameters” on page 436](#)

**Synonym:** ALT TRACE



## Usage notes

Channel initiator traces cannot be altered.

## Parameter descriptions for ALTER TRACE

Specify one of the following trace types:

### GLOBAL

Service data from the entire queue manager (the synonym is G)

### STAT

Statistical data (the synonym is S)

### ACCTG

Accounting data (the synonym is A)

And:

### TNO(*integer*)

The number of the trace to be altered (1 through 32). You can specify only one trace number.

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

''

The command runs on the queue manager on which it was entered.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

**Trace parameters****CLASS( integer )**

The new trace class. See [“START TRACE \(start trace\) on z/OS”](#) on page 957 for a list of allowed classes. A range of classes can be specified as *m:n* (for example, CLASS(01:03)).

For GLOBAL and CHINIT traces, CLASS(\*) activates all classes.

For ACCTG and STAT traces, CLASS(\*) activates classes 1 to 3. Channel initiator statistics and channel accounting data are not started with CLASS(\*), and must be started with CLASS(4).

**COMMENT( string )**

A comment that is reproduced in the trace output record (except in the resident trace tables).

*string* is any character string. If it includes blanks, commas, or special characters, it must be enclosed between single quotation marks (').

**IFCID( ifcid )**

Reserved for IBM Service.

## ARCHIVE LOG (back up the active log) on z/OS

Use the MQSC command ARCHIVE LOG as part of your backup procedure. It takes a copy of the current active log (or both logs if you are using dual logging).

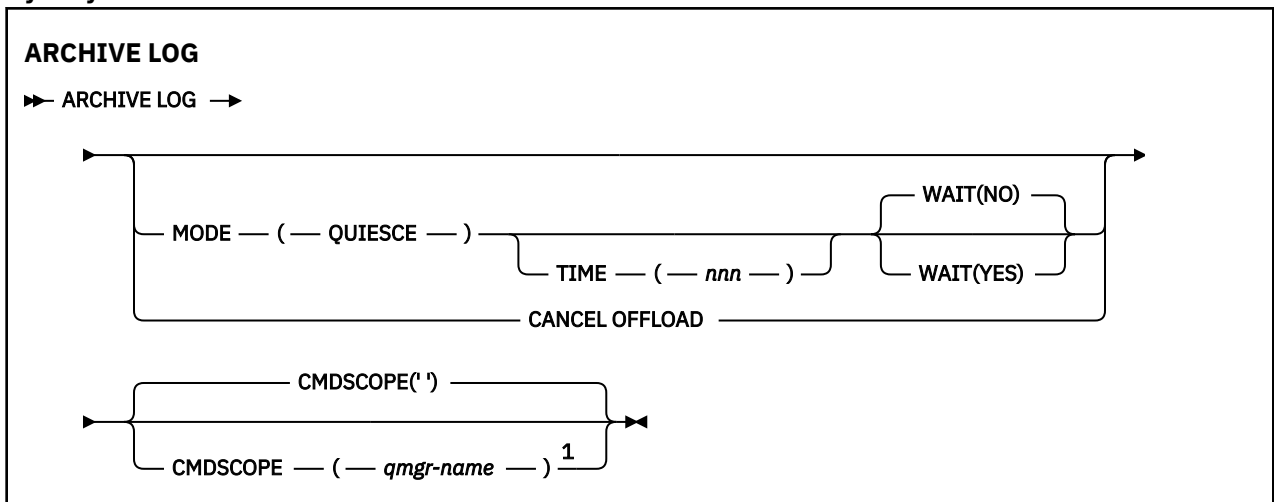
**Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for ARCHIVE LOG”](#) on page 437
- [“Parameter descriptions for ARCHIVE LOG”](#) on page 437

**Synonym:** ARC LOG



Notes:

<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

## Usage notes for ARCHIVE LOG

**ARCHIVE LOG** performs the following actions:

1. Truncates the current active log data sets.
2. Continues logging, switching to the next active log data set.
3. Starts a task to offload the data sets.
4. Archives previous active log data sets not yet archived.

If the **MODE(QUIESCE)** parameter is used, the **ARCHIVE LOG** command quiesces (suspends) all user update activity on the current active log before the offload process. Once a system-wide point of consistency is reached (that is, when all currently active update users have reached a commit point), the current active log data set is immediately truncated, and the offload process is initiated. The resulting point of consistency is captured in the current active log before it is offloaded.

Normally, control returns to the user immediately, and the quiescing is done asynchronously. However, if the **WAIT(YES)** parameter is used, the quiescing is done synchronously, and control does not return to the user until it has finished.

- You cannot issue an **ARCHIVE LOG** command while a previous **ARCHIVE LOG** command is in progress.
- You cannot issue an **ARCHIVE LOG** command when the active log data set is the last available active log data set, because it would use all the available active log data set space, and IBM MQ would halt all processing until an offload had been completed.
- You can issue an **ARCHIVE LOG** command without the **MODE(QUIESCE)** option when a **STOP QMGR MODE(QUIESCE)** is in progress, but not when a **STOP QMGR MODE(FORCE)** is in progress.
- You can issue a **DISPLAY LOG** command to discover whether an **ARCHIVE LOG** command is active. If an **ARCHIVE LOG** command is active, the **DISPLAY** command returns message CSQV400I.
- You can issue an **ARCHIVE LOG** command even if archiving is not being used (that is, **OFFLOAD** is set to NO in the CSQ6LOGP system parameter macro), or dynamically using the **SET LOG** command. In this case, the current active log data sets are truncated and logging continues using the next active log data set, but there is no offloading to archive data sets.

## Parameter descriptions for ARCHIVE LOG

All the parameters are optional. If none are specified, the current active log data sets are switched and offloaded immediately.

### CANCEL OFFLOAD

Cancels any offloading currently in progress and restarts the offload process. The process starts with the oldest active log data set and proceeds through all the active data sets that need offloading.

Use this command only if the offload task does not appear to be working, or if you want to restart a previous offload attempt that failed.

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

### **MODE(QUIESCE)**

Stops any new update activity on the queue manager, and brings all existing users to a point of consistency after a commit. When this state is reached, or the number of active users is zero, the current active log is archived.

The time that the queue manager waits to reach such a state is limited to the value specified by **QUIESCE** in the CSQ6ARVP system parameter macro. The value of **QUIESCE** can be overridden by the **TIME** parameter of this command. If activity has not quiesced in that time, the command fails; no offload is done, and logging continues with the current active log data set.

### **TIME( *nnn* )**

Overrides the quiesce time period specified by the **QUIESCE** value of the CSQ6ARVP system parameter macro.

*nnn* is the time, in seconds, in the range 001 through 999.

To specify the TIME parameter, you must also specify MODE(QUIESCE).

If you specify the TIME parameter, you must specify an appropriate value for the quiesce period. If you make the period too short or too long, one of the following problems might occur:

- The quiesce might not be complete
- IBM MQ lock contention might develop
- A timeout might interrupt the quiesce

### **WAIT**

Specifies whether IBM MQ is to wait until the quiesce process has finished before returning to the issuer of the **ARCHIVE LOG** command.

To specify the **WAIT** parameter, you must also specify **MODE(QUIESCE)**.

### **NO**

Specifies that control is returned to the issuer when the quiesce process starts. (The synonym is N.) This makes the quiesce process asynchronous to the issuer; you can issue further MQSC commands when the **ARCHIVE LOG** command returns control to you. This is the default.

### **YES**

Specifies that control is returned to the issuer when the quiesce process finishes. (The synonym is Y.) This makes the quiesce process synchronous to the issuer; further MQSC commands are not processed until the **ARCHIVE LOG** command finishes.

### **Related tasks**

[Archiving logs with the ARCHIVE LOG command](#)

## **BACKUP CFSTRUCT (back up a CF application structure) on z/OS**

Use the MQSC command BACKUP CFSTRUCT to initiate a CF application structure backup.

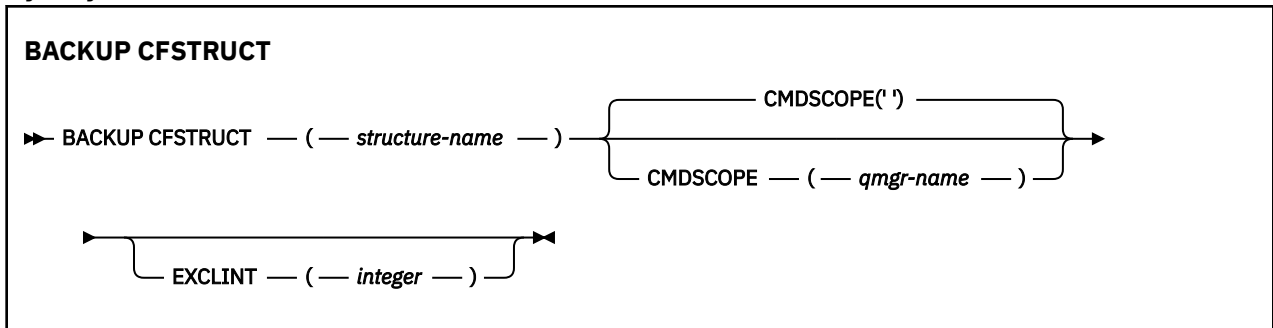
### **Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for BACKUP CFSTRUCT” on page 439](#)
- [“Keyword and parameter descriptions for BACKUP CFSTRUCT” on page 439](#)

**Synonym:** None



## Usage notes for BACKUP CFSTRUCT

1. This command is valid only on z/OS when the queue manager is a member of a queue sharing group.
2. Only persistent shared queue messages are backed up. Non-persistent messages are not backed up and cannot be recovered
3. You can concurrently run separate backups for different application structures on different queue managers within the queue sharing group. You can also concurrently run separate backups for different application structures on the same queue manager.
4. This command fails if the specified CF structure is defined with either a CFLEVEL less than 3, or with RECOVER set to NO.
5. The command fails if a specified application structure is currently in the process of being backed up by another queue manager within the queue sharing group.

## Keyword and parameter descriptions for BACKUP CFSTRUCT

### **structure-name**

The name of the coupling facility (CF) application structure to be backed up. An asterisk (\*) on its own specifies all recoverable CF structures. A trailing asterisk (\*) matches all recoverable structure names with the specified stem followed by zero or more characters. The value (CSQ\*) matches all recoverable CF structures with the specified stem (CSQ) followed by zero or more characters.

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and the command server is enabled.

### **EXCLINT( integer )**

Specifies a value that defines a number of seconds that are used as an exclusion time. The backup excludes backing-up activity during this exclusion time. The exclusion time starts immediately before the back up starts. For example, if EXCLINT(30) is specified, the backup does not include the last 30 seconds worth of activity for this application-structure before back up started.

The value must be in the range 30 through 600. The default value is 30.

## CLEAR QLOCAL (clear messages from local queue)

Use the MQSC command CLEAR QLOCAL to clear the messages from a local queue.

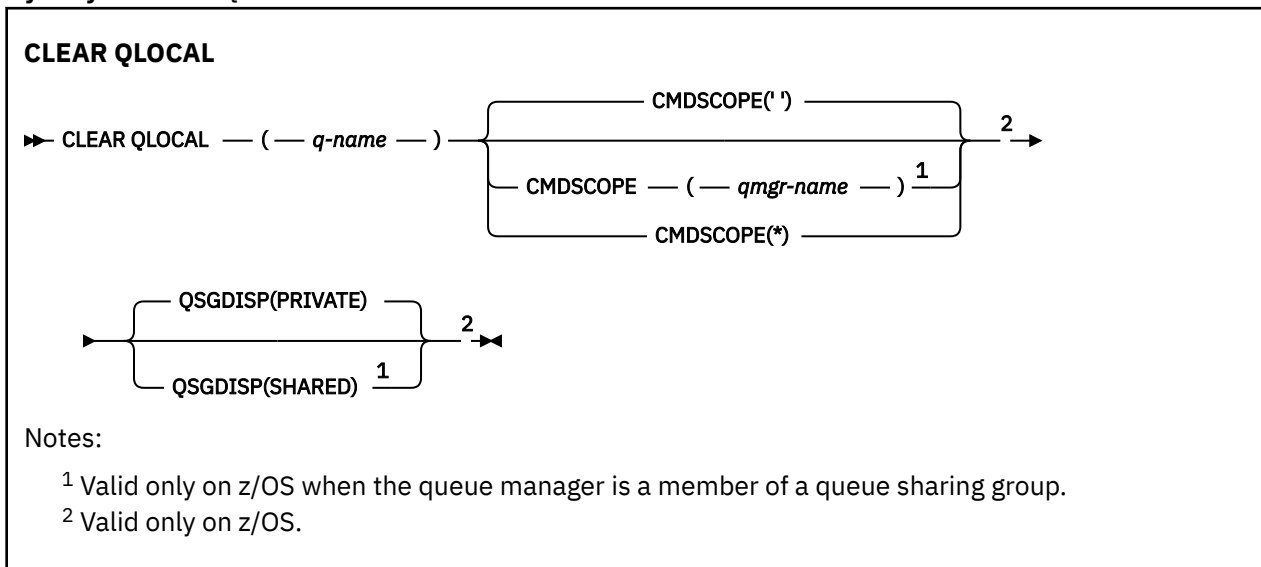
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for CLEAR QLOCAL” on page 440](#)

**Synonym:** CLEAR QL



### Parameter descriptions for CLEAR QLOCAL

You must specify which local queue you want to clear.

The command fails if any one of the following occurs. The queue:

- Has uncommitted messages that have been put on the queue under syncpoint.
- Is currently open by an application (with any open options).
- Is currently open by a channel.

For performance reasons, receiver channels cache opened queue handles for queues to which they deliver messages on behalf of remote queue managers. If a channel has cached a handle to this queue, it cannot be cleared.

To resolve this, stop the channel. Alternatively, empty the queue by getting all the messages from it, instead of using the CLEAR QLOCAL command.

If an application has this queue open, or has a queue open that eventually resolves to this queue, the command fails. The command also fails if this queue is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open.

#### (q-name)

The name of the local queue to be cleared. The name must be defined to the local queue manager.

**z/OS** **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.



CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to SHARED.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### **z/OS QSGDISP**

Specifies whether the queue definition is shared. This parameter applies to z/OS only.

#### **PRIVATE**

Clear only the private queue named *q-name*. The queue is private if it was defined using a command that had the parameters QSGDISP(COPY) or QSGDISP(QMGR). This is the default value.

#### **SHARED**

Clear only the shared queue named *q-name*. The queue is shared if it was defined using a command that had the parameters QSGDISP(SHARED).

#### **Related tasks**

[Clearing a local queue](#)

## **CLEAR TOPICSTR (clear topic string)**

Use the MQSC command CLEAR TOPICSTR to clear the retained message which is stored for the specified topic string.

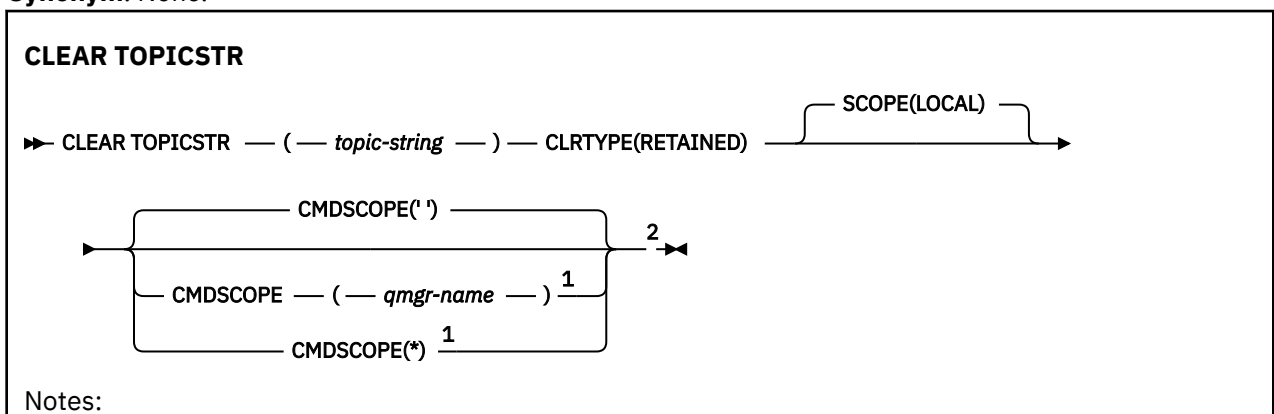
### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [Usage notes for CLEAR TOPICSTR](#)
- [Parameter descriptions for CLEAR TOPICSTR](#)

**Synonym:** None.



<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

<sup>2</sup> Valid only on z/OS.

## Usage notes for CLEAR TOPICSTR

1. If the topic string specified has no retained message the command will complete successfully. You can find out whether a topic string has a retained message by using the DISPLAY TPSTATUS command. The RETAINED field shows whether there is a retained message.
2. The topic-string input parameter on this command must match the topic you want to act on. You are advised to keep the character strings in your topic strings as characters that can be used from location issuing the command. If you issue commands using MQSC, you will have fewer characters available to you than if you are using an application submitting PCF messages, such as the IBM MQ Explorer.
3. You might need to use CLEAR TOPICSTR to remove a retained publication from a publish/subscribe cluster. For example:
  - If you accidentally configure a retained publication, and then need to remove it from all cluster queue managers, you issue this command on all members of the cluster.
  - In a direct routed publish/subscribe cluster, if you move a publishing application to a new queue manager and the previous queue manager holds no subscriptions for the affected topic string, you need to ensure that the previous queue manager does not resend the old retained publication to other members of the cluster. To do this, wait until the application has published on the new queue manager, then issue this command on the previous queue manager to remove the retained publication held there.

See also [Design considerations for retained publications in publish/subscribe clusters](#)

## Parameter descriptions for CLEAR TOPICSTR

You must specify which topic string you want to remove the retained publication from.

### **(topic-string)**

The topic string to be cleared. This string can represent several topics to be cleared by using wildcards as shown in the following table:

Special Character	Behavior
#	Wildcard, multiple topic level
+	Wildcard, single topic level

**Note:** the '+' and '#' are not treated as wildcards if they are mixed in with other characters (including themselves) within a topic level. In the following string, the '#' and '+' characters are treated as ordinary characters.

```
level0/level1/#+/level3/level#
```

To illustrate the effect of wildcards, the following example is used.

Clearing the following topic:

```
/a/b/#/z
```

clears the following topics:

```
/a/b/z  
/a/b/c/z  
/a/b/c/y/z
```

## CLRTYPE

This is a mandatory parameter.

The value must be:

## RETAINED

Remove the retained publication from the specified topic string.

## **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the name of the local queue manager, if the shared queue object definition has its queue sharing group disposition attribute QSGDISP set to SHARED.

..

The command runs on the queue manager on which it was entered. This is the default value.

### ***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## SCOPE

The scope of the deletion of retained messages.

The value can be:

## LOCAL


The retained message is removed from the specified topic string at the local queue manager only. This is the default value.

## DEFINE AUTHINFO (define an authentication information object)

Use the MQSC command **DEFINE AUTHINFO** to define an authentication information object. These objects contain the definitions required to perform certificate revocation checking using OCSP or Certificate Revocation Lists (CRLs) on LDAP servers, and the definitions required to check authentication credentials provided by applications.

## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

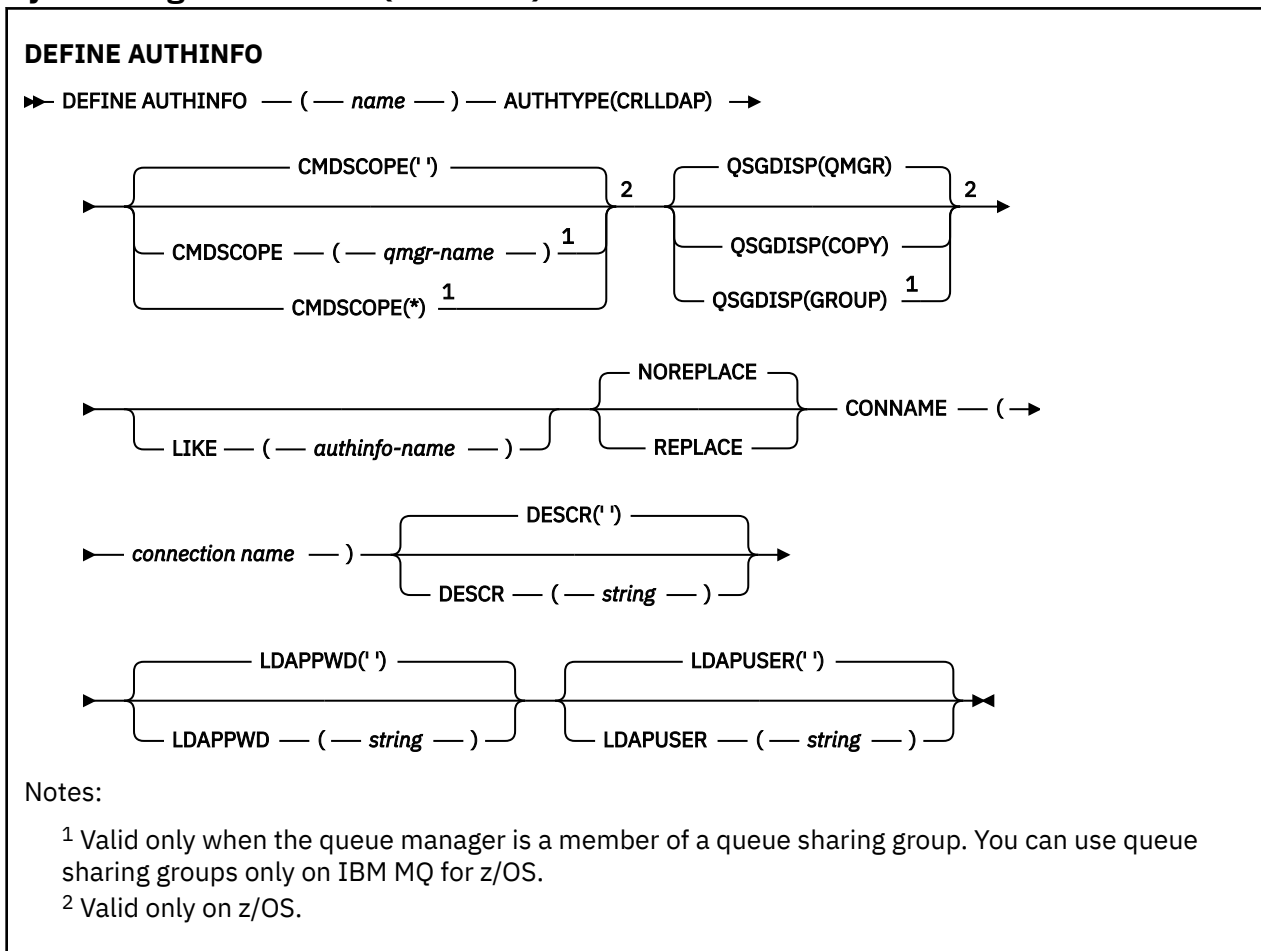
- [Syntax diagram for TYPE\(CRLLDAP\)](#)
- [Syntax diagram for TYPE\(OCSP\)](#)
- [Syntax diagram for TYPE\(IDPWOS\)](#)
- [Syntax diagram for TYPE\(IDPWLDAP\)](#)

- “Usage notes for DEFINE AUTHINFO” on page 448
- “Parameter descriptions for DEFINE AUTHINFO” on page 448

**Synonym:** DEF AUTHINFO

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).

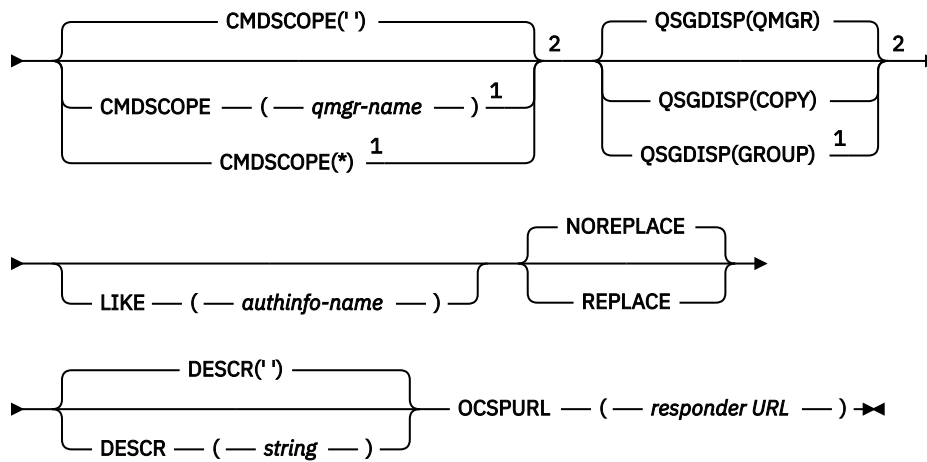
### Syntax diagram for TYPE(CRLLDAP)



## Syntax diagram for TYPE(OCSP)

### DEFINE AUTHINFO

► DEFINE AUTHINFO — ( — *name* — ) — AUTHTYPE(OCSP) —►



#### Notes:

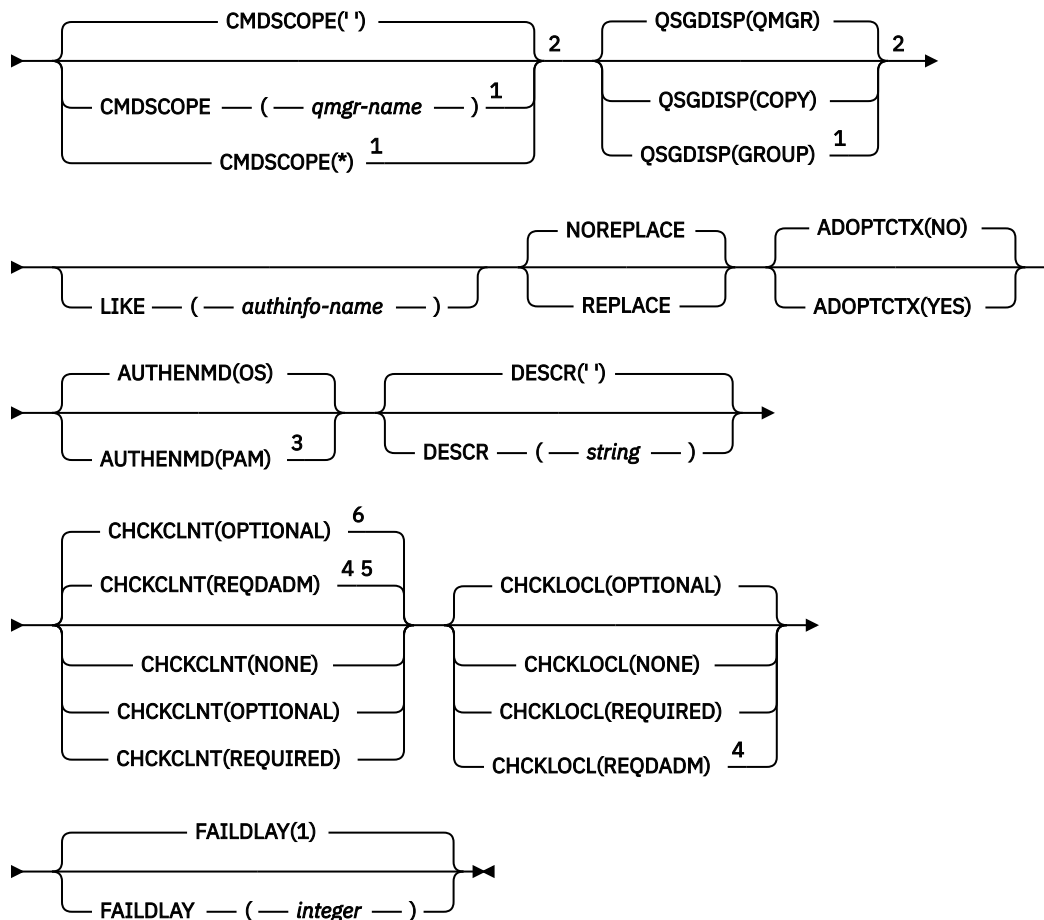
<sup>1</sup> Valid only when the queue manager is a member of a queue sharing group. You can use queue sharing groups only on IBM MQ for z/OS.

<sup>2</sup> Valid only on z/OS.

## Syntax diagram for TYPE(IDPWOS)

### DEFINE AUTHINFO

► DEFINE AUTHINFO — ( — *name* — ) — AUTHTYPE(IDPWOS) —►



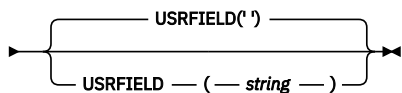
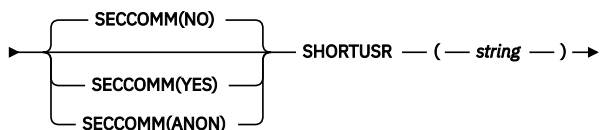
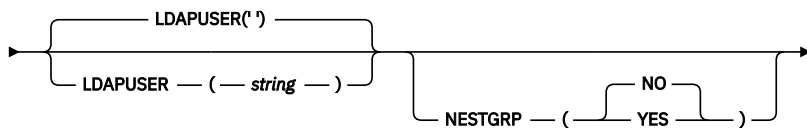
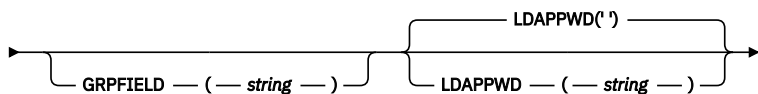
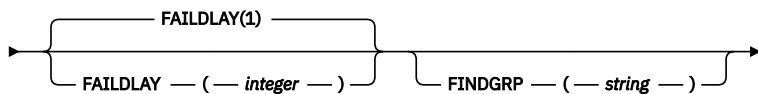
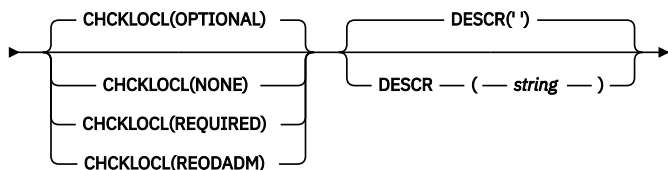
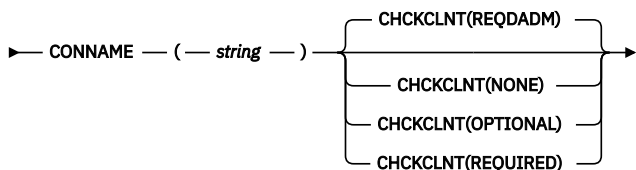
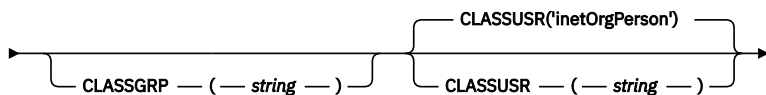
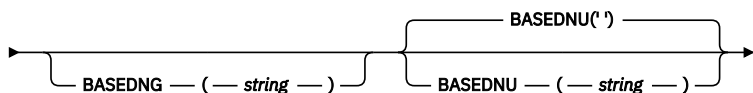
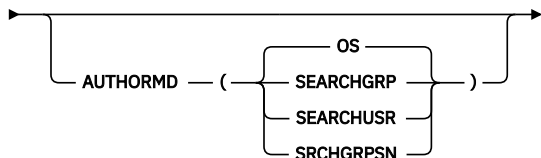
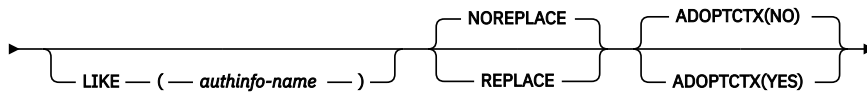
#### Notes:

- <sup>1</sup> Valid only when the queue manager is a member of a queue sharing group. You can use queue sharing groups only on IBM MQ for z/OS.
- <sup>2</sup> Valid only on z/OS.
- <sup>3</sup> Not valid on z/OS and PAM value can be set only on AIX and Linux.
- <sup>4</sup> Not valid on IBM MQ for z/OS.
- <sup>5</sup> Default for platforms other than z/OS.
- <sup>6</sup> Default for z/OS.

# Syntax diagram for TYPE(IDPWLDAP)

## DEFINE AUTHINFO

► DEFINE AUTHINFO — ( — name — ) — AUTHTYPE(IDPWLDAP) —<sup>1</sup>→



Notes:

<sup>1</sup> Not valid on IBM MQ for z/OS.

## Usage notes for DEFINE AUTHINFO

**IBM i** On IBM i, authentication information objects of AUTHTYPE(CRLLDAP) and AUTHTYPE(OCSP) are only used for channels of type CLNTCONN through use of the AMQCLCHL.TAB. Certificates are defined by Digital Certificate Manager for each certificate authority, and are verified against the LDAP servers.



**Attention:** After running the DEFINE AUTHINFO command, you must restart the queue manager. If you do not restart the queue manager, the [setmqaut](#) command does not return the correct result.

## Parameter descriptions for DEFINE AUTHINFO

### *name*

Name of the authentication information object. This parameter is required.

The name must not be the same as any other authentication information object name currently defined on this queue manager (unless **REPLACE** or **ALTER** is specified). See [Rules for naming IBM MQ objects](#).

### **ADOPTCTX**

Whether to use the presented credentials as the context for this application. This means that they are used for authorization checks, shown on administrative displays, and appear in messages.

#### **YES**

The user ID in the authentication credentials presented in the MQCSP structure, which have been successfully validated, is adopted as the context to use for this application. Therefore, this user ID has the credentials checked for authorization to use IBM MQ resources.

If the application presents a user ID and password, the user ID in the MQCSP structure is adopted if the password is successfully validated.

**Linux** **V 9.3.4** **AIX** If the application presents an authentication token, and the token is successfully validated, the user ID in the token user claim is adopted as the context for the application. The name of the token user claim is specified by the **UserClaim** attribute in the **AuthToken** stanza of the `qm.ini` file. For more information about the **UserClaim** attribute, see [UserClaim](#).

If the user ID presented is an LDAP user ID, and authorization checks are done using operating system user IDs, the [SHORTUSR](#) associated with the user entry in LDAP will be adopted as the credentials for authorization checks to be done against.

**ADOPTCTX(YES)** has an effect only if **CHKCLNT** or **CHKLOCL** is set to a value that causes the credentials to be validated.

#### **NO**

Authentication is performed on the credentials presented in the MQCSP structure, but then the credentials are not adopted for further use. Authorization is performed using the user ID that the application is running under.

The **ADOPTCTX** attribute is only valid for an **AUTHTYPE** of IDPWOS and IDPWLDAP.

### **AUTHENMD**

Authentication method. Whether to use the operating system or Pluggable Authentication Method (PAM) to authenticate user passwords.

**Linux** **AIX** **OS**

Use the traditional UNIX password verification method.

**Linux** **AIX** **PAM**

Use the PAM to authenticate the user password.

You can set the PAM value only on AIX and Linux.



Changes to this attribute are effective only after you run the `REFRESH SECURITY TYPE(CONNAUTH)` command.

This attribute is valid only for an **AUTHTYPE** of IDPWOS.

## AUTHORMD

Authorization Method.

### OS

Use operating system groups to determine permissions associated with a user.

This is how IBM MQ has previously worked, and is the default value.

### SEARCHGRP

A group entry in the LDAP repository contains an attribute listing the Distinguished Name of all the users belonging to that group. Membership is indicated by the attribute defined in `FINDGRP`. This value is typically *member* or *uniqueMember*.

### SEARCHUSR

A user entry in the LDAP repository contains an attribute listing the Distinguished Name of all the groups to which the specified user belongs. The attribute to query is defined by the `FINDGRP` value, typically *memberOf*.

### SRCHGRPSN

A group entry in the LDAP repository contains an attribute listing the short user name of all the users belonging to that group. The attribute in the user record that contains the short user name is specified by `SHORTUSR`.

Membership is indicated by the attribute defined in `FINDGRP`. This value is typically *memberUid*.

**Note:** This authorization method should only be used if all user short names are distinct.

Many LDAP servers use an attribute of the group object to determine group membership and you should, therefore, set this value to `SEARCHGRP`.

Microsoft Active Directory typically stores group memberships as a user attribute. The IBM Tivoli Directory Server supports both methods.

In general, retrieving memberships through a user attribute will be faster than searching for groups that list the user as a member.

## AUTHTYPE

The type of authentication information.


### CRLLDAP

Certificate Revocation List checking is done using LDAP servers.

### IDPWLDAP




Connection authentication user ID and password checking is done using an LDAP server.



**Attention:**  This option is not available on IBM MQ for z/OS

### IDPWOS

Connection authentication user ID and password checking is done using the operating system.


   Authentication tokens supplied by IBM MQ MQI clients are validated if the queue manager is configured to accept authentication tokens using the **AuthToken** stanza of the `qm.ini` file. For more information about the **AuthToken** stanza, see `AuthToken` stanza of the `qm.ini` file.

### OCSP

Certificate revocation checking is done using OCSP.

An authentication information object with **AUTHTYPE (OCSP)** does not apply for use on queue managers on the following platforms:

-  IBM i

-  z/OS

However, it can be specified on those platforms to be copied to the client channel definition table (CCDT) for client use.

This parameter is required.

You cannot define an authentication information object as LIKE one with a different **AUTHTYPE**. You cannot alter the **AUTHTYPE** of an authentication information object after you have created it.

### BASEDNG

Base DN for groups.

In order to be able to find group names, this parameter must be set with the base DN to search for groups in the LDAP server.

### BASEDNU(*base DN*)

In order to be able to find the short user name attribute (see [SHORTUSR](#) ) this parameter must be set with the base DN to search for users within the LDAP server.

This attribute is valid only for an **AUTHTYPE** of IDPWLDAP.

### CHCKCLNT

This attribute determines the authentication requirements for client applications, and is valid only for an **AUTHTYPE** of IDPWOS or IDPWLDAP. The possible values are:


#### NONE

Authentication credentials supplied by client applications are not checked. If a user ID and password, or an authentication token, is supplied by a client application, the credentials are ignored. **ADOPTCTX** will have no effect and any user IDs contained within the MQCSP will not be used for authorization checks later.

#### OPTIONAL

Client applications are not required to provide authentication credentials.

Any applications that do provide a user ID and password in the MQCSP structure have them authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection is only allowed to continue if the user ID and password are valid.


 If an application provides an authentication token, and the queue manager is configured to accept authentication tokens, the token is validated. The connection is only allowed to continue if the token is issued by a trusted issuer.

This option might be useful during migration, for example.

#### REQUIRED

All client applications must provide authentication credentials in the MQCSP structure.

If an application provides a user ID and password, these credentials are authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection is only allowed to continue if the user ID and password are valid.

 If an application provides an authentication token, and the queue manager is configured to accept authentication tokens, the token is validated. The connection is only allowed to continue if the token is issued by a trusted issuer.

If an application does not provide any authentication credentials, the connection is rejected.

#### REQDADM

All client applications using a privileged user ID must provide authentication credentials in the MQCSP structure. Any client applications using a non-privileged user ID are not required to provide authentication credentials and are treated as with the OPTIONAL setting.

A privileged user is one that has full administrative authorities for IBM MQ. See [Privileged users](#) for more information.

Any provided user ID and password are authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection is only allowed to continue if the user ID and password are valid.

Linux V 9.3.4 AIX If an application provides an authentication token, and the queue manager is configured to accept authentication tokens, the token is validated. The connection is only allowed to continue if the token is issued by a trusted issuer.

**Note:** The REQDADM value for the **CHCKCLNT** attribute is irrelevant if the authentication type is LDAP. This is because there is no concept of privileged user ID when using LDAP user accounts. LDAP user accounts and groups must be assigned permission explicitly.

z/OS This setting is not allowed on z/OS systems.

### Important:

1. This attribute can be overridden by the **CHCKCLNT** attribute of the CHLAUTH rule that matches the client connection. The **CONNAUTH AUTHINFO CHCKCLNT** attribute on the queue manager therefore determines the default client checking behavior for client connections that do not match a CHLAUTH rule, or where the CHLAUTH rule matched has **CHCKCLNT ASQMGR**.
2. If you select NONE and the client connection matches a CHLAUTH record with **CHCKCLNT REQUIRED** (or REQDADM on platforms other than z/OS), the connection fails. You receive the following message:
  - Multi AMQ9793 on [Multiplatforms](#).
  - z/OS CSQX793E on z/OS.
3. This parameter is valid only with **TYPE (USERMAP)**, **TYPE (ADDRESSMAP)** and **TYPE (SSLPEERMAP)**, and only when **USERSRC** is not set to NOACCESS.
4. This parameter applies only to inbound connections that are server-connection channels.

### CHCKLOCL

This attribute determines the authentication requirements for locally bound applications, and is valid only for an **AUTHTYPE** of IDPWOS or IDPWLDAP.

MQ Appliance For information about use of this attribute on IBM MQ Appliance, see [Control commands on the IBM MQ Appliance](#) in the IBM MQ Appliance documentation.

The possible values are:

#### NONE

Authentication credentials supplied by client applications are not checked. If a user ID and password is supplied by a locally bound application, the credentials are ignored.

#### OPTIONAL

Locally bound applications are not required to provide authentication credentials.

Any applications that do provide a user ID and password in the MQCSP structure have them authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection is only allowed to continue if the user ID and password are valid.

Linux V 9.3.4 AIX Authentication tokens cannot be supplied by locally bound applications.

This option might be useful during migration, for example.

#### REQUIRED

All locally bound applications must provide authentication credentials in the MQCSP structure.

If an application provides a user ID and password, these credentials are authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection will only be allowed to continue if the user ID and password are valid.

Linux V 9.3.4 AIX Authentication tokens cannot be supplied by locally bound applications.

If an application does not provide any authentication credentials, the connection is rejected.

z/OS If your user ID has UPDATE access to the BATCH profile in the MQCONN class, you can treat **CHCKLOCL (REQUIRED)** as if it is **CHCKLOCL (OPTIONAL)**. That is, you do not have to supply a password, but if you do, the password must be the correct one.

See [Using CHCKLOCL on locally bound applications](#).

### REQDADM

All locally bound applications using a privileged user ID must provide authentication credentials in the MQCSP structure. Any locally bound applications using a non-privileged user ID are not required to provide authentication credentials and are treated as with the OPTIONAL setting.

A privileged user is one that has full administrative authorities for IBM MQ. See [Privileged users](#) for more information.

Any provided user ID and password will be authenticated by the queue manager against the password store indicated by the **AUTHTYPE**. The connection will only be allowed to continue if the user ID and password are valid.

Linux V 9.3.4 AIX Authentication tokens cannot be supplied by locally bound applications.

z/OS (This setting is not allowed on z/OS systems.)

### CLASSGRP

The LDAP object class used for group records in the LDAP repository.

If the value is blank, `groupOfNames` is used.

Other commonly used values include `groupOfUniqueNames` or `group`.

### CLASSUSR( LDAP class name )

The LDAP object class used for user records in the LDAP repository.

If blank, the value defaults to `inetOrgPerson`, which is generally the value needed.

For Microsoft Active Directory, the value you require is often `user`.

This attribute is valid only for an **AUTHTYPE** of `IDPWLDAP`.

### z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

### CONNNAME(connection name)

The host name, IPv4 dotted decimal address, or IPv6 hexadecimal notation of the host on which the LDAP server is running, with an optional port number.

If you specify the connection name as an IPv6 address, only systems with an IPv6 stack are able to resolve this address. If the AUTHINFO object is part of the CRL namelist of the queue manager, ensure that any clients using the client channel table generated by the queue manager can resolve the connection name.

**z/OS** On z/OS, if a **CONNAME** is to resolve to an IPv6 network address, a level of z/OS that supports IPv6 for connection to an LDAP server is required.

The syntax for **CONNAME** is the same as for channels. For example,

```
conname(' hostname (nnn)')
```

where *nnn* is the port number.

The maximum length for the field depends on your platform:

- **ALW** On AIX, Linux, and Windows, the maximum length is 264 characters.
- **IBM i** On IBM i, the maximum length is 264 characters.
- **z/OS** On z/OS, the maximum length is 48 characters.

This attribute is valid only for an **AUTHTYPE** of CRLLDAP and IDPWLDAP, when the attribute is mandatory.

When used with an **AUTHTYPE** of IDPWLDAP, this can be a comma separated list of connection names.

#### **DESCR(string)**

Plain-text comment. It provides descriptive information about the authentication information object when an operator issues the **DISPLAY AUTHINFO** command (see [“DISPLAY AUTHINFO \(display authentication information\)”](#) on page 640).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

#### **FAILDLAY(delay time)**

When authentication credentials are provided for connection authentication, and the authentication fails due to the credentials being incorrect, this is the delay, in seconds, before the failure is returned to the application.

This can aid in avoiding busy loops from an application that simply retries, continuously, after receiving a failure.

The value must be in the range 0 - 60 seconds. The default value is 1.

This attribute is only valid for an **AUTHTYPE** of IDPWOS and IDPWLDAP.

#### **FINDGRP**

Name of the attribute used within an LDAP entry to determine group membership.

When **AUTHORMD** = SEARCHGRP, the **FINDGRP** attribute is typically set to member or uniqueMember.

When **AUTHORMD** = SEARCHUSR, the **FINDGRP** attribute is typically set to memberOf.

When **AUTHORMD** = SRCHGRPSN, the **FINDGRP** attribute is typically set to memberUid.

When the **FINDGRP** attribute is left blank:

- If **AUTHORMD** = SEARCHGRP, the **FINDGRP** attribute defaults to memberOf.
- If **AUTHORMD** = SEARCHUSR, the **FINDGRP** attribute defaults to member.
- If **AUTHORMD** = SRCHGRPSN, the **FINDGRP** attribute defaults to memberUid.

## GRPFIELD

LDAP attribute that represents a simple name for the group.

If the value is blank, commands like **setmqaut** must use a qualified name for the group. The value can either be a full DN, or a single attribute.

## LDAPPWD(LDAP password)

The password associated with the Distinguished Name of the user who is accessing the LDAP server. Its maximum size is 32 characters.

This attribute is valid only for an **AUTHTYPE** of CRLLDAP and IDPWLDAP.

**z/OS** On z/OS, the **LDAPPWD** used for accessing the LDAP server might not be the one defined in the **AUTHINFO** object. If more than one **AUTHINFO** object is placed in the namelist referred to by the QMGR parameter **SSLCRLNL**, the **LDAPPWD** in the first **AUTHINFO** object is used for accessing all LDAP servers.

## LDAPUSER(LDAP user)

The Distinguished Name of the user who is accessing the LDAP server. (See the **SSLPEER** parameter for more information about distinguished names.)

This attribute is valid only for an **AUTHTYPE** of CRLLDAP and IDPWLDAP.

The maximum size for the user name is as follows:

- **Multi** 1024 characters on [Multiplatforms](#)
- **z/OS** 256 characters on z/OS

**z/OS** On z/OS, the **LDAPUSER** used for accessing the LDAP Server might not be the one defined in the **AUTHINFO** object. If more than one **AUTHINFO** object is placed in the namelist referred to by the QMGR parameter **SSLCRLNL**, the **LDAPUSER** in the first **AUTHINFO** object is used for accessing all LDAP servers.

**Multi** On [Multiplatforms](#), the maximum accepted line length is defined to be BUFSIZ, which can be found in `stdio.h`.

## LIKE(authinfo-name)

The name of an authentication information object, with parameters that are used to model this definition.

**z/OS** On z/OS, the queue manager searches for an object with the name you specify and a disposition of QMGR or COPY. The disposition of the LIKE object is not copied to the object you are defining.

### Note:

1. **QSGDISP (GROUP)** objects are not searched.
2. LIKE is ignored if **QSGDISP (COPY)** is specified. However, the group object defined is used as a LIKE object.

## NESTGRP

Group nesting.

### NO

Only the initially discovered groups are considered for authorization.

### YES

The group list is searched recursively to enumerate all the groups to which a user belongs.

The group's Distinguished Name is used when searching the group list recursively, regardless of the authorization method selected in [AUTHORMD](#).

### OCSPURL(Responder URL)

The URL of the OCSP responder used to check for certificate revocation. This value must be an HTTP URL containing the host name and port number of the OCSP responder. If the OCSP responder is using port 80, which is the default for HTTP, then the port number can be omitted. HTTP URLs are defined in RFC 1738.

This field is case sensitive. It must start with the string `http://` in lowercase. The rest of the URL might be case sensitive, depending on the OCSP server implementation. To preserve case, use single quotation marks to specify the OCSPURL parameter value, for example:

```
OCSPURL ('http://ocsp.example.ibm.com')
```

This parameter is applicable only for **AUTHTYPE(OCSP)**, when it is mandatory.

### QSGDISP

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

QSGDISP	DEFINE
COPY	<p>The object is defined on the page set of the queue manager that executes the command. It uses the <b>QSGDISP(GROUP)</b> object of the same name as the LIKE object.</p> <p>For example, if you issue the following command,</p> <pre>DEFINE AUTHINFO(auth_name) REPLACE QSGDISP(COPY)</pre> <p>the queue manager searches the shared configuration repository for an AUTHINFO definition called <i>auth_name</i>. If a matching AUTHINFO definition is found, the queue manager creates a local copy of this definition on the queue manager page set.</p> <p>For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.</p>
GROUP	<p>The object definition resides in the shared configuration repository. <b>QSGDISP(GROUP)</b> is allowed only if the queue manager is in a queue sharing group.</p> <p>If the DEFINE for the <b>QSGDISP(GROUP)</b> object is successful, the DEFINE AUTHINFO(<i>auth_name</i>) REPLACE QSGDISP(COPY) command is generated and sent to all active queue managers in the queue sharing group to make or refresh local copies on page set zero.</p> <p>The DEFINE for the group object takes effect regardless of whether the generated command with <b>QSGDISP(COPY)</b> fails.</p>
PRIVATE	Not permitted.
QMGR	The object is defined on the page set of the queue manager that executes the command.

### REPLACE and NOREPLACE

Whether the existing definition (and on z/OS, with the same disposition) is to be replaced with this one. This parameter is optional. Any object with a different disposition is not changed.

**REPLACE**

The definition must replace any existing definition of the same name. If a definition does not exist, one is created.

**NOREPLACE**

The definition must not replace any existing definition of the same name.

**SECCOMM**

Whether connectivity to the LDAP server should be done securely using TLS

**YES**

Connectivity to the LDAP server is made securely using TLS.

The certificate used is the default certificate for the queue manager, named in **CERTLABL** on the queue manager object, or if that is blank, the one described in [Digital certificate labels, understanding the requirements](#).

The certificate is located in the key repository specified in **SSLKEYR** on the queue manager object. A cipherspec will be negotiated that is supported by both IBM MQ and the LDAP server.

If the queue manager is configured to use **SSLFIPS (YES)** or **SUITEB** cipher specs, then this is taken account of in the connection to the LDAP server as well.

**ANON**

Connectivity to the LDAP server is made securely using TLS just as for **SECCOMM (YES)** with one difference.

No certificate is sent to the LDAP server; the connection will be made anonymously. To use this setting, ensure that the key repository specified in **SSLKEYR**, on the queue manager object, does not contain a certificate marked as the default.

**NO**

Connectivity to the LDAP server does not use TLS.

This attribute is valid only for an **AUTHTYPE** of IDPWLDAP.

**SHORTUSR(*user name*)**

A field in the user record to be used as a short user name in IBM MQ.

This field must contain values of 12 characters or less. This short user name is used for the following purposes:

- If LDAP authentication is enabled, but LDAP authorization is not enabled, this is used as an operating system user ID for authorization checks. In this case, the attribute must represent an operating system user ID.
- If LDAP authentication and authorization are both enabled, this is used as the user ID carried with the message in order for the LDAP user name to be rediscovered when the user ID inside the message needs to be used.

For example, on another queue manager, or when writing report messages. In this case, the attribute does not need to represent an operating system user ID, but must be a unique string. An employee serial number is an example of a good attribute for this purpose.

The **SHORTUSR** attribute is valid only for an **AUTHTYPE** of IDPWLDAP and is mandatory.

**USRFIELD(*LDAP field name*)**

If the user ID provided by an application for authentication does not contain a qualifier for the field in the LDAP user record, that is, it does not contain an equals (=) sign, this attribute identifies the field in the LDAP user record that is used to interpret the provided user ID.

This field can be blank. If this is the case, any unqualified user IDs use the **SHORTUSR** parameter to interpret the provided user ID.

The contents of this field will be concatenated with an '=' sign, together with the value provided by the application, to form the full user ID to be located in an LDAP user record. For example, the application provides a user of `fred` and this field has the value `cn`, then the LDAP repository will be searched for `cn=fred`.



This attribute is valid only for an **AUTHTYPE** of **IDPWLDAP**.

## **z/OS** DEFINE BUFFPOOL (define a buffer pool) on z/OS

Use the MQSC command DEFINE BUFFPOOL to define a buffer pool that is used for holding messages in main storage.

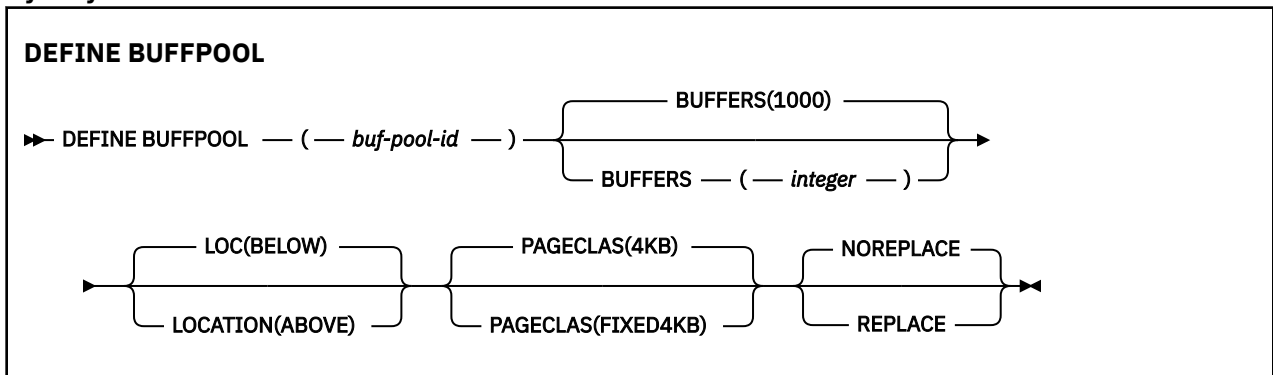
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 1. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 457](#)
- [“Parameter descriptions for DEFINE BUFFPOOL” on page 457](#)

**Synonym:** DEF BP



### Usage notes

1. Specify DEFINE BUFFPOOL commands in a data set identified by the CSQINP1 DD concatenation in the queue manager started task procedure.
2. Use the DISPLAY USAGE TYPE(PAGESET) command to display buffer pool information (see [“DISPLAY USAGE \(display usage information\) on z/OS” on page 873](#) ).
3. Use the ALTER BUFPOOL command to dynamically change the settings of a predefined buffer pool (see [“ALTER BUFFPOOL \(alter buffer pool settings\) on z/OS” on page 266](#) ).

### Parameter descriptions for DEFINE BUFFPOOL

If more than one DEFINE BUFFPOOL command is issued for the same buffer pool, only the last one is processed.

#### **(*buf-pool-id*)**

Buffer pool identifier.

This parameter is an integer in the range zero through 99.

#### **BUFFERS( *integer* )**

This parameter is required and is the number of 4096 byte buffers to be used in this buffer pool.

If the value of the **LOCATION** parameter is **BELW**, the minimum value of buffers is 100 and the maximum value is 500,000. If the value of the **LOCATION** parameter is **ABOVE**, then valid values are in the range of 100 to 999999999 (nine nines). The storage used for buffers in a buffer pool with **LOCATION ABOVE** is obtained in multiples of 4MB. Therefore specifying a **BUFFERS** value which is a multiple of 1024 will make the most efficient use of storage.

## **LOCATION(LOC)(BELOW or ABOVE)**

**LOCATION** and **LOC** are synonyms and either, but not both, can be used.

The **LOCATION** or **LOC** parameter specifies where the memory used by the specified buffer pool is located.



**Attention:** Deprecated From IBM MQ 9.1, **LOCATION(BELOW)** is deprecated and you should use **LOCATION(ABOVE)** only.

This memory location can be either **ABOVE** (64 bit) or **BELOW** (31 bit) the bar. Valid values for this parameter are **BELOW** or **ABOVE**, with **BELOW** being the default.

When altering a buffer pool, you should take care to make sure that there is sufficient storage available if increasing the number of buffers, or changing the **LOCATION** value. Switching the location of the buffer pool can be a CPU and I/O intensive task. You should perform this task when the queue manager is not being heavily used.

For more information, see [Address space storage](#).

## **PAGECLAS( 4KB or FIXED4KB )**

Optional parameter that describes the type of virtual storage pages used for backing the buffers in the buffer pool.

This attribute applies to all buffers in the buffer pool, including any that are added later as a result of using the **ALTER BUFFPOOL** command. The default value is **4KB**, which means that pageable 4KB pages are used to back the buffers in the pool.

**4KB** is the only valid value if the buffer pool has its location attribute set to **BELOW**. If the buffer pool has its **LOCATION** attribute set to **ABOVE**, it is also possible to specify **FIXED4KB**. This means that fixed 4KB pages, which are permanently in real storage and will never be paged out to auxiliary storage, are used to back the buffers in the buffer pool.

The **PAGECLAS** attribute of a buffer pool can be altered at any time. However, the alteration only takes place when the buffer pool switches location from above the bar, to below the bar, or the other way round. Otherwise, the value is stored in the log of the queue manager and is applied when the queue manager next restarts.

When you specify **PAGECLAS(FIXED4KB)** the whole buffer pool is backed by page-fixed 4KB pages, so ensure that there is sufficient real storage available on the LPAR. Otherwise, the queue manager might not start, or other address spaces might be impacted; for more information, see [Address space storage](#).

See IBM MQ Support Pac MP16: [IBM MQ for z/OS - Capacity planning & tuning](#) for advice on when to use the **FIXED4KB** value of the **PAGECLAS** attribute.

## **REPLACE/NOREPLACE**

Optional attribute describing whether this definition of a buffer pool overrides any definition that might already be contained in the log of the queue manager.

The queue manager records the current buffer pool settings in checkpoint log records. These buffer pool settings are automatically restored when a queue manager is later restarted. This restoration occurs after processing of the **CSQINP1** data set.

Therefore, if you have used **ALTER BUFFPOOL** since the buffer pool was last defined, any **DEFINE BUFFPOOL** command in **CSQINP1** will be ignored at restart, unless the **REPLACE** attribute is specified.



**Attention:** If you specify the **REPLACE** attribute on the **DEFINE BUFFPOOL** command in **CSQINP1**, then sometime later you specify the **ALTER BUFFPOOL** command to increase the buffer pool size due to an increase in workload, and hence demand on buffer pool usage for example, this could lead to an issue when the queue manager is next restarted.

Following restart, your buffer pools would revert to the original size defined in **CSQINP1** and hence not be able to handle the increased workload. Therefore, you should consider removing

the REPLACE attribute from the DEFINE BUFFPOOL commands in CSQINP1 as soon as the buffer pools are successfully defined.

This will ensure that any changes made to the buffer pools, as a result of specifying ALTER BUFFPOOL commands, are preserved across restarts of the queue manager.

#### **REPLACE**

This definition of the buffer pool overrides the definition stored in the log of the queue manager, if there is one. If the definition in the log of the queue manager is different from this definition, the differences are discarded and message [CSQP064I](#) is issued.

#### **NOREPLACE**

This is the default value, and provides the same behavior as with previous releases of IBM MQ. If there is a definition of the buffer pool in the log of the queue manager that is used, and this definition is ignored.



**Attention:** The queue manager records the current buffer pool settings in checkpoint log records. These buffer pool settings are automatically restored when a queue manager is later restarted. This restoration occurs after processing of the CSQINP1 data set. Therefore, if you have used **ALTER BUFFPOOL** since the buffer pool was last defined, any **DEFINE BUFFPOOL** command in CSQINP1 has been ignored at restart, unless the **REPLACE** attribute has been specified.

## **DEFINE CFSTRUCT (define coupling facility application structure)** **on z/OS**

Use the MQSC command DEFINE CFSTRUCT to define queue manager CF level capability, message offload environment, and backup and recovery parameters for a coupling facility application structure.

### **Using MQSC commands on z/OS**

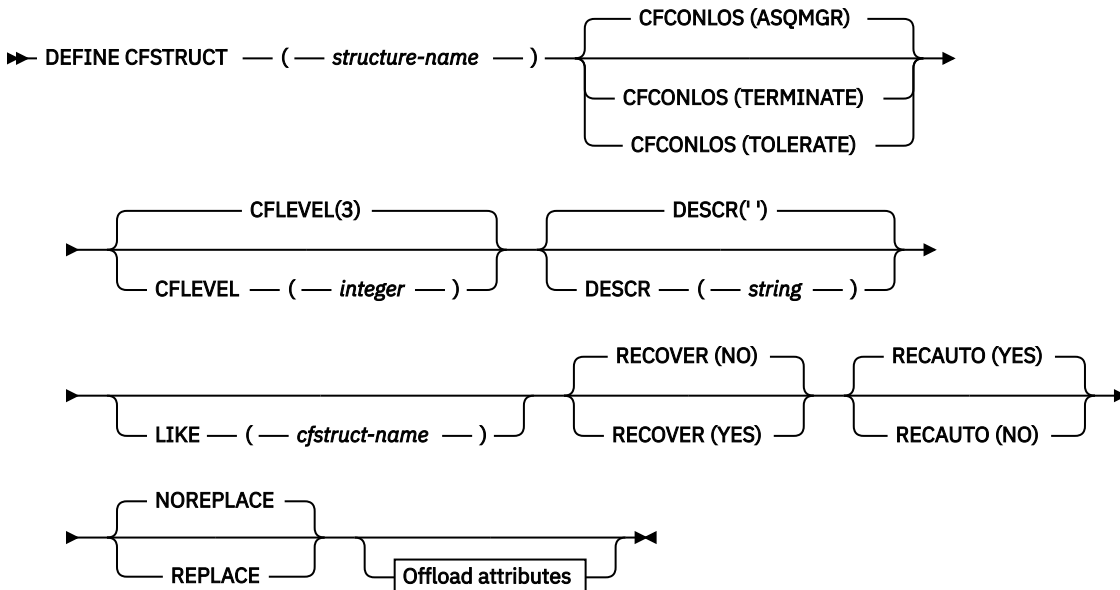
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

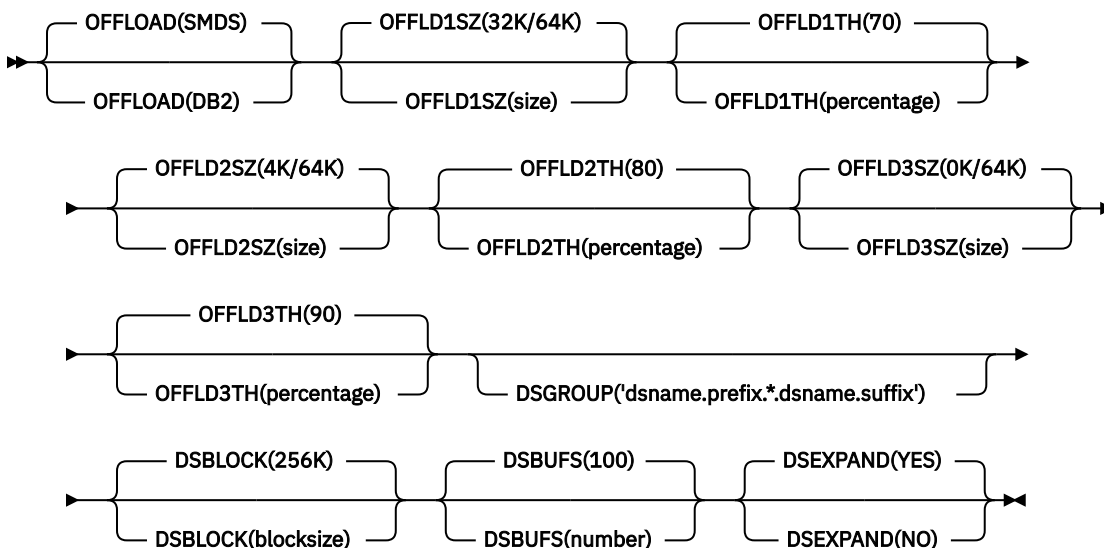
- [Syntax diagram](#)
- [“Usage notes for DEFINE CFSTRUCT” on page 460](#)
- [“Parameter descriptions for DEFINE CFSTRUCT” on page 460](#)

**Synonym:** DEF CFSTRUCT

## DEFINE CFSTRUCT



### Offload attributes



## Usage notes for DEFINE CFSTRUCT

1. This command is valid only on z/OS when the queue manager is a member of a queue sharing group.
2. This command cannot specify the CF administration structure (CSQ\_ADMIN).
3. Before any newly defined CF structure can be used by any queues, the structure must be defined in the Coupling Facility Resource Management (CFRM) policy data set.
4. Only CF structures with RECOVER(YES) defined can be backed up and recovered.

## Parameter descriptions for DEFINE CFSTRUCT

### (*structure-name*)

Name of the coupling facility application structure that has queue manager CF level capability and backup and recovery parameters you want to define. This parameter is required.

The name:

- Cannot have more than 12 characters.
- Must start with an uppercase letter (A through Z).
- Can include only the characters A through Z and 0 through 9.

The name of the queue sharing group to which the queue manager is connected is prefixed to the name you supply. The name of the queue sharing group is always four characters, padded with @ symbols if necessary. For example, if you use a queue sharing group named NY03 and you supply the name PRODUCT7, the resultant coupling facility structure name is NY03PRODUCT7. The administrative structure for the queue sharing group (in this case NY03CSQ\_ADMIN) cannot be used for storing messages.

## **CFCONLOS**

This parameter specifies the action to be taken when a queue manager loses connectivity to the CF structure. The value can be:

### **ASQMGR**

The action taken is based on the setting of the CFCONLOS queue manager attribute.

### **TERMINATE**

The queue manager ends when connectivity to the structure is lost.

### **TOLERATE**

The queue manager tolerates loss of connectivity to the structure without terminating.

This parameter is only valid from CFLEVEL(5).

## **CFLEVEL( *integer* )**

Specifies the functional capability level for this CF application structure. Value can be one of the following:

### **1**

A CF structure that can be "auto-created" by a queue manager at command level 520.

### **2**

A CF structure at command level 520 that can only be created or deleted by a queue manager at command level 530 or greater.

### **3**

A CF structure at command level 530. This CFLEVEL is required if you want to use persistent messages on shared queues (if RECOVER(YES) is set), or for message grouping (when a local queue is defined with INDXTYPE(GROUPID)), or both.

You can only increase the value of CFLEVEL to 3 if all the queue managers in the queue sharing group are at command level 530 or greater - this is to ensure that there are no latent command level 520 connections to queues referencing the structure.

You can only decrease the value of CFLEVEL from 3 if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

### **4**

This CFLEVEL supports all the CFLEVEL(3) functions. CFLEVEL(4) allows queues defined with CF structures at this level to have messages with a length greater than 63 KB.

Only a queue manager with a command level of 600 or above can connect to a CF structure at CFLEVEL(4).

You can only increase the value of CFLEVEL to 4 if all the queue managers in the queue sharing group are at command level 600 or greater.

You can only decrease the value of CFLEVEL from 4 if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

## 5

This CFLEVEL supports all functions for CFLEVEL(4). In addition, CFLEVEL(5) enables the following new functions. If altering an existing CFSTRUCT to CFLEVEL(5), you must review other attributes as indicated:

- queues defined with CF structures at this level can have message data offloaded to either shared message data sets (SMDS), or Db2, under control of the OFFLOAD attribute. The offload threshold and size parameters (such as OFFLD1TH, and OFFLD1SZ) determine whether any particular messages are offloaded given its size and current CF structure utilization. If using SMDS offload, the DSGROUP, DSBUFS, DSEXPAND and DSBLOCK attributes are respected.
- structures at CFLEVEL(5) allow the queue manager to tolerate a loss of connectivity to the CF structure. The CFCONLOS attribute determines queue manager behavior when a loss of connectivity is detected, and the RECAUTO attribute controls subsequent automatic structure recovery behavior.
- messages containing IBM MQ message properties are stored in a different format on shared queues in a CFLEVEL(5) structure. This format leads to internal processing optimizations. Additional application migration capabilities are also available and these are enabled via the queue PROPCTL attribute.

Only a queue manager with a command level of 710 or above can connect to a CF structure at CFLEVEL(5).

### **Note:**

You can decrease the value of CFLEVEL from 5 if all the queues that reference the CF structure are both empty, that is the queues, and CF structure have no messages or uncommitted activity, and are closed.

### **DESCR( *string* )**

Plain-text comment that provides descriptive information about the object when an operator issues the DISPLAY CFSTRUCT command.

The string should contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

### **LIKE( *cfstruct-name* )**

The name of a CFSTRUCT object, with attributes used to model this definition.

The initial values of all attributes are copied from the object, except any DSGROUP attribute is ignored because each structure requires its own unique value.

### **OFFLOAD**

Specify whether offloaded message data is to be stored in a group of shared message data sets or in Db2.

#### **SMDS**

Offload messages from coupling facility to shared message data set (SMDS). This value is the default assumption when a new structure is defined with CFLEVEL(5).

#### **DB2**

Offload messages from coupling facility to Db2. This value is the default assumption when an existing structure is increased to CFLEVEL(5) using DEFINE with the REPLACE option.

Offloading messages using Db2 has significant performance impact. If you want to use the offload rules as a means of increasing capacity, the SMDS option should be specified or assumed.

This parameter is only valid from CFLEVEL(5). At CFLEVEL(4) any message offloading is always to Db2, and only applies to messages greater than the maximum coupling facility entry size.

### **Note:**

If you change the offload technique (from Db2 to SMDS or the other way) then all new messages will be written using the new method but any existing large messages stored using the previous technique can still be retrieved. The relevant Db2 message table or shared message data sets will continue to be used until the queue managers have detected that there are no further messages stored in the old format.

If SMDS is specified or assumed, then the DSGROUP parameter is also required. It can be specified either on the same command or on a previous DEFINE or ALTER command for the same structure.

**OFFLD1TH(percentage) OFFLD1SZ(size)**  
**OFFLD2TH(percentage) OFFLD2SZ(size)**  
**OFFLD3TH(percentage) OFFLD3SZ(size)**

Specify rules for when messages smaller than the maximum coupling facility entry size are to be offloaded to external storage (shared message data sets or Db2 tables) instead of being stored in the application structure. These rules can be used to increase the effective capacity of the structure. The offloaded message still requires an entry in the coupling facility containing message control information, and a descriptor referring to the offloaded message data, but the amount of structure space required is less than the amount that would be needed to store the whole message.

If the message data is very small (of the order of 100 bytes) it might fit into the same coupling facility entry as the message control information, without needing additional data elements. In this case, no space can be saved, so any offload rules are ignored and the message data is not offloaded. The actual number varies, depending whether more than the default headers are used, or if message properties are being stored.

Messages exceeding the maximum coupling facility entry size (63.75 KB including control information) are always offloaded as they cannot be stored in a coupling facility entry. Messages where the message body exceeds 63 KB are also offloaded to ensure that enough space is available for the control information. Additional rules to request offloading of smaller messages can be specified using these pairs of keywords. Each rule indicates that when the usage of the structure (in either elements or entries) exceeds the specified threshold percentage value, the message data will be offloaded if the total size of the coupling facility entry required to store the whole message (including message data, headers and descriptors) exceeds the specified size value. The minimal set for the two elements of headers and descriptors require 512 bytes, however this could be greater if other headers or properties are added. This figure would also be greater if an MQMD version greater than 1 is used.

#### **percentage**

The usage threshold percentage value is an integer in the range 0 (meaning this rule always applies) to 100 (meaning this rule only applies when the structure is full). For example, OFFLD1TH(75) OFFLD1SZ(32K) means that when the structure is over 75% full, messages greater than 32 kilobytes in size are offloaded.

#### **size**

The message size value should be specified as an integer followed by K, giving the number of kilobytes in the range **0K** to **64K**. As messages exceeding 63.75 KB are always offloaded, the value 64K is allowed as a simple way to indicate that the rule is not being used.

In general, the smaller the numbers, the more messages are offloaded.

A message is offloaded if any offload rule matches. The normal convention is that a later rule would be for a higher usage level and a smaller message size than an earlier one, but no check is made for consistency or redundancy between the rules.

When structure ALTER processing is active, the number of used elements or entries can temporarily exceed the reported total number, giving a percentage exceeding 100, because the new elements or entries are made available during ALTER processing but the total is only updated when the ALTER completes. At such times, a rule specifying 100 for the threshold may temporarily take effect. If a rule is not intended to be used at all, it should specify 64K for the size.

The default values assumed for the offload rules when defining a new structure at CFLEVEL(5) or upgrading an existing structure to CFLEVEL(5) depend on the OFFLOAD method option. For OFFLOAD(SMDS), the default rules specify increasing amounts of offloading as the structure becomes full. This increases the effective structure capacity with minimal performance impact. For OFFLOAD( Db2 ), the default rules have the same threshold values as for SMDS but the size values are set to 64K so that the rules never apply and messages are offloaded only if they are too large to be stored in the structure, as for CFLEVEL(4).

For OFFLOAD(SMDS) the defaults are:

- OFFLD1TH(70) OFFLD1SZ(32K)
- OFFLD2TH(80) OFFLD2SZ(4K)
- OFFLD3TH(90) OFFLD3SZ(0K)

For OFFLOAD( Db2 ) the defaults are:

- OFFLD1TH(70) OFFLD1SZ(64K)
- OFFLD2TH(80) OFFLD2SZ(64K)
- OFFLD3TH(90) OFFLD3SZ(64K)

If the OFFLOAD method option is changed from Db2 to SMDS or back when the current offload rules all match the default values for the old method, the offload rules are switched to the default values for the new method. However, if any of the rules have been changed, the current values are kept when switching method.

These parameters are only valid from CFLEVEL(5). At CFLEVEL(4) any message offloading is always to Db2, and only applies to messages greater than the maximum coupling facility entry size.

## **DSGROUP**

For OFFLOAD(SMDS), specify the generic data set name to be used for the group of shared message data sets associated with this structure (one for each queue manager), with exactly one asterisk indicating where the queue manager name should be inserted to form the specific data set name.

### **dsname.prefix.\*.dsname.suffix**

The value must be a valid data set name when the asterisk is replaced by a queue manager name of up to four characters.

The entire parameter value must be enclosed in quotation marks.

This parameter cannot be changed after any data sets have been activated for the structure.

If SMDS is specified or assumed, then the DSGROUP parameter must also be specified.

This parameter is only valid from CFLEVEL(5).

## **DSBLOCK**

For OFFLOAD(SMDS), specify the logical block size, which is the unit in which shared message data set space is allocated to individual queues.

**8K**  
**16K**  
**32K**  
**64K**  
**128K**  
**256K**  
**512K**  
**1M**

Each message is written starting at the next page within the current block and is allocated further blocks as needed. A larger size decreases space management requirements and reduces I/O for



large messages, but increases buffer space requirements and disk space requirements for small queues.

This parameter cannot be changed after any data sets have been activated for the structure.

This parameter is only valid from CFLEVEL(5).

## **DSBUFS**

For OFFLOAD(SMDS), specify the number of buffers to be allocated in each queue manager for accessing shared message data sets, as a number in the range 1 - 9999. The size of each buffer is equal to the logical block size. SMDS buffers are allocated in memory objects residing in z/OS 64-bit storage (above the bar).

### **number**

This parameter can be overridden for individual queue managers using the DSBUFS parameter on ALTER SMDS.

When this parameter is altered, any queue managers which are already connected to the structure (and which do not have an individual DSBUFS override value) dynamically increase or decrease the number of data set buffers being used for this structure to match the new value. If the specified target value cannot be reached, the affected queue manager adjusts the DSBUFS parameter associated with its own individual SMDS definition (as for the ALTER SMDS command) to match the actual new number of buffers.

This parameter is only valid from CFLEVEL(5).

## **DSEXPAND**

For OFFLOAD(SMDS), this parameter controls whether the queue manager should expand a shared message data set when it becomes nearly full, and further blocks are required in the data set.

### **YES**

Expansion is supported.

Each time expansion is required, the data set is expanded by the secondary allocation specified when the data set was defined. If no secondary allocation was specified, or it was specified as zero, then a secondary allocation amount of approximately 10% of the existing size is used

### **NO**

No automatic data set expansion is to take place.

This parameter can be overridden for individual queue managers using the DSEXPAND parameter on ALTER SMDS.

If an expansion attempt fails, the DSEXPAND override for the affected queue manager is automatically changed to NO to prevent further expansion attempts, but it can be changed back to YES using the ALTER SMDS command to enable further expansion attempts.

When this parameter is altered, any queue managers which are already connected to the structure (and which do not have an individual DSEXPAND override value) immediately start using the new parameter value.

This parameter is only valid from CFLEVEL(5).

## **RECOVER**

Specifies whether CF recovery is supported for the application structure. Values are:

### **NO**

CF application structure recovery is not supported. (The synonym is **N**.)

### **YES**

CF application structure recovery is supported. (The synonym is **Y**.)

You can only set RECOVER(YES) if the structure has a CFLEVEL of 3 or higher. Set RECOVER(YES) if you intend to use persistent messages.

You can only change RECOVER(NO) to RECOVER(YES) if all the queue managers in the queue sharing group are at command level 530 or greater; this is to ensure that there are no latent command level 520 connections to queues referencing the CFSTRUCT.

You can only change RECOVER(YES) to RECOVER(NO) if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

### **RECAUTO**

Specifies the automatic recovery action to be taken when a queue manager detects that the structure is failed or when a queue manager loses connectivity to the structure and no systems in the sysplex have connectivity to the coupling facility that the structure is allocated in. Values can be:

#### **YES**

The structure and associated shared message data sets which also need recovery will be automatically recovered (The synonym is **Y**.)

#### **NO**

The structure will not be automatically recovered. (The synonym is **N**.)

This parameter has no effect for structures defined with RECOVER(NO).

This parameter is only valid from CFLEVEL(5).

### **REPLACE and NOREPLACE**

Defines whether the existing definition is to be replaced with this one. This parameter is optional.

#### **REPLACE**

The definition should replace any existing definition of the same name. If a definition does not exist, one is created. If you use the REPLACE option, all queues that use this CF structure must be empty and closed.

#### **NOREPLACE**


The definition should not replace any existing definition of the same name.

## **DEFINE CHANNEL (define a new channel)**

Use the MQSC command **DEFINE CHANNEL** to define a new channel, and set its parameters.

### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Synonym: DEF CHL

- [“Usage notes” on page 466](#)
- [“Parameter descriptions for DEFINE CHANNEL” on page 466](#)

### **Usage notes**

- For CLUSSDR channels, you can specify the REPLACE option only for manually created channels.
- Successful completion of the command does not mean that the action completed. To check for true completion, see the [DEFINE CHANNEL](#) step in [Checking that async commands for distributed networks have finished](#).

### **Parameter descriptions for DEFINE CHANNEL**

The following table shows the parameters that are relevant for each type of channel:

#### **SDR**

[“Sender channel” on page 506](#)

**SVR**

[“Server channel” on page 508](#)

**RCVR**

[“Receiver channel” on page 510](#)

**RQSTR**

[“Requester channel” on page 512](#)

**CLNTCONN**

[“Client-connection channel” on page 514](#)

**SVRCONN**

[“Server-connection channel” on page 516](#)

**CLUSDR**

[“Cluster-sender channel” on page 518](#)

**CLUSRCVR**

[“Cluster-receiver channel” on page 520](#)

**Multi AMQP**

[“AMQP channel” on page 522](#)

There is a description of each parameter after the table. Parameters are optional unless the description states that they are required.

<i>Table 148. DEFINE and ALTER CHANNEL parameters</i>									
Parameter	SDR	SVR	RCVR	RQSTR	CLNTCONN	SVRCONN	CLUSDR	CLUSRCVR	AMQP
<a href="#">AFFINITY</a>					✓				
<a href="#">AMQPKA</a>									✓
<a href="#">BACKLOG</a>									
<a href="#">BATCHHB</a>	✓	✓					✓	✓	
<a href="#">BATCHINT</a>	✓	✓					✓	✓	
<a href="#">BATCHLIM</a>	✓	✓					✓	✓	
<a href="#">BATCHSZ</a>	✓	✓	✓	✓			✓	✓	
<a href="#">CERTLABL</a>	✓	✓	✓	✓	✓	✓		✓	✓
<a href="#">channel-name</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">CHLTYPE</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">CLNTWGHT</a>					✓				
<a href="#">CLUSNL</a>							✓	✓	
<a href="#">CLUSTER</a>							✓	✓	
<a href="#">CLWLPRTY</a>							✓	✓	
<a href="#">CLWLRANK</a>							✓	✓	
<a href="#">CLWLWGHT</a>							✓	✓	

Table 148. DEFINE and ALTER CHANNEL parameters (continued)


Parameter	SDR	SVR	RCVR	RQSTR	CLNTC ONN	SVRCO NN	CLUSSD R	CLUSR CVR	AMQP
 <b>CMDSCOPE</b>	✓	✓	✓	✓	✓	✓	✓	✓	
<b>COMPHDR</b>	✓	✓	✓	✓	✓	✓	✓	✓	
<b>COMPMSG</b>	✓	✓	✓	✓	✓	✓	✓	✓	
<b>CONNAME</b>	✓	✓		✓	✓		✓	✓	
<b>CONVERT</b>	✓	✓					✓	✓	
<b>DEFCDISP</b>	✓	✓	✓	✓		✓			
<b>DEFRECON</b>					✓				
<b>DESCR</b>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>DISCINT</b>	✓	✓				✓	✓	✓	
<b>HBINT</b>	✓	✓	✓	✓	✓	✓	✓	✓	
<b>JAASCFG</b>									
<b>KAINT</b>	✓	✓	✓	✓	✓	✓	✓	✓	
<b>LIKE</b>	✓	✓	✓	✓	✓	✓	✓	✓	
<b>LOCLADDR</b>	✓	✓		✓	✓		✓	✓	✓
<b>LONGRTY</b>	✓	✓					✓	✓	
<b>LONGTMR</b>	✓	✓					✓	✓	
<b>MAXINST</b>						✓			✓
<b>MAXINSTC</b>						✓			
<b>MAXMSGL</b>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>MCANAME</b>	✓	✓		✓			✓	✓	
<b>MCATYPE</b>	✓	✓		✓			✓	✓	
<b>MCAUSER</b>			✓	✓		✓		✓	✓
<b>MODENAME</b>	✓	✓		✓	✓		✓	✓	
<b>MONCHL</b>	✓	✓	✓	✓		✓	✓	✓	
<b>MRDATA</b>			✓	✓				✓	
<b>MREXIT</b>			✓	✓				✓	
<b>MRRTY</b>			✓	✓				✓	

Table 148. DEFINE and ALTER CHANNEL parameters (continued)



Parameter	SDR	SVR	RCVR	RQSTR	CLNTC ONN	SVRCO NN	CLUSSD R	CLUSR CVR	AMQP
<u>MRTMR</u>			✓	✓				✓	
<u>MSGDATA</u>	✓	✓	✓	✓			✓	✓	
<u>MSGEXIT</u>	✓	✓	✓	✓			✓	✓	
<u>NETPRTY</u>								✓	
<u>NPMSPEED</u>	✓	✓	✓	✓			✓	✓	
<u>PASSWORD</u>	✓	✓		✓	✓		✓		
<u>PORT</u>									✓
<u>PROPCTL</u>	✓	✓					✓	✓	
<u>PUTAUT</u>			✓	✓		✓		✓	
<u>QMNAME</u>					✓				
 <u>QSGDISP</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>RCVDATA</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>RCVEXIT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>REPLACE</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SCYDATA</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SCYEXIT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SENDDATA</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SENDEXIT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SEQWRAP</u>	✓	✓	✓	✓			✓	✓	
<u>SHARECNV</u>					✓	✓			
<u>SHORTRTY</u>	✓	✓					✓	✓	
<u>SHORTTMR</u>	✓	✓					✓	✓	
 <u>SPLPROT</u>	✓	✓	✓	✓					
<u>SSLCAUTH</u>		✓	✓	✓		✓		✓	
<u>SSLCIPH</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>SSLKEYP</u>									

Table 148. DEFINE and ALTER CHANNEL parameters (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNTC ONN	SVRCO NN	CLUSSD R	CLUSR CVR	AMQP
<u>SSLPEER</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>STATCHL</u>	✓	✓	✓	✓			✓	✓	
<div style="background-color: #d9d9d9; padding: 2px;">Multi</div> <div style="background-color: #000080; color: white; padding: 2px;">V 9.3.0</div> <u>TMPMODEL</u>									✓
<div style="background-color: #d9d9d9; padding: 2px;">Multi</div> <div style="background-color: #000080; color: white; padding: 2px;">V 9.3.0</div> <u>TMPQPRFX</u>									✓
<u>TPNAME</u>	✓	✓		✓	✓	✓	✓	✓	
<u>TPROOT</u>									✓
<u>TRPTYPE</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>USECLTID</u>									✓
<u>USEDLQ</u>	✓	✓	✓	✓			✓	✓	
<u>USERID</u>	✓	✓		✓	✓		✓		
<u>XMITQ</u>	✓	✓							

#### AFFINITY

Use the channel affinity attribute when client applications connect multiple times using the same queue manager name. With the attribute, you can choose whether the client uses the same client channel definition for each connection. This attribute is intended to be used when multiple applicable channel definitions are available.

#### PREFERRED

The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions. The list is based on the weightings, with any applicable **CLNTWGHT (0)** definitions first and in alphabetic order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non- **CLNTWGHT (0)** definitions are moved to the end of the list. **CLNTWGHT (0)** definitions remain at the start of the list and are selected first for each connection. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT was modified since the list was created. Each client process with the same host name creates the same list.

#### NONE

The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable **CLNTWGHT (0)** definitions selected first in alphabetic order. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT was modified since the list was created.

For example, suppose that we had the following definitions in the CCDT:

```

CHLNAME(A) QMNAME (QM1) CLNTWGHT(3)
CHLNAME(B) QMNAME (QM1) CLNTWGHT(4)
CHLNAME(C) QMNAME (QM1) CLNTWGHT(4)
    
```

The first connection in a process creates its own ordered list based on the weightings. So it might, for example, create the ordered list CHLNAME(B), CHLNAME(A), CHLNAME(C).

For **AFFINITY(PREFERRED)**, each connection in the process attempts to connect using CHLNAME(B). If a connection is unsuccessful the definition is moved to the end of the list which now becomes CHLNAME(A), CHLNAME(C), CHLNAME(B). Each connection in the process then attempts to connect using CHLNAME(A).

For **AFFINITY(NONE)**, each connection in the process attempts to connect using one of the three definitions selected at random based on the weightings.

If sharing conversations is enabled with a non-zero channel weighting and **AFFINITY(NONE)**, multiple connections do not have to share an existing channel instance. They can connect to the same queue manager name using different applicable definitions rather than sharing an existing channel instance.

#### **Multi** **AMQPKA(integer)**

The keep alive time for an AMQP channel in milliseconds. If the AMQP client has not sent any frames within the keep alive interval, then the connection is closed with an `amqp:resource-limit-exceeded` AMQP error condition.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of AMQP

#### **BATCHHB(integer)**

Specifies whether batch heartbeats are to be used. The value is the length of the heartbeat in milliseconds.

Batch heartbeats allow a sending channel to verify that the receiving channel is still active just before committing a batch of messages. If the receiving channel is not active, the batch can be backed out rather than becoming in-doubt, as would otherwise be the case. By backing out the batch, the messages remain available for processing so they could, for example, be redirected to another channel.

If the sending channel received a communication from the receiving channel within the batch heartbeat interval, the receiving channel is assumed to be still active. If not, a 'heartbeat' is sent to the receiving channel to check.

The value must be in the range 0 - 999999. A value of zero indicates that batch heart beats are not used.

This parameter is valid for channels with a channel type (**CHLTYPE**) of only SDR, SVR, CLUSSDR, and CLUSRCVR.

#### **BATCHINT(integer)**

The minimum amount of time, in milliseconds, that a channel keeps a batch open.

The batch is terminated when one of the following conditions is met:

- **BATCHSZ** messages are sent.
- **BATCHLIM** kilobytes are sent.
- The transmission queue is empty and **BATCHINT** is exceeded.

The value must be in the range 0 - 999999999. Zero means that the batch is terminated as soon as the transmission queue becomes empty, or the **BATCHSZ** limit is reached.

This parameter is valid for channels with a channel type (**CHLTYPE**) of only SDR, SVR, CLUSSDR, and CLUSRCVR.

#### **BATCHLIM(integer)**

The limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point. A sync point is taken after the message that caused the limit to be reached flows across the channel. A value of zero in this attribute means that no data limit is applied to batches over this channel.

The batch is terminated when one of the following conditions is met:

- **BATCHSZ** messages are sent.
- **BATCHLIM** kilobytes are sent.
- The transmission queue is empty and **BATCHINT** is exceeded.

This parameter is valid for channels with a channel type (**CHLTYPE**) of only SDR, SVR, CLUSSDR, and CLUSRCVR.





The value must be in the range 0 - 999999. The default value is 5000.

This parameter is supported on all platforms.

### **BATCHSZ(integer)**

The maximum number of messages that can be sent through a channel before taking a sync point.

The maximum batch size used is the lowest of the following values:

- The **BATCHSZ** of the sending channel.
- The **BATCHSZ** of the receiving channel.
-  On z/OS, three less than the maximum number of uncommitted messages allowed at the sending queue manager (or one if this value is zero or less).
-  On Multiplatforms, the maximum number of uncommitted messages allowed at the sending queue manager (or one if this value is zero or less).
-  On z/OS, three less than the maximum number of uncommitted messages allowed at the receiving queue manager (or one if this value is zero or less).
-  On Multiplatforms, the maximum number of uncommitted messages allowed at the receiving queue manager (or one if this value is zero or less).

While non-persistent messages sent over an **NPMSPEED(FAST)** channel are delivered to a queue immediately (without waiting for a complete batch), the messages still contribute to the batch size for a channel and, therefore, cause confirm flows to occur when **BATCHSZ** messages have flowed.

If the batch flows are causing a performance impact when moving only non-persistent messages, and **NPMSPEED** is set to FAST, you should consider setting the **BATCHSZ** to the maximum permissible value of 9999, and **BATCHLIM** to zero.

Additionally, setting **BATCHINT** to a high value, for example, 999999999 keeps each batch "open" for longer, even if there are no new messages waiting on the transmission queue.

The above settings minimize the frequency of confirm flows, but be aware that if any persistent messages are moved over a channel with these settings, there will be significant delays in the delivery of those persistent messages only.


The maximum number of uncommitted messages is specified by the **MAXUMSGS** parameter of the **ALTER QMGR** command.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR.

The value must be in the range 1 - 9999.

### **CERTLABL**

Certificate label for this channel to use.

The label identifies which personal certificate in the key repository is sent to the remote peer. If this attribute is blank, the certificate is determined by the queue manager **CERTLABL**, or  on z/OS the **CERTQSGL** (if the queue manager is part of a queue sharing group) parameter.

Note that inbound channels (including receiver, requester, cluster-receiver, unqualified server, and server-connection channels) only send the configured certificate if the IBM MQ version of the remote



peer fully supports certificate label configuration, and the channel is using a TLS CipherSpec. See [Interoperability of Elliptic Curve and RSA CipherSpecs](#) for further information.

An unqualified server channel is one that does not have the CONNAME field set.

In all other cases, the queue manager **CERTLABL** parameter determines the certificate sent. In particular, all current Java and JMS clients only ever receive the certificate configured by the **CERTLABL** parameter of the queue manager, regardless of the channel-specific label setting

You do not need to run the **REFRESH SECURITY TYPE(SSL)** command if you make any changes to **CERTLABL** on a channel. However, you must run a **REFRESH SECURITY TYPE(SSL)** command if you make any changes to **CERTLABL** on the queue manager.

**Note:** It is an error to inquire, or set, this attribute for cluster-sender channels. If you attempt to do so, you receive the error MQRCCF\_WRONG\_CHANNEL\_TYPE. However, the attribute is present in cluster-sender channel objects (including MQCD structures) and a channel auto-definition (CHAD) exit might set it programmatically if required.

### **(channel-name)**

The name of the new channel definition.

This parameter is required on all types of channel.

**Multi** On CLUSSDR channels, this parameter can take a different form to the other channel types. If your convention for naming CLUSSDR channels includes the name of the queue manager, you can define a CLUSSDR channel using the +QMNAME+ construction. After connection to the matching CLUSRCVR channel, IBM MQ substitutes the correct repository queue manager name in place of +QMNAME+ in the CLUSSDR channel definition. See [Components of a cluster](#).

The name must not be the same as any existing channel defined on this queue manager, unless REPLACE or ALTER is specified.

**z/OS** On z/OS, CLNTCONN channel names can duplicate others.

The maximum length of the string is 20 characters, and the string must contain only valid characters; see [Rules for naming IBM MQ objects](#).

**Multi** On CLUSRCVR channels when using automatic cluster setup, this parameter can use some additional inserts:

- +AUTOCL+ resolves to the automatic cluster name
- +QMNAME+ resolves to the local queue manager name.

When using these inserts, both the unexpanded string and the string with the replaced values must fit inside the maximum size of the field. If there are configured automatic cluster full repositories in the AutoCluster configuration, the channel name must also fit in the maximum channel name length when +QMNAME+ is replaced with each of the configured full repository names.

### **CHLTYPE**

Channel type. This parameter is required.

**Multi** On [Multiplatforms](#), it must follow immediately after the (*channel-name*) parameter.

#### **SDR**

Sender channel

#### **SVR**

Server channel

#### **RCVR**

Receiver channel

#### **RQSTR**

Requester channel

**CLNTCONN**

Client-connection channel

**SVRCONN**

Server-connection channel

**CLUSSDR**

CLUSSDR channel.

**CLUSRCVR**

Cluster-receiver channel.

**AMQP**

AMQP channel

**Note:** If you are using the REPLACE option, you cannot change the channel type.

**CLNTWGHT**

Set the client channel weighting attribute to select a client channel definition at random based on its weighting when more than one suitable definition is available. Specify a value in the range 0 - 99.

The special value 0 indicates that no random load balancing is performed and applicable definitions are selected in alphabetic order. To enable random load balancing the value can be in the range 1 - 99, where 1 is the lowest weighting and 99 is the highest.

If a client application issues MQCONN with a queue manager name of *\*name* a client channel definition can be selected at random. The chosen definition is randomly selected based on the weighting. Any applicable **CLNTWGHT(0)** definitions selected are selected first in alphabetic order. Randomness in the selection of client connection definitions is not guaranteed.

For example, suppose that we had the following two definitions in the CCDT:

```
CHLNAME(TO.QM1) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address1) CLNTWGHT(2)
CHLNAME(TO.QM2) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address2) CLNTWGHT(4)
```

A client MQCONN with queue manager name *\*GRP1* would choose one of the two definitions based on the weighting of the channel definition. (A random integer 1 - 6 would be generated. If the integer was in the range 1 through 2, *address1* would be used otherwise *address2* would be used). If this connection was unsuccessful the client would then use the other definition.

The CCDT might contain applicable definitions with both zero and non-zero weighting. In this situation, the definitions with zero weighting are chosen first and in alphabetic order. If these connections are unsuccessful the definitions with non-zero weighting are chosen based on their weighting.

For example, suppose that we had the following four definitions in the CCDT:

```
CHLNAME(TO.QM1) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address1) CLNTWGHT(1)
CHLNAME(TO.QM2) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address2) CLNTWGHT(2)
CHLNAME(TO.QM3) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address3) CLNTWGHT(0)
CHLNAME(TO.QM4) CHLTYPE(CLNTCONN) QMNAME(GRP1) CONNAME(address4) CLNTWGHT(0)
```

A client MQCONN with queue manager name *\*GRP1* would first choose definition *TO.QM3*. If this connection was unsuccessful the client would then choose definition *TO.QM4*. If this connection was also unsuccessful the client would then randomly choose one of the remaining two definitions based on their weighting.

**CLNTWGHT** is supported for all transport protocols.

**CLUSNL(*nlname*)**


The name of the namelist that specifies a list of clusters to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming IBM MQ objects.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR and CLUSRCVR channels. Only one of the resultant values of **CLUSTER** or **CLUSNL** can be nonblank, the other must be blank.

### **CLUSTER(*clustername*)**

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming IBM MQ objects.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR and CLUSRCVR channels. Only one of the resultant values of **CLUSTER** or **CLUSNL** can be nonblank, the other must be blank.

 On CLUSRCVR channels, when using automatic cluster setup, this parameter can use the value +AUTOCL+, which is automatically expanded to the name of the automatic cluster.

### **CLWLPRTY(*integer*)**

Specifies the priority of the channel for the purposes of cluster workload distribution. The value must be in the range 0 - 9 where 0 is the lowest priority and 9 is the highest.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR and CLUSRCVR channels.

For more information about this attribute, see [CLWLPRTY channel attribute](#).

### **CLWLRANK(*integer*)**

Specifies the rank of the channel for the purposes of cluster workload distribution. The value must be in the range 0 - 9 where 0 is the lowest rank and 9 is the highest.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR and CLUSRCVR channels.

For more information about this attribute, see [CLWLRANK channel attribute](#).

### **CLWLWGHT(*integer*)**

Specifies the weighting to be applied to a channel so that the proportion of messages sent down the channel can be controlled by workload management. The value must be in the range 1 - 99 where 1 is the lowest rank and 99 is the highest.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLUSSDR and CLUSRCVR channels.

For more information about this attribute, see [CLWLWGHT channel attribute](#).

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must either be left blank, or if **QSGDISP** is set to GROUP, the local queue manager name.

• •

The command runs on the queue manager on which it was entered.

#### ***QmgrName***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which the command was entered. To do so, you must be using a shared queue environment, and the command server must be enabled.

★

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of ★ is the same as entering the command on every queue manager in the queue sharing group.

### **COMPHDR**

The list of header data compression techniques supported by the channel.

For SDR, SVR, CLUSSDR, CLUSRCVR, and CLNTCONN channels, the values must be specified in order of preference. The first compression technique in the list that is supported by the remote end of the channel is used.

The mutually supported compression techniques of the channel are passed to the message exit of the sending channel. The message exit can alter the compression technique on a per message basis. Compression alters the data passed to send and receive exits.

**NONE**

No header data compression is performed.

**SYSTEM**

Header data compression is performed.

**COMPMSG**

The list of message data compression techniques supported by the channel.

For SDR, SVR, CLUSSDR, CLUSRCVR, and CLNTCONN channels, the values must be specified in order of preference. The first compression technique in the list that is supported by the remote end of the channel is used.

The mutually supported compression techniques of the channel are passed to the message exit of the sending channel. The message exit can alter the compression technique on a per message basis. Compression alters the data passed to send and receive exits.

**NONE**


No message data compression is performed.

**RLE**

Message data compression is performed using run-length encoding.

**ZLIBFAST**

Message data compression is performed using ZLIB encoding with speed prioritized.

 On z/OS systems with [zEDC Express](#) facility enabled, compression can be offloaded to zEDC Express.

**ZLIBHIGH**

Message data compression is performed using ZLIB encoding with compression prioritized.


**ANY**


Any compression technique supported by the queue manager can be used. This value is only valid for RCVR, RQSTR, and SVRCONN channels.

**CONNNAME(string <, string >)**


Connection name.


For CLUSRCVR channels, **CONNNAME** relates to the local queue manager, and for other channels it relates to the target queue manager.


 On CLUSRCVR channels, when using automatic cluster setup, this parameter can use any variable configured at queue manager create time surrounded by +; for example +CONNNAME+.

 See the [crtmqm -iv](#) option for more information.

**Note:** When using these inserts, both the unexpanded inserts and the expanded values must fit inside the field maximum size.

 On z/OS, **CONNNAME** is mandatory for CLUSRCVR channels. In addition, whether you specify **CONNNAME**, or the name is generated for you, the **CONNNAME** produced must be a valid connection name for the local queue manager, otherwise the full repository is not able to make a connection back to the local queue manager.

 On z/OS, the maximum length of the string is 48 characters.

 On [Multiplatforms](#), the maximum length of the string is 264 characters

A workaround to the 48 character limit might be one of the following suggestions:

- Set up your DNS servers so that you use, for example, host name of `myserver` instead of `myserver.location.company.com`, ensuring you can use the short host name.
- Use IP addresses.

Specify **CONNAME** as a comma-separated list of names of machines for the stated **TRPTYPE**. Typically only one machine name is required. You can provide multiple machine names to configure multiple connections with the same properties. The connections are usually tried in the order they are specified in the connection list until a connection is successfully established. The order is modified for clients if the **CLNTWGHT** attribute is provided. If no connection is successful, the channel attempts the connection again, as determined by the attributes of the channel. With client channels, a connection-list provides an alternative to using queue manager groups to configure multiple connections. With message channels, a connection list is used to configure connections to the alternative addresses of a multi-instance queue manager.

**CONNAME** is required for channels with a channel type (**CHLTYPE**) of SDR, RQSTR, CLNTCONN, and CLUSSDR. It is optional for SVR channels, and for CLUSRCVR channels of **TRPTYPE(TCP)**, and is not valid for RCVR or SVRCONN channels.

**Multi** On [Multiplatforms](#), the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

```
(1415)
```

The generated **CONNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

**Tip:** If you are using any of the special characters in your connection name (for example, parentheses) you must enclose the string in single quotation marks.

The value you specify depends on the transport type (**TRPTYPE**) to be used:

## LU62

- **z/OS** On z/OS, there are two forms in which to specify the value:

### Logical unit name

The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. Logical unit name can be specified in one of three forms:

Form	Example
<b>luname</b>	IGY12355
<b>luname/TPname</b>	IGY12345/APING
<b>luname/TPname/modename</b>	IGY12345/APINGD/#INTER

For the first form, the TP name and mode name must be specified for the **TPNAME** and **MODENAME** parameters; otherwise these parameters must be blank.

**Note:** For CLNTCONN channels, only the first form is allowed.

### Symbolic name

The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The **TPNAME** and **MODENAME** parameters must be blank.

**Note:** For CLUSRCVR channels, the side information is on the other queue managers in the cluster. Alternatively, it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

The specified or implied LU name can be that of a VTAM generic resources group.

- **Multi** On IBM i, AIX, Linux, and Windows, **CONNAME** is the name of the CPI-C communications side object. Alternatively, if the **TPNAME** is not blank, **CONNAME** is the fully qualified name of the partner logical unit. See [Configuration parameters for an LU 6.2 connection](#).

### NetBIOS

A unique NetBIOS name (limited to 16 characters).

### SPX

The 4-byte network address, the 6-byte node address, and the 2-byte socket number. These values must be entered in hexadecimal, with a period separating the network and node addresses. The socket number must be enclosed in brackets, for example:

```
CONNAME('0a0b0c0d.804abcde23a1(5e86)')
```

### TCP

Either the host name, or the network address of the remote machine (or the local machine for CLUSRCVR channels). This address can be followed by an optional port number, enclosed in parentheses.

If the **CONNAME** is a host name, the host name is resolved to an IP address.

The IP stack used for communication depends on the value specified for **CONNAME** and the value specified for **LOCLADDR**. See [LOCLADDR](#) for information about how this value is resolved.

**z/OS** On z/OS, the connection name can include the IP\_name of an z/OS dynamic DNS group or a Network Dispatcher input port. Do not include the IP\_name or input port for channels with a channel type (**CHLTYPE**) of CLUSSDR.

On all platforms, you do not always need to specify the network address of your queue manager. If you define a channel with a channel type (**CHLTYPE**) of CLUSRCVR that is using TCP/IP, IBM MQ generates a **CONNAME** for you. It assumes the default port and uses the current IPv4 address of the system. If the system does not have an IPv4 address, the current IPv6 address of the system is used.

**Note:** If you are using clustering between IPv6-only and IPv4-only queue managers, do not specify an IPv6 network address as the **CONNAME** for CLUSRCVR channels. A queue manager that is capable only of IPv4 communication is unable to start a CLUSSDR channel definition that specifies the **CONNAME** in IPv6 hexadecimal form. Consider, instead, using host names in a heterogeneous IP environment.

### CONVERT

Specifies whether the sending message channel agent attempts conversion of the application message data, if the receiving message channel agent cannot perform this conversion.

#### NO

No conversion by sender

#### YES

Conversion by sender

**z/OS** On z/OS, N and Y are accepted as synonyms of NO and YES.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, CLUSSDR, or CLUSRCVR.

### DEFCDISP

Specifies the default channel disposition of the channel.

**PRIVATE**

The intended disposition of the channel is as a private channel.

**FIXSHARED**

The intended disposition of the channel is as a shared channel associated with a specific queue manager.

**SHARED**

The intended disposition of the channel is as a shared channel.

This parameter does not apply to channels with a channel type (**CHLTYPE**) of CLNTCONN, CLUSSDR, or CLUSRCVR.

**DEFRECON**

Specifies whether a client connection automatically reconnects a client application if its connection breaks.

**NO (default)**

Unless overridden by **MQCONN**, the client is not reconnected automatically.

**YES**

Unless overridden by **MQCONN**, the client reconnects automatically.

**QMGR**

Unless overridden by **MQCONN**, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as MQCNO\_RECONNECT\_Q\_MGR.

**DISABLED**

Reconnection is disabled, even if requested by the client program using the **MQCONN** MQI call.

*Table 150. Automatic reconnection depends on the values set in the application and in the channel definition*

DEFRECON	Reconnection options set in the application			
	MQCNO_RECONNECT	MQCNO_RECONNECT_Q_MGR	MQCNO_RECONNECT_AS_DEF	MQCNO_RECONNECT_DISABLED
NO (default)	YES	QMGR	NO	NO
YES	YES	QMGR	YES	NO
QMGR	YES	QMGR	QMGR	NO
DISABLED	NO	NO	NO	NO

**DESCR(string)**

Plain-text comment. It provides descriptive information about the channel when an operator issues the **DISPLAY CHANNEL** command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If the information is sent to another queue manager they might be translated incorrectly. The characters must be in the coded character set identifier (CCSID) of the local queue manager.

**DISCINT(integer)**

The minimum time in seconds for which the channel waits for a message to arrive on the transmission queue. The waiting period starts after a batch ends. After the end of the waiting period, if there are no more messages, the channel is ended. A value of zero causes the message channel agent to wait indefinitely.

The value must be in the range 0 - 999 999.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SVRCONN, SDR, SVR, CLUSSDR, CLUSRCVR.

For SVRCONN channels using the TCP protocol, **DISCINT** has a different interpretation. It is the minimum time in seconds for which the SVRCONN instance remains active without any communication from its partner client. A value of zero disables this disconnect processing. The SVRCONN inactivity interval applies only between IBM MQ API calls from a client, so no client is disconnected during an extended MQGET with wait call. This attribute is ignored for SVRCONN channels using protocols other than TCP.

### **HBINT(integer)**

**HBINT** specifies the approximate time between heartbeat flows sent by a message channel agent (MCA). The flows are sent when there are no messages on the transmission queue.

Heartbeat flows unblock the receiving MCA, which is waiting for messages to arrive or for the disconnect interval to expire. When the receiving MCA is unblocked, it can disconnect the channel without waiting for the disconnect interval to expire. Heartbeat flows also free any storage buffers that are allocated for large messages. They also close any queues that are left open at the receiving end of the channel.

The value is in seconds and must be in the range 0 - 999999. A value of zero means that no heartbeat flows are to be sent. The default value is 300. To be most useful, the value needs to be less than the disconnect interval value.

For SVRCONN and CLNTCONN channels, heartbeats can flow from both the server side as well as the client side independently. If no data is transferred across the channel during the heartbeat interval, the CLNTCONN MQI agent sends a heartbeat flow. The SVRCONN MQI agent responds to it with another heartbeat flow. The flows happen irrespective of the state of the channel. For example, irrespective of whether it is inactive while making an API call, or is inactive waiting for client user input. The SVRCONN MQI agent is also capable of initiating a heartbeat to the client, again irrespective of the state of the channel. The SVRCONN and CLNTCONN MQI agents are prevented from heart beating to each other at the same time. The server heartbeat is flowed if no data is transferred across the channel for the heartbeat interval plus 5 seconds.

For more information, see [Heartbeat interval \(HBINT\)](#).

### **KAINT(integer)**

The value passed to the communications stack for keepalive timing for this channel.

For this attribute to be effective, TCP/IP keepalive must be enabled both in the queue manager and in TCP/IP.

**z/OS** On z/OS, enable TCP/IP keepalive in the queue manager by issuing the **ALTER QMGR TCPKEEP(YES)** command. If the **TCPKEEP** queue manager parameter is NO, the value is ignored, and the keepalive facility is not used.

**Multi** On [Multiplatforms](#), TCP/IP keepalive is enabled when the **KEEPALIVE=YES** parameter is specified in the TCP stanza. Modify the TCP stanza in the distributed queuing configuration file, `qm.ini`, or through the IBM MQ Explorer.

Keepalive must also be enabled within TCP/IP itself. Refer to your TCP/IP documentation for information about configuring keepalive:

- **AIX** On AIX, use the **no** command.
- **Windows** On Windows, edit the registry.
- **z/OS** On z/OS, update your TCP/IP PROFILE data set and add or change the **INTERVAL** parameter in the TCPCONFIG section.

**z/OS** Although the **KAINT** parameter is available on all platforms, its setting is implemented only on z/OS.



**Multi** On Multiplatforms, you can access and modify the parameter, but there is no functional implementation of the parameter, it is only stored and forwarded. This functionality is useful in a clustered environment where a value set in a cluster-receiver channel definition on AIX, for example, flows to (and is implemented by) z/OS queue managers that are in, or join, the cluster. On Multiplatforms, if you need the functionality provided by the **KAINT** parameter, use the Heartbeat Interval (**HBINT**) parameter, as described in HBINT.

**(integer)**

The KeepAlive interval to be used, in seconds, in the range 1 through 99999.

**0**

The value used is that specified by the **INTERVAL** statement in the TCP profile configuration data set.

**AUTO**

The KeepAlive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated **HBINT** is greater than zero, keepalive interval is set to that value plus 60 seconds.
- If the negotiated **HBINT** is zero, the keepalive value used is that specified by the **INTERVAL** statement in the TCP/IP PROFILE configuration data set.

If **AUTO** is specified for **KAINT**, and it is a server-connection channel, the **TCP INTERVAL** value is used instead for the keepalive interval.

In this case, **KAINT** is zero in **DISPLAY CHSTATUS**; it would be non-zero if an integer had been coded instead of **AUTO**.

This parameter is valid for all channel types. It is ignored for channels with a **TRPTYPE** other than TCP or SPX.

**LIKE(channel-name)**

The name of a channel. The parameters of this channel are used to model this definition.

If you do not set **LIKE**, and do not set a parameter field related to the command, its value is taken from one of the default channels. The default values depend upon the channel type:

**SYSTEM.DEF.SENDER**

Sender channel

**SYSTEM.DEF.SERVER**

Server channel

**SYSTEM.DEF.RECEIVER**

Receiver channel

**SYSTEM.DEF.REQUESTER**

Requester channel

**SYSTEM.DEF.SVRCONN**

Server-connection channel

**SYSTEM.DEF.CLNTCONN**

Client-connection channel

**SYSTEM.DEF.CLUSSDR**

CLUSSDR channel

**SYSTEM.DEF.CLUSRCVR**

Cluster-receiver channel

**SYSTEM.DEF.AMQP**

AMQP channel

This parameter is equivalent to defining the following object for a SDR channel, and similarly for other channel types:

```
LIKE(SYSTEM.DEF.SENDER)
```

These default channel definitions can be altered by the installation to the default values required.

**z/OS** On z/OS, the queue manager searches page set zero for an object with the name you specify and a disposition of QMGR or COPY. The disposition of the **LIKE** object is not copied to the object and channel type you are defining.

**Note:**

1. **QSGDISP (GROUP)** objects are not searched.
2. **LIKE** is ignored if **QSGDISP (COPY)** is specified. However, the group object defined is used as a **LIKE** object.

**LOCLADDR(string)**

**LOCLADDR** is the local communications address for the channel. For channels other than AMQP channels, use this parameter if you want a channel to use a particular IP address, port, or port range for outbound communications. **LOCLADDR** might be useful in recovery scenarios where a channel is restarted on a different TCP/IP stack. **LOCLADDR** is also useful to force a channel to use an IPv4 or IPv6 stack on a dual-stack system. You can also use **LOCLADDR** to force a channel to use a dual-mode stack on a single-stack system.

**Note:** AMQP channels do not support the same format of **LOCLADDR** as other IBM MQ channels. For the format supported by AMQ, see the next parameter **AMQP: LOCLADDR**.

For channels other than AMQP channels, the **LOCLADDR** parameter is valid only for channels with a transport type (**TRPTYPE**) of TCP. If **TRPTYPE** is not TCP, the data is ignored and no error message is issued.

The value is the optional IP address, and optional port or port range used for outbound TCP/IP communications. The format for this information is as follows:

```
LOCLADDR([ip-addr] [(low-port[, high-port])] [, [ip-addr] [(low-port[, high-port])]])
```

The maximum length of **LOCLADDR**, including multiple addresses, is MQ\_LOCAL\_ADDRESS\_LENGTH.

If you omit **LOCLADDR**, a local address is automatically allocated.

Note, that you can set **LOCLADDR** for a C client using the Client Channel Definition Table (CCDT).

All the parameters are optional. Omitting the *ip-addr* part of the address is useful to enable the configuration of a fixed port number for an IP firewall. Omitting the port number is useful to select a particular network adapter without having to identify a unique local port number. The TCP/IP stack generates a unique port number.

Specify *[, [ip-addr] [(low-port[, high-port])]]* multiple times for each additional local address. Use multiple local addresses if you want to specify a specific subset of local network adapters. You can also use *[, [ip-addr] [(low-port[, high-port])]]* to represent a particular local network address on different servers that are part of a multi-instance queue manager configuration.

**ip-addr**

*ip-addr* is specified in one of three forms:

**IPv4 dotted decimal**

For example, 192.0.2.1

**IPv6 hexadecimal notation**

For example, 2001:DB8:0:0:0:0:0:0

**Alphanumeric host name form**

For example WWW.EXAMPLE.COM

**low-port and high-port**

*low-port* and *high-port* are port numbers enclosed in parentheses.

The following table shows how the **LOCLADDR** parameter can be used:

<i>Table 151. Examples of how the <b>LOCLADDR</b> parameter can be used</i>	
<b>LOCLADDR</b>	<b>Meaning</b>
9.20.4.98	Channel binds to this address locally
9.20.4.98, 9.20.4.99	Channel binds to either IP address. The address might be two network adapters on one server, or a different network adapter on two different servers in a multi-instance configuration.
9.20.4.98(1000)	Channel binds to this address and port 1000 locally
9.20.4.98(1000,2000)	Channel binds to this address and uses a port in the range 1000 - 2000 locally
(1000)	Channel binds to port 1000 locally
(1000,2000)	Channel binds to port in range 1000 - 2000 locally

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, OR CLUSRCVR.

On CLUSSDR channels, the IP address and port to which the outbound channel binds, is a combination of fields. It is a concatenation of the IP address, as defined in the **LOCLADDR** parameter, and the port range from the cluster cache. If there is no port range in the cache, the port range defined in the **LOCLADDR** parameter is used.

**z/OS** This port range does not apply to z/OS systems.

Even though this parameter is similar in form to **CONNAME**, it must not be confused with it. The **LOCLADDR** parameter specifies the characteristics of the local communications, whereas the **CONNAME** parameter specifies how to reach a remote queue manager.

When a channel is started, the values specified for **CONNAME** and **LOCLADDR** determine the IP stack to be used for communication; see [Table 3](#) and [Local Address \( \*\*LOCLADDR\*\* \)](#).

If the TCP/IP stack for the local address is not installed or configured, the channel does not start and an exception message is generated.

**z/OS** For example, on z/OS systems, the message is "CSQO015E: Command issued but no reply received." The message indicates that the connect() request specifies an interface address that is not known on the default IP stack. To direct the connect() request to the alternative stack, specify the **LOCLADDR** parameter in the channel definition as either an interface on the alternative stack, or a DNS host name. The same specification also works for listeners that might not use the default stack. To find the value to code for **LOCLADDR**, run the **NETSTAT HOME** command on the IP stacks that you want to use as alternatives.

Table 152. How the IP stack to be used for communication is determined

Protocols supported	CONNAME	LOCLADDR	Action of channel
IPv4 only	IPv4 address <sup>1</sup>		Channel binds to IPv4 stack
	IPv6 address <sup>2</sup>		Channel fails to resolve <b>CONNAME</b>
	IPv4 and 6 host name <sup>3</sup>		Channel binds to IPv4 stack
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve <b>CONNAME</b>
	IPv4 and 6 host name	IPv4 address	Channel binds to IPv4 stack
	Any address <sup>4</sup>	IPv6 address	Channel fails to resolve <b>LOCLADDR</b>
	IPv4 address	IPv4 and 6 host name	Channel binds to IPv4 stack
	IPv6 address	IPv4 and 6 host name	Channel fails to resolve <b>CONNAME</b>
	IPv4 and 6 host name	IPv4 and 6 host name	Channel binds to IPv4 stack
IPv4 and IPv6	IPv4 address		Channel binds to IPv4 stack
	IPv6 address		Channel binds to IPv6 stack
	IPv4 and 6 host name		Channel binds to stack determined by <b>IPADDRV</b>
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve <b>CONNAME</b>
	IPv4 and 6 host name	IPv4 address	Channel binds to IPv4 stack
	IPv4 address	IPv6 address	Channel maps <b>CONNAME</b> to IPv6 <sup>5</sup>
	IPv6 address	IPv6 address	Channel binds IPv6 stack
	IPv4 and 6 host name	IPv6 address	Channel binds IPv6 stack
	IPv4 address	IPv4 and 6 host name	Channel binds to IPv4 stack
	IPv6 address	IPv4 and 6 host name	Channel binds to IPv6 stack
IPv4 and 6 host name	IPv4 and 6 host name	Channel binds to stack determined by <b>IPADDRV</b>	

Table 152. How the IP stack to be used for communication is determined (continued)

Protocols supported	CONNAME	LOCLADDR	Action of channel
IPv6 only	IPv4 address		Channel maps <b>CONNAME</b> to IPv6 <sup>5</sup>
	IPv6 address		Channel binds to IPv6 stack
	IPv4 and 6 host name		Channel binds to IPv6 stack
	Any address	IPv4 address	Channel fails to resolve <b>LOCLADDR</b>
	IPv4 address	IPv6 address	Channel maps <b>CONNAME</b> to IPv6 <sup>5</sup>
	IPv6 address	IPv6 address	Channel binds to IPv6 stack
	IPv4 and 6 host name	IPv6 address	Channel binds to IPv6 stack
	IPv4 address	IPv4 and 6 host name	Channel maps <b>CONNAME</b> to IPv6 <sup>5</sup>
	IPv6 address	IPv4 and 6 host name	Channel binds to IPv6 stack
	IPv4 and 6 host name	IPv4 and 6 host name	Channel binds to IPv6 stack

**Notes:**

1. IPv4 address. An IPv4 host name that resolves only to an IPv4 network address or a specific dotted notation IPv4 address, for example 1 . 2 . 3 . 4. This note applies to all occurrences of ' IPv4 address' in this table.
2. IPv6 address. An IPv6 host name that resolves only to an IPv6 network address or a specific hexadecimal notation IPv6 address, for example 4321 : 54bc. This note applies to all occurrences of ' IPv6 address' in this table.
3. IPv4 and 6 host name. A host name that resolves to both IPv4 and IPv6 network addresses. This note applies to all occurrences of ' IPv4 and 6 host name' in this table.
4. Any address. IPv4 address, IPv6 address, or IPv4 and 6 host name. This note applies to all occurrences of 'Any address' in this table.
5. Maps IPv4 **CONNAME** to IPv4 mapped IPv6 address. IPv6 stack implementations that do not support IPv4 mapped IPv6 addressing fail to resolve the **CONNAME**. Mapped addresses might require protocol translators in order to be used. The use of mapped addresses is not recommended.

**AMQP: LOCLADDR(ip-addr)**

**Note:** For the format of **LOCLADDR** that other IBM MQ channels use, see the previous parameter **LOCLADDR**.

For AMQP channels, **LOCLADDR** is the local communications address for the channel. Use this parameter if you want to force the client to use a particular IP address. **LOCLADDR** is also useful to force a channel to use an IPv4 or IPv6 address if a choice is available, or to use a particular network adapter on a system with multiple network adapters.

The maximum length of **LOCLADDR** is MQ\_LOCAL\_ADDRESS\_LENGTH.

If you omit **LOCLADDR**, a local address is automatically allocated.

**ip-addr**

ip-addr is a single network address, specified in one of three forms:

**IPv4 dotted decimal**

For example, 192.0.2.1

**IPv6 hexadecimal notation**

For example, 2001:DB8:0:0:0:0:0:0

**Alphanumeric host name form**

For example, WWW.EXAMPLE.COM

If an IP address is entered, only the address format is validated. The IP address itself is not validated.

**LONGRTY(integer)**

The **LONGRTY** parameter specifies the maximum number of further attempts that are made by a SDR, SVR, or CLUSSDR channel to connect to a remote queue manager. The interval between attempts is specified by **LONGTMR**. The **LONGRTY** parameter takes effect if the count specified by **SHORTRTY** is exhausted.

If this count is exhausted without success, an error is logged to the operator, and the channel stops. In this circumstance, the channel must be restarted with a command. It is not started automatically by the channel initiator.

The **LONGRTY** value must be in the range 0 - 9999999.

This parameter is valid only for channels with a channel type ( **CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

A channel attempts to reconnect if it fails to connect initially, whether it is started automatically by the channel initiator or by an explicit command. It also tries to connect again if the connection fails after the channel successfully connecting. If the cause of the failure is such that more attempts are unlikely to be successful, they are not attempted.

**LONGTMR(integer)**

For **LONGRTY**, **LONGTMR** is the maximum number of seconds to wait before reattempting connection to the remote queue manager.

The time is approximate; zero means that another connection attempt is made as soon as possible.

The interval between attempting to reconnect might be extended if the channel has to wait to become active.

The **LONGTMR** value must be in the range 0 - 9999999.

**Note:** For implementation reasons, the maximum **LONGTMR** value is 999,999; values exceeding this maximum are treated as 999,999. Similarly, the minimum interval between attempting to reconnect is 2 seconds. Values less than this minimum are treated as 2 seconds.

This parameter is valid only for channels with a channel type ( **CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

**MAXINST(integer)**

The maximum number of simultaneous instances of an individual SVRCONN channel or AMQP channel that can be started.

The value must be in the range 0 - 999999999.

A value of zero prevents all client access on this channel.

New instances of SVRCONN channels cannot start if the number of running instances equals or exceeds the value of this parameter. If **MAXINST** is changed to less than the number of instances of the SVRCONN channel that are currently running, the number of running instances is not affected.

If an AMQP client attempts to connect to an AMQP channel, and the number of connected clients has reached **MAXINST**, the channel closes the connection with a close frame. The close frame contains the following message: `amqp:resource-limit-exceeded`. If a client connects with an ID that is already connected (that is, it performs a client-takeover), and the client is permitted to take over the

connection, the takeover will succeed regardless of whether the number of connected clients has reached **MAXINST**.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SVRCONN or AMQP.

### **MAXINSTC(integer)**

The maximum number of simultaneous individual SVRCONN channels that can be started from a single client. In this context, connections that originate from the same remote network address are regarded as coming from the same client.

The value must be in the range 0 - 999999999.

A value of zero prevents all client access on this channel.

If you reduce the value of **MAXINSTC** to less than the number of instances of the SVRCONN channel that is currently running from an individual client, the running instances are not affected. New SVRCONN instances from that client cannot start until the client is running fewer instances than the value of **MAXINSTC**.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SVRCONN.

### **MAXMSGL(integer)**

Specifies the maximum message length that can be transmitted on the channel. This parameter is compared with the value for the partner and the actual maximum used is the lower of the two values. The value is ineffective if the MQCB function is being executed and the channel type (**CHLTYPE**) is SVRCONN.

The value zero means the maximum message length for the queue manager; see [ALTER QMGR MAXMSGL](#).

**Multi** On [Multiplatforms](#), specify a value in the range zero to the maximum message length for the queue manager.

**z/OS** On z/OS, specify a value in the range 0 - 104857600 bytes (100 MB).

Note that by adding the digital signature and key to the message, [Advanced Message Security](#) increases the length of the message.

### **MCANAME(string)**

Message channel agent name.

This parameter is reserved, and if specified must be set to blanks (maximum length 20 characters).

### **MCATYPE**

Specifies whether the message-channel-agent program on an outbound message channel runs as a thread or a process.

#### **PROCESS**

The message channel agent runs as a separate process.

#### **THREAD**

The message channel agent runs as a separate thread

In situations where a threaded listener is required to service many incoming requests, resources can become strained. In this case, use multiple listener processes and target incoming requests at specific listeners though the port number specified on the listener.

**Multi** On [Multiplatforms](#), this parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RQSTR, CLUSSDR, or CLUSRCVR.

**z/OS** On z/OS, this parameter is supported only for channels with a channel type of CLUSRCVR. When specified in a CLUSRCVR definition, **MCATYPE** is used by a remote machine to determine the corresponding CLUSSDR definition.

### **MCAUSER(string)**

Message channel agent user identifier.

**Note:** An alternative way of providing a user ID for a channel to run under is to use channel authentication records. With channel authentication records, different connections can use the same channel while using different credentials. If both **MCAUSER** on the channel is set and channel authentication records are used to apply to the same channel, the channel authentication records take precedence. The **MCAUSER** on the channel definition is only used if the channel authentication record uses **USERSRC (CHANNEL)**. For more details, see [Channel authentication records](#)

This parameter interacts with **PUTAUT**, see [PUTAUT](#).

If **MCAUSER** is nonblank, a user identifier is used by the message channel agent for authorization to access IBM MQ resources. If **PUTAUT** is DEF, authorization includes authorization to put the message to the destination queue for RCVR or RQSTR channels.

If it is blank, the message channel agent uses its default user identifier.

The default user identifier is derived from the user ID that started the receiving channel. The possible values are:

#### **z/OS**

The user ID assigned to the channel-initiator started task by the z/OS started-procedures table.

#### **TCP/IP, Multiplatforms**

The user ID from the `inetd.conf` entry, or the user that started the listener.

#### **SNA, Multiplatforms**

The user ID from the SNA server entry. In the absence of the user ID from the SNA server entry, the user from the incoming attach request, or the user that started the listener.

#### **NetBIOS or SPX**

The user ID that started the listener.

The maximum length of the string is:

- **Windows** 64 characters on Windows.

For channels with a **CHLTYPE** of AMQP, before IBM MQ 9.2.0, the MCAUSER user ID setting is only supported for user IDs up to 12 characters in length. From IBM MQ 9.2.0, the 12 character limit is removed.

- 12 characters on platforms other than Windows.

**Windows** On Windows, you can optionally qualify a user identifier with the domain name in the format `user@domain`.

This parameter is not valid for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLNTCONN, CLUSSDR.

#### **MODENAME(string)**

LU 6.2 mode name (maximum length 8 characters).

This parameter is valid only for channels with a transport type (**TRPTYPE**) of LU62. If **TRPTYPE** is not LU62, the data is ignored and no error message is issued.

If specified, this parameter must be set to the SNA mode name unless the **CONNAME** contains a side-object name. If **CONNAME** is a side-object name it must be set to blanks. The actual name is then taken from the CPI-C Communications Side Object, or APPC side information data set, see [Configuration parameters for an LU 6.2 connection](#).

This parameter is not valid for channels with a channel type (**CHLTYPE**) of RCVR or SVRCONN.

#### **MONCHL**

Controls the collection of online monitoring data for channels:

##### **QMGR**

Collect monitoring data according to the setting of the queue manager parameter **MONCHL**.

##### **OFF**

Monitoring data collection is turned off for this channel.



**LOW**

If the value of the queue manager **MONCHL** parameter is not NONE, online monitoring data is turned on. Data is collected at a low rate for this channel.

**MEDIUM**

If the value of the queue manager **MONCHL** parameter is not NONE, online monitoring data is turned on. Data is collected at a medium rate for this channel.

**HIGH**

If the value of the queue manager **MONCHL** parameter is not NONE, online monitoring data is turned on. Data is collected at a high rate for this channel.

Changes to this parameter take effect only on channels started after the change occurs.

For cluster channels, the value of this parameter is not replicated in the repository and, therefore, not used in the auto-definition of CLUSSDR channels. For auto-defined CLUSSDR channels, the value of this parameter is taken from the queue manager attribute **MONACLS**. This value might then be overridden in the channel auto-definition exit.

**MRDATA(string)**

Channel message-retry exit user data. The maximum length is 32 characters.

This parameter is passed to the channel message-retry exit when it is called.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**MREXIT(string)**

Channel message-retry exit name.

The format and maximum length of the name is the same as for **MSGEXIT**, however you can specify only one message-retry exit.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**MRRTY(integer)**

The number of times the channel tries again before it decides it cannot deliver the message.

This parameter controls the action of the MCA only if the message-retry exit name is blank. If the exit name is not blank, the value of **MRRTY** is passed to the exit to use. The number of attempts to redeliver the message is controlled by the exit, and not by this parameter.

The value must be in the range 0 - 999999999. A value of zero means that no attempts to redeliver the message are tried.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**MRTMR(integer)**

The minimum interval of time that must pass before the channel can try the MQPUT operation again. The time interval is in milliseconds.

This parameter controls the action of the MCA only if the message-retry exit name is blank. If the exit name is not blank, the value of **MRTMR** is passed to the exit to use. The number of attempts to redeliver the message is controlled by the exit, and not by this parameter.

The value must be in the range 0 - 999999999. A value of zero means that if the value of **MRRTY** is greater than zero, the channel reattempts delivery as soon as possible.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

**MSGDATA(string)**

User data for the channel message exit. The maximum length is 32 characters.

This data is passed to the channel message exit when it is called.

**ALW** On AIX, Linux, and Windows, you can specify data for more than one exit program by specifying multiple strings separated by commas. The total length of the field must not exceed 999 characters.

**IBM i** On IBM i, you can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

**z/OS** On z/OS, you can specify up to eight strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

On other platforms, you can specify only one string of message exit data for each channel.

**Note:** This parameter is accepted but ignored for SVRCONN and CLNTCONN channels.

### MSGEXIT(string)

Channel message exit name.

If **MSGEXIT** is nonblank the exit is called at the following times:

- Immediately after a SDR or SVR channel retrieves a message from the transmission queue.
- Immediately before a RQSTR channel puts a message on destination queue.
- When the channel is initialized or ended.





The exit is passed the entire application message and transmission queue header for modification.

**MSGEXIT** is accepted and ignored by CLNTCONN and SVRCONN channels. CLNTCONN or SVRCONN channels do not call message exits.

The format and maximum length of the exit name depends on the platform; see [Table 153 on page 490](#).

If the **MSGEXIT**, **MREXIT**, **SCYEXIT**, **SENDEXIT**, and **RCVEXIT** parameters are all left blank, the channel user exit is not invoked. If any of these parameters is nonblank, the channel exit program is called. You can enter text string for these parameters. The maximum length of the string is 128 characters.

Platform	Exit name format	Maximum length	Comment
<ul style="list-style-type: none"> <li>▶ Linux</li> <li>▶ AIX</li> <li>▶ Linux</li> <li>▶ AIX</li> </ul> AIX and Linux	<i>libraryname (functionname)</i>	128	You can specify the name of more than one exit program. Specify multiple strings separated by commas. However, the total number of characters specified must not exceed 999.
<ul style="list-style-type: none"> <li>▶ Windows</li> <li>▶ Windows</li> </ul> Windows	<i>dllname (functionname)</i>	128	1. You can specify the name of more than one exit program. Specify multiple strings separated by commas. However, the total number of characters specified must not exceed 999. 2. <i>dllname</i> is specified without the suffix (.DLL).

Table 153. Message exit format and length (continued)			
Platform	Exit name format	Maximum length	Comment
  IBM i	<i>progrname libname</i>	20	1. You can specify the names of up to 10 exit programs by specifying multiple strings separated by commas. 2. <i>program name</i> occupies the first 10 characters and <i>libname</i> the second 10 characters. If necessary, both fields are padded to the right with blanks.
  z/OS	<i>loadModuleName</i>	8	1. You can specify the names of up to eight exit programs by specifying multiple strings separated by commas. 2. 128 characters are allowed for exit names for CLNTCONN channels, subject to a maximum total length including commas of 999.

### NETPRTY(*integer*)

The priority for the network connection. Distributed queuing chooses the path with the highest priority if there are multiple paths available. The value must be in the range 0 - 9; 0 is the lowest priority.

This parameter is valid only for CLUSRCVR channels.

### NPMSPEED

The class of service for nonpersistent messages on this channel:

#### FAST

Fast delivery for nonpersistent messages; messages might be lost if the channel is lost. Messages are retrieved using MQGMO\_SYNCPOINT\_IF\_PERSISTENT and so are not included in the batch unit of work.

#### NORMAL

Normal delivery for nonpersistent messages.

If the value of **NPMSPEED** differs between the sender and receiver, or either one does not support it, NORMAL is used.

#### Notes:

1. If the active recovery logs for IBM MQ for z/OS are switching and archiving more frequently than expected, given that the messages being sent across a channel are non-persistent, setting NPMSPEED(FAST) on both the sending and receiving ends of the channel can minimize the SYSTEM.CHANNEL.SYNCQ updates.
2. If you are seeing high CPU usage relating to updates to the SYSTEM.CHANNEL.SYNCQ, setting NPMSPEED(FAST) can significantly reduce the CPU usage.

This parameter is valid only for channels with a **CHLTYPE** of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR.

### PASSWORD(*string*)

Password used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent. The maximum length is 12 characters.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RQSTR, CLNTCONN, or CLUSSDR.

 On z/OS, it is supported only for channels with a channel type (**CHLTYPE**) of CLNTCONN.

Although the maximum length of the parameter is 12 characters, only the first 10 characters are used.

**PORT(integer)**

The port number used to connect an AMQP channel. The default port for AMQP 1.0 connections is 5672. If you are already using port 5672, you can specify a different port.

**PROPCTL**

Property control attribute; see **PROPCTL** channel options.

**PROPCTL** specifies what happens to message properties when a message is sent to another queue manager; see

This parameter is applicable to SDR, SVR, CLUSSDR, and CLUSRCVR channels.

This parameter is optional.

Permitted values are:

**COMPAT**

COMPAT allows applications which expect JMS-related properties to be in an MQRFH2 header in the message data to continue to work unmodified.

*Table 154. Range of results, depending on which message properties are set, when PROPCTL value is COMPAT*

Message properties	Result
The message contains a property with a prefix of mcd., jms., usr. or mqext.	If the <b>Support</b> value is MQPD_SUPPORT_OPTIONAL, all optional message properties are placed in one or more MQRFH2 headers. This rule does not apply to properties in the message descriptor or extension, which remain in the same place. Optional message properties are moved into the message data before the message is sent to the remote queue manager.
The message does not contain a property with a prefix of mcd., jms., usr. or mqext.	All message properties, except properties in the message descriptor or extension, are removed from the message before the message is sent to the remote queue manager.
The message contains a property where the <b>Support</b> field of the property descriptor is not set to MQPD_SUPPORT_OPTIONAL	The message is rejected with reason MQRC_UNSUPPORTED_PROPERTY and treated in accordance with its report options.
The message contains one or more properties where the <b>Support</b> field of the property descriptor is set to MQPD_SUPPORT_OPTIONAL. Other fields of the property descriptor are set to non-default values.	The properties with non-default values are removed from the message before the message is sent to the remote queue manager.

**NONE**

All properties of the message, except properties in the message descriptor or extension, are removed from the message. The properties are removed before the message is sent to the remote queue manager.

If the message contains a property where the **Support** field of the property descriptor is not set to MQPD\_SUPPORT\_OPTIONAL then the message is rejected with reason MQRC\_UNSUPPORTED\_PROPERTY. The error is reported in accordance with the report options set in the message header.

**ALL**

All properties of the message are included with the message when it is sent to the remote queue manager. The properties, except properties in the message descriptor (or extension), are placed in one or more MQRFH2 headers in the message data.

## PUTAUT

**PUTAUT** specifies which user identifiers are used to establish authority for a channel. It specifies the user identifier to put messages to the destination queue using a message channel, or to run an MQI call using an MQI channel.

### DEF

The default user ID is used.

**z/OS** On z/OS, DEF might involve using both the user ID received from the network and that derived from **MCAUSER**.

### CTX

The user ID from the *UserIdentifier* field of the message descriptor is used.

**z/OS** On z/OS, CTX might involve also using the user ID received from the network or that derived from **MCAUSER**, or both.

### **z/OS** ONLYMCA

The user ID derived from MCAUSER is used. Any user ID received from the network is not used. This value is supported only on z/OS.

### **z/OS** ALTMCA

The user ID from the *UserIdentifier* field of the message descriptor is used. Any user ID received from the network is not used. This value is supported only on z/OS.

**z/OS** On z/OS, the user IDs that are checked, and how many user IDs are checked, depends on the setting of the MQADMIN RACF class h1q.RESLEVEL profile. Depending on the level of access the user ID of the channel initiator has to h1q.RESLEVEL, zero, one, or two user IDs are checked. To see how many user IDs are checked, see [RESLEVEL and the channel initiator connection](#). For more information about which user IDs are checked, see [User IDs used by the channel initiator](#).

**z/OS** On z/OS, this parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, CLUSRCVR, or SVRCONN. CTX and ALTMCA are not valid for SVRCONN channels.

**Multi** On [Multiplatforms](#), this parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, RQSTR, or CLUSRCVR.

## QMNAME(string)

Queue manager name.

For CLNTCONN channels, **QMNAME** is the name of a queue manager to which an IBM MQ MQI client application can request connection. **QMNAME** is not necessarily the same as the name of the queue manager on which the channel is defined; see [Queue manager groups in the CCDT](#).

For channels of other types, the **QMNAME** parameter is not valid.

### **z/OS** QSGDISP

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

Table 155. Object dispositions for **QSGDISP** options

QSGDISP	DEFINE
COPY	<p>The object is defined on the page set of the queue manager that executes the command. It uses the <b>QSGDISP (GROUP)</b> object of the same name as the LIKE object.</p> <p>For example, if you issue the following command,</p> <pre data-bbox="505 436 1468 499">DEFINE CHANNEL(<i>channel_name</i>) CHLTYPE(<i>channel_type</i>) REPLACE QSGDISP(COPY)</pre> <p>the queue manager searches the shared configuration repository for a CHANNEL definition for <i>channel_name</i> and <i>channel_type</i>. If a matching CHANNEL definition is found, the queue manager creates a local copy of this definition on the queue manager page set.</p> <p>For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.</p>
GROUP	<p>The object definition resides in the shared configuration repository. <b>QSGDISP (GROUP)</b> is allowed only if the queue manager is in a queue sharing group.</p> <p>If the DEFINE for the <b>QSGDISP (GROUP)</b> object is successful, the <code>DEFINE CHANNEL(<i>channel_name</i>) CHLTYPE(<i>channel_type</i>) REPLACE QSGDISP (COPY)</code> command is generated and sent to all active queue managers in the queue sharing group to make or refresh local copies on page set zero.</p> <p>The DEFINE for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
PRIVATE	Not permitted.
QMGR	The object is defined on the page set of the queue manager that executes the command.

**RCVDATA(string)**

Channel receive exit user data (maximum length 32 characters).

This parameter is passed to the channel receive exit when it is called.

**ALW** On AIX, Linux, and Windows, you can specify data for more than one exit program by specifying multiple strings separated by commas. The total length of the field must not exceed 999 characters.

**IBM i** On IBM i, you can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first receive exit specified, the second string to the second exit, and so on.

**z/OS** On z/OS, you can specify up to eight strings, each of length 32 characters. The first string of data is passed to the first receive exit specified, the second string to the second exit, and so on.

On other platforms, you can specify only one string of receive exit data for each channel.

**RCVEXIT(string)**

Channel receive exit name.

If this name is nonblank, the exit is called at the following times:

- Immediately before the received network data is processed.

The exit is given the complete transmission buffer as received. The contents of the buffer can be modified as required.

- At initialization and termination of the channel.

**ALW** On AIX, Linux, and Windows, you can specify the name of more than one exit program by specifying multiple strings separated by commas. However, the total number of characters specified must not exceed 999.

**IBM i** On IBM i, you can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

**z/OS** On z/OS, you can specify the names of up to eight exit programs by specifying multiple strings separated by commas.

On other platforms, you can specify only one receive exit name for each channel.

The format and maximum length of the name is the same as for **MSGEXIT**.

### **REPLACE and NOREPLACE**

Replace the existing definition with this one, or not. This parameter is optional.

**z/OS** On z/OS it must have the same disposition. Any object with a different disposition is not changed.

#### **REPLACE**

The definition replaces any existing definition of the same name. If a definition does not exist, one is created. **REPLACE** does not alter the channel status.

#### **NOREPLACE**

The definition does not replace any existing definition of the same name.

### **SCYDATA(string)**

Channel security exit user data (maximum length 32 characters).

This parameter is passed to the channel security exit when it is called.

### **SCYEXIT(string)**

Channel security exit name.

If this name is nonblank, the exit is called at the following times:

- Immediately after establishing a channel.

Before any messages are transferred, the exit is able to instigate security flows to validate connection authorization.

- Upon receipt of a response to a security message flow.

Any security message flows received from the remote processor on the remote queue manager are given to the exit.

- At initialization and termination of the channel.

The format and maximum length of the name is the same as for **MSGEXIT** but only one name is allowed.

### **SENDDATA(string)**

Channel send exit user data. The maximum length is 32 characters.

This parameter is passed to the channel send exit when it is called.

**ALW** On AIX, Linux, and Windows, you can specify data for more than one exit program by specifying multiple strings separated by commas. The total length of the field must not exceed 999 characters.

**IBM i** On IBM i, you can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first send exit specified, the second string to the second exit, and so on.

**z/OS** On z/OS, you can specify up to eight strings, each of length 32 characters. The first string of data is passed to the first send exit specified, the second string to the second exit, and so on.

On other platforms, you can specify only one string of send exit data for each channel.

### **SENDEXIT(string)**

Channel send exit name.

If this name is nonblank, the exit is called at the following times:

- Immediately before data is sent out on the network.

The exit is given the complete transmission buffer before it is transmitted. The contents of the buffer can be modified as required.

- At initialization and termination of the channel.

**ALW** On AIX, Linux, and Windows, you can specify the name of more than one exit program by specifying multiple strings separated by commas. However, the total number of characters specified must not exceed 999.

**IBM i** On IBM i, you can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

**z/OS** On z/OS, you can specify the names of up to eight exit programs by specifying multiple strings separated by commas.

On other platforms, you can specify only one send exit name for each channel.

The format and maximum length of the name is the same as for **MSGEXIT**.

### **SEQWRAP(integer)**

When this value is reached, sequence numbers wrap to start again at 1.

This value is nonnegotiable and must match in both the local and remote channel definitions.

The value must be in the range 100 - 999999999.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR.

### **SHARECNV(integer)**

Specifies the maximum number of conversations that can be sharing each TCP/IP channel instance. A **SHARECNV** value of:

**1**

Specifies no sharing of conversations over a TCP/IP channel instance. Client heart beating is available whether in an MQGET call or not. Read ahead and client asynchronous consumption are also available, and channel quiescing is more controllable.

**0**

Specifies no sharing of conversations over a TCP/IP channel instance.

The value must be in the range zero through 999999999.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of CLNTCONN or SVRCONN. If the CLNTCONN **SHARECNV** value does not match the SVRCONN **SHARECNV** value, the lower of the two values is used. This parameter is ignored for channels with a transport type (**TRPTYPE**) other than TCP.

All the conversations on a socket are received by the same thread.

High **SHARECNV** limits have the advantage of reducing queue manager thread usage. If many conversations sharing a socket are all busy, there is a possibility of delays. The conversations contend with one another to use the receiving thread. In this situation, a lower **SHARECNV** value is better.

The number of shared conversations does not contribute to the **MAXINST** or **MAXINSTC** totals.



**Note:** You should restart the client for this change to take effect.

### **SHORTRTY**(integer)

**SHORTRTY** specifies the maximum number of attempts that are made by a SDR, SVR, or CLUSSDR channel to connect to the remote queue manager, at intervals specified by **SHORTTMR**. After the number of attempts is exhausted, the channel tries to reconnect using to the schedule defined by **LONGRTY**.

The value must be in the range 0 - 999999999.

This parameter is valid only for channels with a channel type ( **CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

A channel attempts to reconnect if it fails to connect initially, whether it is started automatically by the channel initiator or by an explicit command. It also tries to connect again if the connection fails after the channel successfully connecting. If the cause of the failure is such that more attempts are unlikely to be successful, they are not attempted.

### **SHORTTMR**(integer)

For **SHORTRTY**, **SHORTTMR** is the maximum number of seconds to wait before reattempting connection to the remote queue manager.

The time is approximate. From IBM MQ 8.0, zero means that another connection attempt is made as soon as possible.

The interval between attempting to reconnect might be extended if the channel has to wait to become active.

The value must be in the range 0 - 999999999.

**Note:** For implementation reasons, the maximum **SHORTTMR** value is 999,999; values exceeding this maximum are treated as 999,999. From IBM MQ 8.0, if **SHORTTMR** is set to 1 then the minimum interval between attempting to connect is 2 seconds.

This parameter is valid only for channels with a channel type ( **CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

**z/OS**

### **SPLPROT**

**SPLPROT** (Security Policy Protection) specifies how a server-to-server Message Channel Agent should deal with message protection when AMS is active and an applicable policy exists.

This parameter applies to z/OS only, from IBM MQ 9.1.3 onwards.

The permitted values are:

#### **PASSTHRU**

Pass through, unchanged, any messages sent or received by the message channel agent for this channel.

This value is valid for channels with a channel type (**CHLTYPE**) of SDR, SVR, RCVR, or RQSTR, and is the default value.

#### **REMOVE**

Remove any AMS protection from messages retrieved from the transmission queue by the message channel agent, and send the messages to the partner.

When the message channel agent gets a message from the transmission queue, if an AMS policy is defined for the transmission queue, it is applied to remove any AMS protection from the message prior to sending the message across the channel. If an AMS policy is not defined for the transmission queue, the message is sent as is.

This value is valid only for channels with a channel type of SDR or SVR.

#### **ASPOLICY**

Based on the policy defined for the target queue, apply AMS protection to inbound messages prior to putting them on to the target queue.

When the message channel agent receives an inbound message, if an AMS policy is defined for the target queue, AMS protection is applied to the message prior to the message being put to the target queue. If an AMS policy is not defined for the target queue, the message is put to the target queue as is.

This value is valid only for channels with a channel type of RCVR or RQSTR.

## SSLCAUTH

**SSLCAUTH** defines whether IBM MQ requires a certificate from the TLS client. The TLS client is the initiating end of the channel. **SSLCAUTH** is applied to the TLS server, to determine the behavior required of the client. The TLS server is the end of the channel that receives the initiation flow.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of RCVR, SVRCONN, CLUSRCVR, SVR, OR RQSTR.

The parameter is used only for channels with **SSLCIPH** specified. If **SSLCIPH** is blank, the data is ignored and no error message is issued.

### REQUIRED

IBM MQ requires and validates a certificate from the TLS client.



### OPTIONAL

The peer TLS client system might still send a certificate. If it does, the contents of this certificate are validated as normal.

## SSLCIPH(string)



Specifies the CipherSpec that is used on the channel. The maximum length is 32 characters.



**Attention:**   On IBM MQ for z/OS, you can also specify the four digit hexadecimal code of a CipherSpec, whether or not it appears in the following table. On IBM i, you can also specify the two digit hexadecimal code of a CipherSpec, whether or not it appears in the following table. Also, on IBM i, installation of AC3 is a prerequisite for the use of TLS. You should not specify hexadecimal cipher values in SSLCIPH, because it is unclear from the value which cipher will be used, and the choice of which protocol to be used is indeterminate. Using hexadecimal cipher values can lead to CipherSpec mismatch errors.


The **SSLCIPH** values must specify the same CipherSpec on both ends of the channel.


This parameter is valid on all channel types that use transport type **TRPTYPE(TCP)**. If the parameter is blank, no attempt is made to use TLS on the channel.

  If SecureCommsOnly is enabled, plain text communication is not supported and the channel fails to start.

The value for this parameter is also used to set the value of SECPROT, which is an output field on the DISPLAY CHSTATUS command.

**Note:** When **SSLCIPH** is used with a telemetry channel, it means TLS Cipher Suite. See the **SSLCIPH** description for **DEFINE CHANNEL (MQTT)**.

 From IBM MQ 9.1.1, you can specify a value of ANY\_TLS12, which represents a subset of acceptable CipherSpecs that use the TLS 1.2 protocol; these CipherSpecs are listed in the following table.

 From IBM MQ 9.1.4, on AIX, Linux, and Windows, IBM MQ provides an expanded set of alias CipherSpecs that includes ANY\_TLS12\_OR\_HIGHER, and ANY\_TLS13\_OR\_HIGHER. These alias CipherSpecs are listed in the following table.



**Attention:** If your enterprise needs to guarantee that a certain CipherSpec is negotiated and used, you must not use an alias CipherSpec value such as ANY\_TLS12.

For information on changing your existing security configurations to use the ANY\_TLS12\_OR\_HIGHER CipherSpec, see Migrating existing security configurations to use the ANY\_TLS12\_OR\_HIGHER CipherSpec.

Table 156. CipherSpecs you can use with IBM MQ TLS support



Platform support "1" on page 501	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 501	Suite B
<b>Alias CipherSpecs</b>							
All	ANY_TLS13_OR_HIGHER "3" on page 501 "4" on page 501	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS13 "4" on page 501 "5" on page 501	N/A	TLS 1.3	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS12_OR_HIGHER "4" on page 501 "6" on page 501	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS12 "7" on page 501	N/A	TLS 1.2	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY "8" on page 501	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
<b>CipherSpecs for TLS 1.3</b>							
All	TLS_AES_128_GCM_SHA256	1301	TLS 1.3	GCM	AES-128 with GCM (128)	Yes	No
All	TLS_AES_256_GCM_SHA384	1302	TLS 1.3	GCM	AES-256 with GCM (256)	Yes	No
All	TLS_CHACHA20_POLY1305_SHA256	1303	TLS 1.3	POLY1305	CHACHA20 (256)	No	No
 ALW	TLS_AES_128_CCM_SHA256	1304	TLS 1.3	CBC-MAC	AES-128 with CTR (128)	Yes	No
 ALW	TLS_AES_128_CCM_8_SHA256 "10" on page 501	1305	TLS 1.3	CBC-MAC	AES-128 with CTR (128)	Yes	No
<b>CipherSpecs for TLS 1.2</b>							
All	TLS_RSA_WITH_AES_128_CBC_SHA256 "9" on page 501	003C	TLS 1.2	SHA-256	AES (128)	Yes	No
All	TLS_RSA_WITH_AES_256_CBC_SHA256 "9" on page 501 "11" on page 501	003D	TLS 1.2	SHA-256	AES (256)	Yes	No
All	TLS_RSA_WITH_AES_128_GCM_SHA256 "9" on page 501 "12" on page 501	009C	TLS 1.2	SHA-256 and AEAD GCM	AES (128)	Yes	No
All	TLS_RSA_WITH_AES_256_GCM_SHA384 "9" on page 501 "11" on page 501 "12" on page 501	009D	TLS 1.2	SHA-384 and AEAD GCM	AES (256)	Yes	No
All	ECDHE_ECDSA_AES_128_CBC_SHA256 "9" on page 501	C023	TLS 1.2	SHA-256	AES (128)	Yes	No

Table 156. CipherSpecs you can use with IBM MQ TLS support (continued)







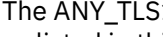



Platform support "1" on page 501	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 501	Suite B
All	ECDHE_ECDSA_AES_256_CBC_SHA384 <a href="#">"9" on page 501</a> <a href="#">"11" on page 501</a>	C024	TLS 1.2	SHA-384	AES (256)	Yes	No
All	ECDHE_RSA_AES_128_CBC_SHA256 <a href="#">"9" on page 501</a>	C027	TLS 1.2	SHA-256	AES (128)	Yes	No
All	ECDHE_RSA_AES_256_CBC_SHA384 <a href="#">"9" on page 501</a> <a href="#">"11" on page 501</a>	C028	TLS 1.2	SHA-384	AES (256)	Yes	No
 Multi	ECDHE_ECDSA_AES_128_GCM_SHA256 <a href="#">"11" on page 501</a> <a href="#">"12" on page 501</a>	C02B	TLS 1.2	SHA-256 and AEAD GCM	AES (SHA384)	Yes	128 bit
 Multi	ECDHE_ECDSA_AES_256_GCM_SHA384 <a href="#">"11" on page 501</a> <a href="#">"12" on page 501</a>	C02C	TLS 1.2	SHA-384 and AEAD GCM	AES (SHA384)	Yes	192 bit
All	ECDHE_RSA_AES_128_GCM_SHA256 <a href="#">"12" on page 501</a>	C02F	TLS 1.2	SHA-256 and AEAD GCM	AES (128)	Yes	No
All	ECDHE_RSA_AES_256_GCM_SHA384 <a href="#">"11" on page 501</a> <a href="#">"12" on page 501</a>	C030	TLS 1.2	AEAD AES-128 GCM	AES (SHA384)	Yes	No

Table 156. CipherSpecs you can use with IBM MQ TLS support (continued)

Platform support "1" on page 501	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 501	Suite B
----------------------------------	-----------------	----------	---------------	---------------	--	----------------------	---------

**Notes:**



1. For a list of platforms covered by each platform icon, see [Icons used in the product documentation](#).
2. Specifies whether the CipherSpec is FIPS-certified on a FIPS-certified platform. See [Federal Information Processing Standards \(FIPS\)](#) for an explanation of FIPS.
3.  The ANY\_TLS13\_OR\_HIGHER alias CipherSpec negotiates the highest level of security that the remote end will allow but will only connect using a TLS 1.3 or higher protocol.
4.  To use TLS 1.3, or the ANY CipherSpec, on IBM i the underlying operating system version must support TLS 1.3. See [System TLS support for TLSv1.3](#) for more information.
5.  The ANY\_TLS13 alias CipherSpec represents a subset of acceptable CipherSpecs that use the TLS 1.3 protocol, as listed in this table for each platform.
6.  The ANY\_TLS12\_OR\_HIGHER alias CipherSpec negotiates the highest level of security that the remote end will allow but will only connect using a TLS 1.2 or higher protocol.
7. The ANY\_TLS12 CipherSpec represents a subset of acceptable CipherSpecs that use the TLS 1.2 protocol, as listed in this table for each platform.
8.  The ANY alias CipherSpec negotiates the highest level of security that the remote end will allow.
9.  These CipherSpecs are not enabled on IBM i 7.4 systems that have System Value QSSLCSLCTL set to \*OPSSYS.
10.  These CipherSpecs use an 8-octet Integrity Check Value (ICV) instead of a 16-octet ICV.
11. This CipherSpec cannot be used to secure a connection from the IBM MQ Explorer to a queue manager unless the appropriate unrestricted policy files are applied to the JRE used by the Explorer.
12.  Following a recommendation by GSKit, TLS 1.2 GCM CipherSpecs have a restriction which means that after 2<sup>24.5</sup> TLS records are sent, using the same session key, the connection is terminated with message `AMQ9288E`. This GCM restriction is active, regardless of the FIPS mode being used.

To prevent this error from happening, avoid using TLS 1.2 GCM Ciphers, enable secret key reset, or start your IBM MQ queue manager or client with the environment variable `GSK_ENFORCE_GCM_RESTRICTION=GSK_FALSE` set. For GSKit libraries, you must set this environment variable on both sides of the connection, and apply it to both client to queue manager connections and queue manager to queue manager connections. Note that this setting affects unmanaged .NET clients, but not Java or managed .NET clients. For more information, see [AES-GCM cipher restriction](#).

This restriction does not apply to IBM MQ for z/OS.

For more information about CipherSpecs, see [Enabling CipherSpecs](#).

When you request a personal certificate, you specify a key size for the public and private key pair. The key size that is used during the SSL handshake can depend on the size stored in the certificate and on the CipherSpec:

-   On z/OS, AIX, Linux, and Windows, when a CipherSpec name includes `_EXPORT`, the maximum handshake key size is 512 bits. If either of the certificates exchanged

during the SSL handshake has a key size greater than 512 bits, a temporary 512-bit key is generated for use during the handshake.

- **z/OS** For z/OS, System SSL state that if a TLS V1.3 connection is being negotiated:
  - The minimum key size for an RSA peer certificate is the larger of the following two values: 2048, or the value specified in the GSK\_PEER\_RSA\_MIN\_KEY\_SIZE attribute.
  - The minimum key size for an ECC peer certificate is the larger of the following two values: 256, or the value specified in the GSK\_PEER\_ECC\_MIN\_KEY\_SIZE attribute.
- **ALW** On AIX, Linux, and Windows, when a CipherSpec name includes \_EXPORT1024, the handshake key size is 1024 bits.
- Otherwise the handshake key size is the size stored in the certificate.

### SSLPEER (string)

Specifies the certificate filter used by the peer queue manager or client at the other end of the channel. The filter is used to compare with the distinguished name of the certificate. A *distinguished name* is the identifier of the TLS certificate. If the distinguished name in the certificate received from the peer does not match the **SSLPEER** filter, the channel does not start.

**Note:** An alternative way of restricting connections into channels by matching against the TLS Subject distinguished name, is to use channel authentication records. With channel authentication records, different TLS subject distinguished name patterns can be applied to the same channel. Both **SSLPEER** and a channel authentication record can be applied to the same channel. If so, the inbound certificate must match both patterns in order to connect. For more information, see [Channel authentication records](#).

**SSLPEER** is optional. If it is not specified, the distinguished name of the peer is not checked at channel startup. The distinguished name from the certificate is still written into the **SSLPEER** definition held in memory, and passed to the security exit. If **SSLCIPH** is blank, the data is ignored and no error message is issued.

This parameter is valid for all channel types.

The **SSLPEER** value is specified in the standard form used to specify a distinguished name. For example:

```
SSLPEER( 'SERIALNUMBER=4C:D0:49:D5:02:5F:38,CN="H1_C_FR1",O=IBM,C=GB')
```

You can use a semi-colon as a separator instead of a comma.

The possible attribute types supported are:

Table 157. Attribute types supported by SSLPEER	
Attribute	Description
SERIALNUMBER	Certificate serial number
MAIL	Email address
<b>Deprecated</b> E	Email address (Deprecated in preference to MAIL)
UID or USERID	User identifier
CN	Common Name
T	Title
OU	Organizational Unit name
DC	Domain component

<i>Table 157. Attribute types supported by SSLPEER (continued)</i>	
<b>Attribute</b>	<b>Description</b>
O	Organization name
STREET	Street / First line of address
L	Locality name
ST (or SP or S)	State or Province name
PC	Postal code / zipcode
C	Country
UNSTRUCTUREDNAME	Host name
UNSTRUCTUREDADDRESS	IP address
DNQ	Distinguished name qualifier

IBM MQ accepts only uppercase letters for the attribute types.

If any of the unsupported attribute types are specified in the **SSLPEER** string, an error is output either when the attribute is defined, or at run time. When the error is output depends on which platform you are running on. An error implies that the **SSLPEER** string does not match the distinguished name of the flowed certificate.

If the distinguished name of the flowed certificate contains multiple organizational unit (OU) attributes, and **SSLPEER** specifies that these attributes are to be compared, they must be defined in descending hierarchical order. For example, if the distinguished name of the flowed certificate contains the OUs OU=Large Unit, OU=Medium Unit, OU=Small Unit, specifying the following **SSLPEER** values works:

```
('OU=Large Unit,OU=Medium Unit')
('OU=*,OU=Medium Unit,OU=Small Unit')
('OU=*,OU=Medium Unit')
```

but specifying the following **SSLPEER** values fails:

```
('OU=Medium Unit,OU=Small Unit')
('OU=Large Unit,OU=Small Unit')
('OU=Medium Unit')
('OU=Small Unit, Medium Unit, Large Unit')
```

As indicated in these examples, attributes at the low end of the hierarchy can be omitted. For example, ('OU=Large Unit,OU=Medium Unit') is equivalent to ('OU=Large Unit,OU=Medium Unit,OU=\*')

If two DNs are equal in all respects except for their domain component (DC) values, almost the same matching rules apply as for OUs. The exception is that with DC values, the left-most DC is the lowest-level and most specific, and the comparison ordering differs accordingly.

Any or all the attribute values can be generic, either an asterisk \* on its own, or a stem with initiating or trailing asterisks. Asterisks allow the **SSLPEER** to match any distinguished name value, or any value starting with the stem for that attribute. You can specify an asterisk at the beginning or end of any attribute value in the DN on the certificate. If you do so, you can still check for an exact match with **SSLPEER**. Specify \\* to check for an exact match. For example, if you have an attribute of CN= 'Test\*' in the DN of the certificate, you use the following command to check for an exact match:

```
SSLPEER('CN=Test\*')
```

**Multi** The maximum length of the parameter is 1024 bytes on [Multiplatforms](#).

**z/OS** The maximum length of the parameter is 256 bytes on z/OS.

Channel authentication records provide greater flexibility when using SSLPEER and support 1024 bytes on all platforms.

## STATCHL

Controls the collection of statistics data for channels:

### QMGR

The value of the **STATCHL** parameter of the queue manager is inherited by the channel.

### OFF

Statistics data collection is turned off for this channel.

### LOW

If the value of the **STATCHL** parameter of the queue manager is not NONE, statistics data collection is turned on. Data is collected at a low rate for this channel.

### MEDIUM

If the value of the **STATCHL** parameter of the queue manager is not NONE, statistics data collection is turned on. Data is collected at a medium rate for this channel.

### HIGH

If the value of the **STATCHL** parameter of the queue manager is not NONE, statistics data collection is turned on. Data is collected at a high rate for this channel.

Changes to this parameter take effect only on channels started after the change occurs.

**z/OS** On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

For cluster channels, the value of this parameter is not replicated in the repository and therefore is not used in the auto-definition of CLUSSDR channels. For auto-defined CLUSSDR channels, the value of this parameter is taken from the attribute **STATACLS** of the queue manager. This value might then be overridden in the channel auto-definition exit.

## Multi V 9.3.0 **TMPMODEL(string)**

The name of the model queue to be used while creating a temporary queue (maximum length 48 characters).

The default queue is SYSTEM.DEFAULT.MODEL.QUEUE.

## Multi V 9.3.0 **TMPQPRFX(string?)**

The temporary queue name prefix to add to the beginning of the model queue when deriving a temporary queue name (maximum length 48 characters).

The default is AMQP.\*.

## **TPNAME(string)**

LU 6.2 transaction program name (maximum length 64 characters).

This parameter is valid only for channels with a transport type (**TRPTYPE**) of LU62.

Set this parameter to the SNA transaction program name, unless the **CONNAME** contains a side-object name in which case set it to blanks. The actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set. See [Configuration parameters for an LU 6.2 connection](#)

**Windows** **z/OS** On Windows SNA Server, and in the side object on z/OS, the TPNAME is wrapped to uppercase.

This parameter is not valid for channels with a channel type (**CHLTYPE**) of RCVR.

## **TPROOT**

The topic root for an AMQP channel. The default value for TPROOT is SYSTEM.BASE.TOPIC. With this value, the topic string an AMQP client uses to publish or subscribe has no prefix, and the client can



exchange messages with other IBM MQ publish/subscribe applications. Alternatively, AMQP clients can publish and subscribe under a different topic prefix, specified in the TPROOT attribute.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of AMQP.

## TRPTYPE

Transport type to be used:

### LU62

SNA LU 6.2

### NETBIOS

**Windows** Supported on Windows, and DOS.

**z/OS** Also used on z/OS for defining client-connection channels that connect to servers on the platforms supporting NetBIOS.

### SPX

Sequenced packet exchange

**Windows** Supported on Windows, and DOS.

**z/OS** Also used on z/OS for defining client-connection channels that connect to servers on the platforms supporting SPX.

### TCP

Transmission Control Protocol - part of the TCP/IP protocol suite.

If you do not enter a value for this parameter, the value specified in the `SYSTEM.DEF.channel-type` definition is used. If the channel is initiated from the other end, no check is made that the correct transport type is specified.

**Multi** On Multiplatforms, if the `SYSTEM.DEF.channel-type` definition does not exist, you must specify a value.

**z/OS** On z/OS, if the `SYSTEM.DEF.channel-type` definition does not exist, the default is LU62.

### **Multi** USECLTID

Specifies that the client ID should be used for authorization checks for an AMQP channel, instead of the MCAUSER attribute value.

#### NO

The MCA user ID should be used for authorization checks.

#### YES

The client ID should be used for authorization checks.

## USEDLQ

Determines whether the dead-letter queue is used when messages cannot be delivered by channels.

#### NO

Messages that cannot be delivered by a channel are treated as a failure. The channel either discards the message, or the channel ends, in accordance with the **NPMSPEED** setting.

#### YES

When the **DEADQ** queue manager attribute provides the name of a dead-letter queue, then it is used, else the behavior is as for NO. YES is the default value.

## USERID(string)

Task user identifier. The maximum length is 12 characters.

This parameter is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

**Multi** On Multiplatforms, this parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, RQSTR, CLNTCONN, or CLUSSDR.

**z/OS** On z/OS, this parameter is supported only for CLNTCONN channels.

Although the maximum length of the parameter is 12 characters, only the first 10 characters are used.

At the receiving end, if passwords are encrypted and the LU 6.2 software is using a different encryption method, the channel does not start. The error is diagnosed as invalid security details. You can avoid invalid security details by modifying the receiving SNA configuration to either:

- Turn off password substitution, or
- Define a security user ID and password.

### **XMITQ(string)**

Transmission queue name.

The name of the queue from which messages are retrieved. See [Rules for naming IBM MQ objects](#).

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR or SVR. For these channel types, this parameter is required.

There is a separate syntax diagram for each type of channel.

### **Sender channel**

Syntax diagram for a sender channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



- <sup>2</sup> Valid only on IBM MQ for z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Not valid on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Default for z/OS.
- <sup>7</sup> Default for Multiplatforms.
- <sup>8</sup> Valid only on Windows.

The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

### ***Server channel***

Syntax diagram for a server channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



- <sup>2</sup> Valid only on IBM MQ for z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Not valid on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Default for z/OS.
- <sup>7</sup> Default for Multiplatforms.
- <sup>8</sup> Valid only on Windows.

The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

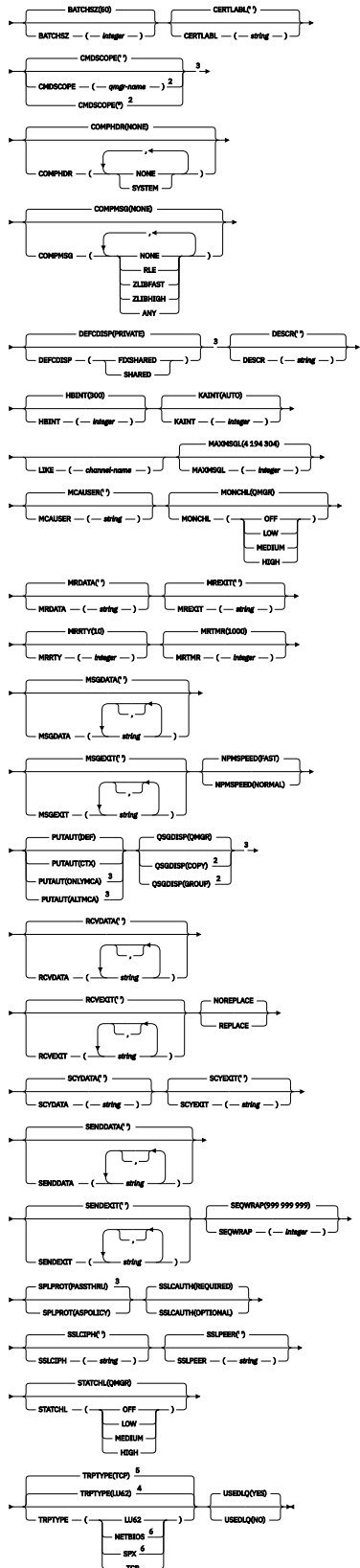
### ***Receiver channel***

Syntax diagram for a receiver channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).

## DEFINE CHANNEL

DEFINE CHANNEL (channel-name) CHLTYPE(QV) <sup>1</sup>



### Notes:

<sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.

- <sup>2</sup> Valid only on IBM MQ for z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Default for z/OS.
- <sup>5</sup> Default for Multiplatforms.
- <sup>6</sup> Valid only on Windows.

The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

### ***Requester channel***

Syntax diagram for a requester channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).





- <sup>2</sup> Valid only on IBM MQ for z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Not valid on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Default for z/OS.
- <sup>7</sup> Default for Multiplatforms.
- <sup>8</sup> Valid only on Windows.

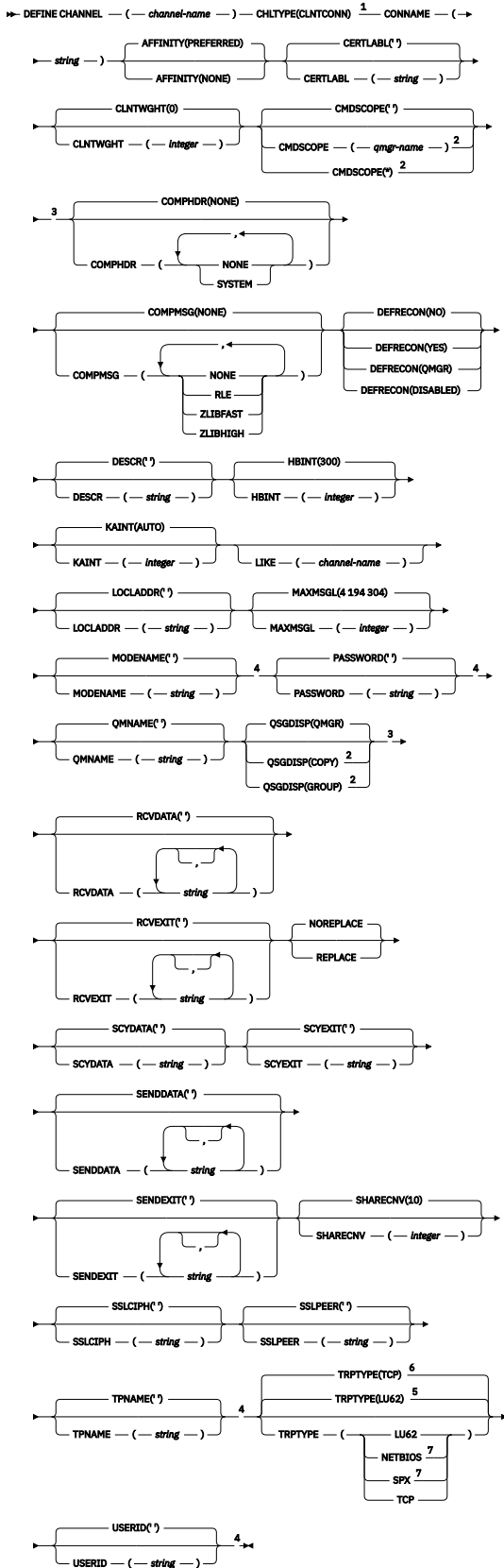
The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

### ***Client-connection channel***

Syntax diagram for a client-connection channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).

## DEFINE CHANNEL



### Notes:

<sup>1</sup> This parameter must follow immediately after the channel name except on z/OS.

- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Valid only if TRPTYPE is LU62.
- <sup>5</sup> Default for z/OS.
- <sup>6</sup> Default for Multiplatforms.
- <sup>7</sup> Valid only for clients to be run on DOS or Windows.

The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

### ***Server-connection channel***

Syntax diagram for a server-connection channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Default for z/OS.
- <sup>5</sup> Default for Multiplatforms.
- <sup>6</sup> Valid only for clients to be run on Windows.

The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

### ***Cluster-sender channel***

Syntax diagram for a cluster-sender channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



- <sup>2</sup> This parameter must follow immediately after the channel name except on z/OS.
- <sup>3</sup> Valid only on IBM MQ for z/OS when the queue manager is a member of a queue sharing group.
- <sup>4</sup> Valid only on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Default for z/OS.
- <sup>7</sup> Default for Multiplatforms.
- <sup>8</sup> Valid only on Windows.

The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

### ***Cluster-receiver channel***

Syntax diagram for a cluster-receiver channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).





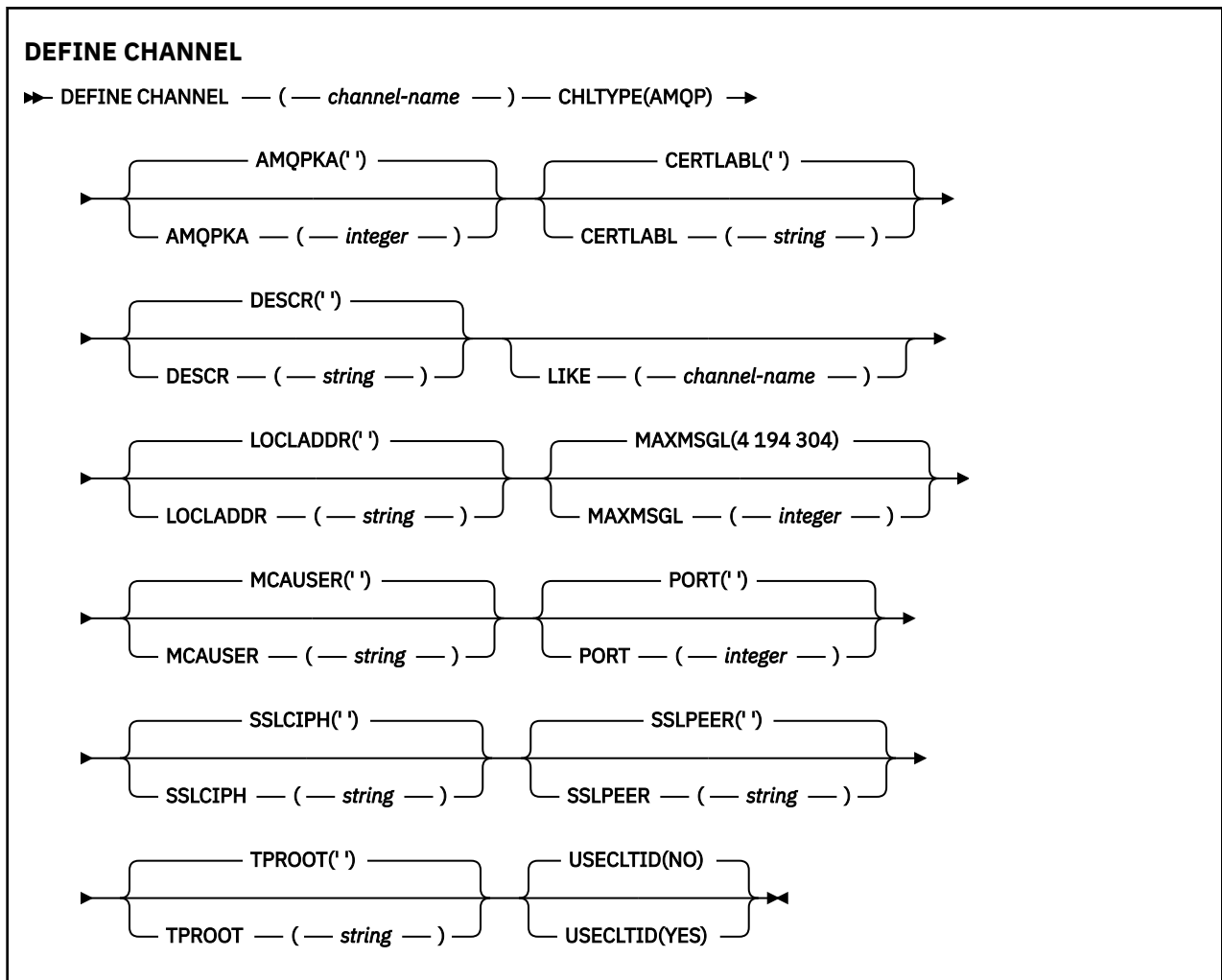
- <sup>2</sup> This parameter is optional if TRPTYPE is TCP.
- <sup>3</sup> Valid only on IBM MQ for z/OS when the queue manager is a member of a queue sharing group.
- <sup>4</sup> Valid only on z/OS.
- <sup>5</sup> Valid only if TRPTYPE is LU62.
- <sup>6</sup> Default for z/OS.
- <sup>7</sup> Default for Multiplatforms.
- <sup>8</sup> Valid only on Windows.

The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

### **ALW** **AMQP channel**

Syntax diagram for an AMQP channel when using the DEFINE CHANNEL command.

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



The parameters are described in [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

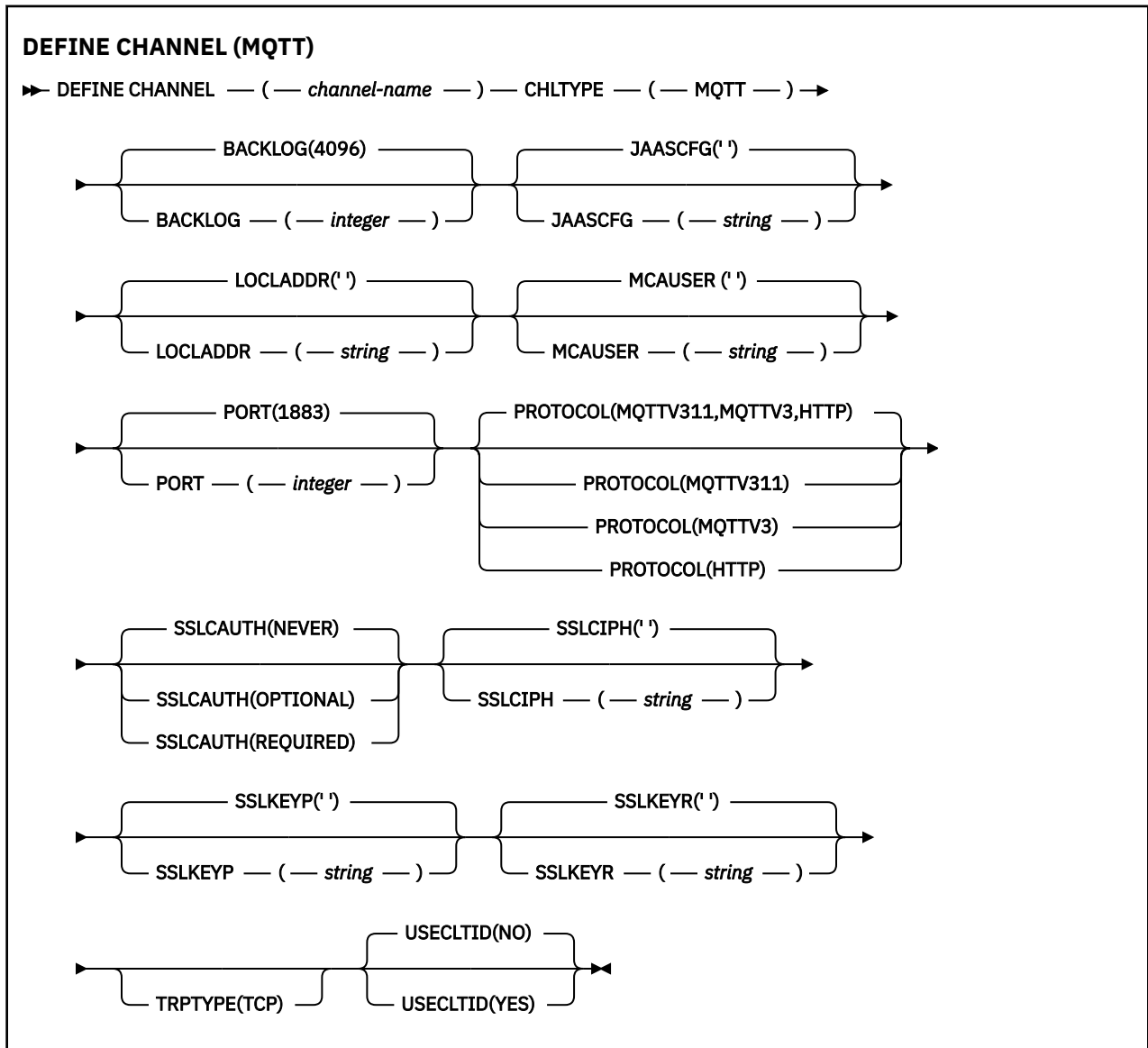
**MQTT**

Syntax diagram for a telemetry channel when using the **DEFINE CHANNEL** command.

**Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

Synonym: DEF CHL

**Usage notes**

The telemetry (MQXR) service must be running when you issue this command. For instructions on how to start the telemetry (MQXR) service, see [Configuring a queue manager for telemetry on Linux or Configuring a queue manager for telemetry on Windows](#).

**Parameter descriptions for DEFINE CHANNEL (MQTT)**

(*channel-name*)

The name of the new channel definition.

The name must not be the same as any existing channel defined on this queue manager (unless REPLACE or ALTER is specified).

The maximum length of the string is 20 characters, and the string must contain only valid characters; see [Rules for naming IBM MQ objects](#).

### **BACKLOG(*integer*)**

The number of outstanding connection requests that the telemetry channel can support at any one time. When the backlog limit is reached, any further clients trying to connect will be refused connection until the current backlog is processed.

The value is in the range 0 - 999999999.

The default value is 4096.

### **CHLTYPE**

Channel type. MQTT (telemetry) channel.

### **JAASCFG (*string*)**

The name of a stanza in the JAAS configuration file.

See [Authenticating an MQTT client Java app with JAAS](#)

### **LOCLADDR (*ip-addr*)**

LOCLADDR is the local communications address for the channel. Use this parameter if you want to force the client to use a particular IP address. LOCLADDR is also useful to force a channel to use an IPv4 or IPv6 address if a choice is available, or to use a particular network adapter on a system with multiple network adapters.

The maximum length of **LOCLADDR** is MQ\_LOCAL\_ADDRESS\_LENGTH.

If you omit **LOCLADDR**, a local address is automatically allocated.

#### **ip-addr**

*ip-addr* is a single network address, specified in one of three forms:

#### **IPv4 dotted decimal**

For example 192.0.2.1

#### **IPv6 hexadecimal notation**

For example 2001:DB8:0:0:0:0:0:0

#### **Alphanumeric host name form**

For example WWW.EXAMPLE.COM

If an IP address is entered, only the address format is validated. The IP address itself is not validated.

### **MCAUSER(*string*)**

Message channel agent user identifier.

The maximum length of the string is 12 characters. On Windows, you can optionally qualify a user identifier with the domain name in the format `user@domain`.

If this parameter is nonblank, and **USECLNTID** is set to NO, then this user identifier is used by the telemetry service for authorization to access IBM MQ resources.

If this parameter is blank, and **USECLNTID** is set to NO, then the user name flowed in the MQTT CONNECT Packet is used. See [MQTT client identity and authorization](#).

### **PORT(*integer*)**

The port number on which the telemetry (MQXR) service accepts client connections. The default port number for a telemetry channel is 1883; and the default port number for a telemetry channel secured using SSL is 8883. Specifying a port value of 0 causes MQTT to dynamically allocate an available port number.

## PROTOCOL

The following communication protocols are supported by the channel:

### **MQTTV311**

The channel accepts connections from clients using the protocol defined by the [MQTT 3.1.1](#) Oasis standard. The functionality provided by this protocol is almost identical to that provided by the pre-existing MQTTV3 protocol.

### **MQTTV3**

The channel accepts connections from clients using the [MQTT V3.1 Protocol Specification](#) from [mqtt.org](#).

### **HTTP**

The channel accepts HTTP requests for pages, or WebSockets connections to MQ Telemetry.

To accept connections from clients using different protocols, specify the acceptable values as a comma-delimited list. For example if you specify MQTTV3, HTTP the channel accepts connections from clients using either MQTTV3 or HTTP. If you specify no client protocols, the channel accepts connections from clients using any of the supported protocols.

If you are using IBM MQ 8.0.0 Fix Pack 3 or later, and your configuration includes an MQTT channel that was last modified in an earlier version of the product, you must explicitly change the protocol setting to prompt the channel to use the MQTTV311 option. This is so even if the channel does not specify any client protocols, because the specific protocols to use with the channel are stored at the time the channel is configured, and previous versions of the product have no awareness of the MQTTV311 option. To prompt a channel in this state to use the MQTTV311 option, explicitly add the option then save your changes. The channel definition is now aware of the option. If you subsequently change the settings again, and specify no client protocols, the MQTTV311 option is still included in the stored list of supported protocols.

## SSLCAUTH

Defines whether IBM MQ requires a certificate from the TLS client. The initiating end of the channel acts as the TLS client, so this parameter applies to the end of the channel that receives the initiation flow, which acts as the TLS server.

### **NEVER**

IBM MQ never requests a certificate from the TLS client.

### **REQUIRED**

IBM MQ requires and validates a certificate from the TLS client.

### **OPTIONAL**

IBM MQ lets the TLS client decide whether to provide a certificate. If the client sends a certificate, the contents of this certificate are validated as normal.

## SSLCIPH(*string*)

When **SSLCIPH** is used with a telemetry channel, it means TLS Cipher Suite. The TLS cipher suite is the one supported by the JVM that is running the telemetry (MQXR) service. If the parameter is blank, no attempt is made to use TLS on the channel.

If you plan to use SHA-2 cipher suites, see [System requirements for using SHA-2 cipher suites with MQTT channels](#).

## SSLKEYP(*string*)



The passphrase for the TLS key repository.

**V9.3.0** If the MQXR service is configured for encryption of passphrases by specifying the **-sf** option in STARTARG for the service, then the pass phrase will be encrypted. For more information on encryption of passphrases, see [Encryption of passphrases for MQTT TLS channels](#).

## SSLKEYR(*string*)

The full path name of the TLS key repository file, the store for digital certificates and their associated private keys. If you do not specify a key file, TLS is not used.

The maximum length of the string is 256 characters;

-  On AIX and Linux, the name is of the form *pathname/keyfile*.
-  On Windows, the name is of the form *pathname\keyfile*.

where *keyfile* is specified without the suffix `.jks`, and identifies a Java keystore file.

## TRPTYPE (*string*)

The transmission protocol to be used:

### TCP

TCP/IP.

## USECLTID

Decide whether you want to use the MQTT client ID for the new connection as the IBM MQ user ID for that connection. If this property is specified, the user name supplied by the client is ignored.

If you set this parameter to YES, then **MCAUSER** must be blank.

If **USECLTID** is set to NO, and **MCAUSER** is blank, then the user name flowed in the MQTT CONNECT Packet is used. See [MQTT client identity and authorization](#).

## Related concepts

[Telemetry channel configuration for MQTT client authentication using TLS](#)

[Telemetry channel configuration for channel authentication using TLS](#)

[CipherSpecs and CipherSuites](#)

## Related reference

[“ALTER CHANNEL \(alter channel settings\) MQTT” on page 329](#)

Syntax diagram for a telemetry channel when using the **ALTER CHANNEL** command.

[System requirements for using SHA-2 cipher suites with MQTT channels](#)

## **DEFINE COMMINFO (define a new communication information object) on Multiplatforms**

Use the MQSC command **DEFINE COMMINFO** to define a new communication information object. These objects contain the definitions required for Multicast messaging.

## Using MQSC commands

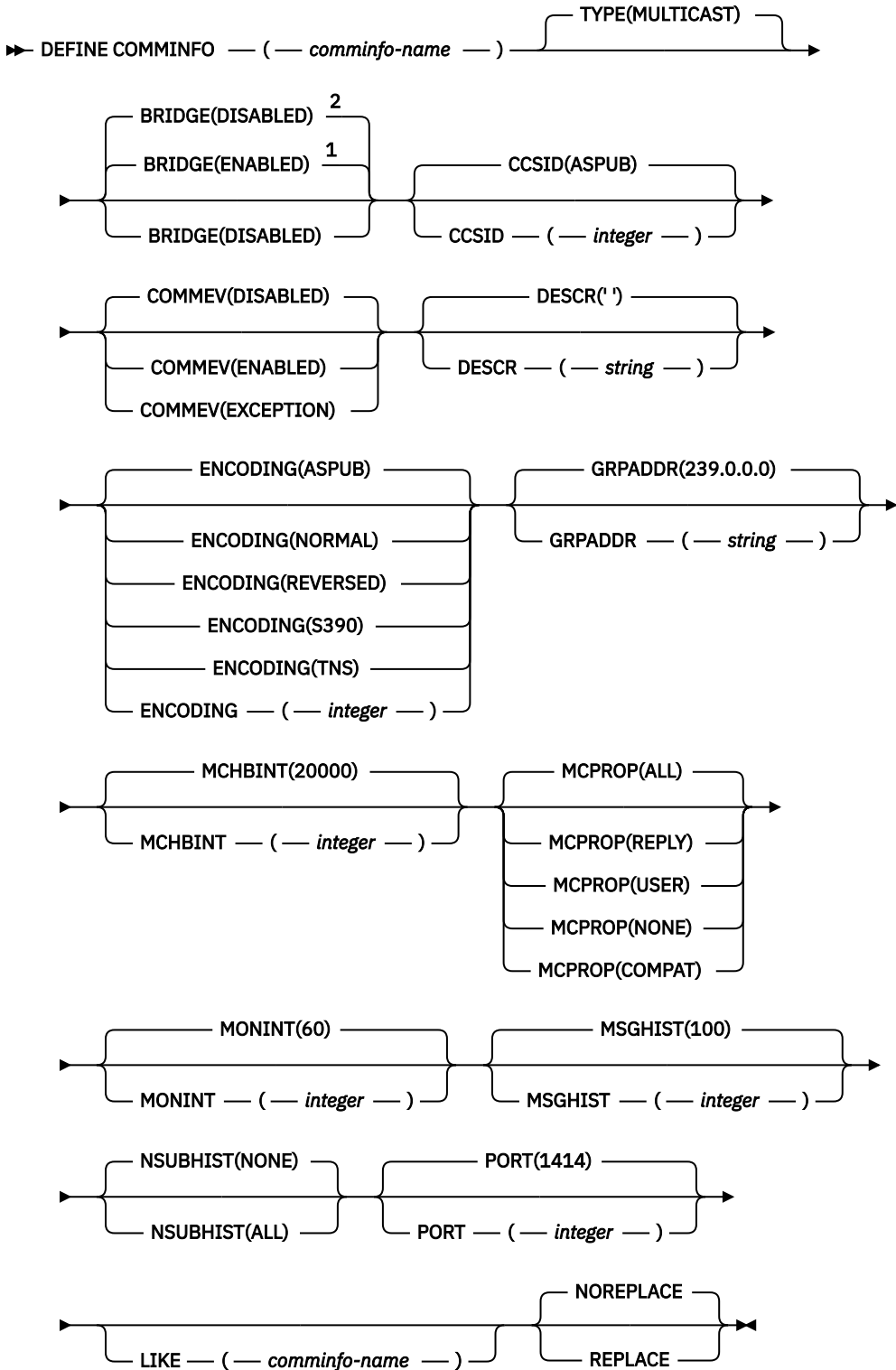
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DEFINE COMMINFO” on page 528](#)

**Synonym:** DEF COMMINFO

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).

## DEFINE COMMINFO



### Notes:

- <sup>1</sup> Default for platforms other than IBM i.
- <sup>2</sup> Default for IBM i.

## Parameter descriptions for DEFINE COMMINFO

### *(comminfo name)*

Name of the communications information object. This is required.

The name must not be the same as any other communications information object name currently defined on this queue manager. See [Rules for naming IBM MQ objects](#).

### TYPE


The type of the communications information object. The only type supported is MULTICAST.

### BRIDGE

Controls whether publications from applications not using Multicast are bridged to applications using Multicast. Bridging does not apply to topics that are marked as **MCAST (ONLY)**. As these topics can only be Multicast traffic, it is not applicable to bridge to the queue's publish/subscribe domain.

### DISABLED

Publications from applications not using Multicast are not bridged to applications that do use Multicast.

 This is the default for IBM i.

### ENABLED

Publications from applications not using Multicast are bridged to applications that do use Multicast. This is the default for platforms other than IBM i.

### CCSID( *integer* )

The coded character set identifier that messages are transmitted on. Specify a value in the range 1 through 65535.

The CCSID must specify a value that is defined for use on your platform, and use a character set that is appropriate to the platform. If you use this parameter to change the CCSID, applications that are running when the change is applied continue to use the original CCSID. Because of this, you must stop and restart all running applications before you continue. This includes the command server and channel programs. To do this, stop and restart the queue manager after making the change.

The default value is ASPUB which means that the coded character set is taken from the one that is supplied in the published message.

### COMMEV

Controls whether event messages are generated for Multicast handles that are created using this COMMINFO object. Events will only be generated if they are enabled using the **MONINT** parameter.

### DISABLED

Event messages are not generated for Multicast handles that are created using the COMMINFO object. This is the default value.

### ENABLED

Event messages are generated for Multicast handles that are created using the COMMINFO object.

### EXCEPTION

Event messages are written if the message reliability is below the reliability threshold. The reliability threshold is set to 90 by default.

### DESCR( *string* )

Plain-text comment. It provides descriptive information about the communication information object when an operator issues the DISPLAY COMMINFO command (see [“DISPLAY COMMINFO \(display communication information\) on Multiplatforms”](#) on page 722).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

### ENCODING

The encoding that the messages are transmitted in.



**ASPub**

The encoding of the message is taken from the one that is supplied in the published message. This is the default value.

**REVERSED****NORMAL****S390****TNS****encoding****GRPADDR**

The group IP address or DNS name.

It is the administrator's responsibility to manage the group addresses. It is possible for all multicast clients to use the same group address for every topic; only the messages that match outstanding subscriptions on the client are delivered. Using the same group address can be inefficient because every client must examine and process every multicast packet in the network. It is more efficient to allocate different IP group addresses to different topics or sets of topics, but this requires careful management, especially if other non-MQ multicast applications are in use on the network. The default value is 239.0.0.0.

**MCHBINT**

The heartbeat interval is measured in milliseconds, and specifies the frequency at which the transmitter notifies any receivers that there is no further data available. The value is in the range 0 to 999 999. The default value is 2000 milliseconds.

**MCPROP**

The multicast properties control how many of the MQMD properties and user properties flow with the message.

**All**

All user properties and all the fields of the MQMD are transported.

**Reply**

Only user properties, and MQMD fields that deal with replying to the messages, are transmitted. These properties are:

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

**User**


Only the user properties are transmitted.

**NONE**

No user properties or MQMD fields are transmitted.

**COMPAT**

This value causes the transmission of the message to be done in a compatible mode to RMM. This allows some inter-operation with the current XMS applications and Broker RMM applications.

 XMS .NET Multicast messaging (using RMM) was deprecated from IBM MQ 9.2 and is removed at IBM MQ 9.3.

**MONINT( integer )**

How frequently, in seconds, that monitoring information is updated. If events messages are enabled, this parameter also controls how frequently event messages are generated about the status of the Multicast handles created using this COMMINFO object.

A value of 0 means that there is no monitoring.

The default value is 60.

### **MSGHIST**

This value is the amount of message history in kilobytes that is kept by the system to handle retransmissions in the case of NACKs (negative acknowledgments).

The value is in the range 0 to 999 999 999. A value of 0 gives the least level of reliability. The default value is 100.

### **NSUBHIST**

The new subscriber history controls whether a subscriber joining a publication stream receives as much data as is currently available, or receives only publications made from the time of the subscription.

#### **NONE**

A value of NONE causes the transmitter to transmit only publication made from the time of the subscription. This is the default value.

#### **ALL**

A value of ALL causes the transmitter to retransmit as much history of the topic as is known. In some circumstances this can give a similar behavior to retained publications.

**Note:** Using the value of ALL might have a detrimental effect on performance if there is a large topic history because all the topic history is retransmitted.

### **PORT( *integer* )**

The port number to transmit on. The default port number is 1414.

### **LIKE( *authinfo-name* )**

The name of a communication information object, with parameters that are used to model this definition.

If this field is not complete, and you do not complete the parameter fields related to the command, the values are taken from the default definition for an object of this type.

This default communication information object definition can be altered by the installation to the default values required.

### **REPLACE and NOREPLACE**

Whether the existing definition is to be replaced with this one. This is optional. The default is NOREPLACE. Any object with a different disposition is not changed.

#### **REPLACE**

The definition replaces an existing definition of the same name. If a definition does not exist, one is created.

#### **NOREPLACE**

The definition does not replace an existing definition of the same name.

### **Related tasks**

[Getting started with multicast](#)

## **DEFINE LISTENER (define a new listener) on Multiplatforms**

Use the MQSC command DEFINE LISTENER to define a new IBM MQ listener definition, and set its parameters.

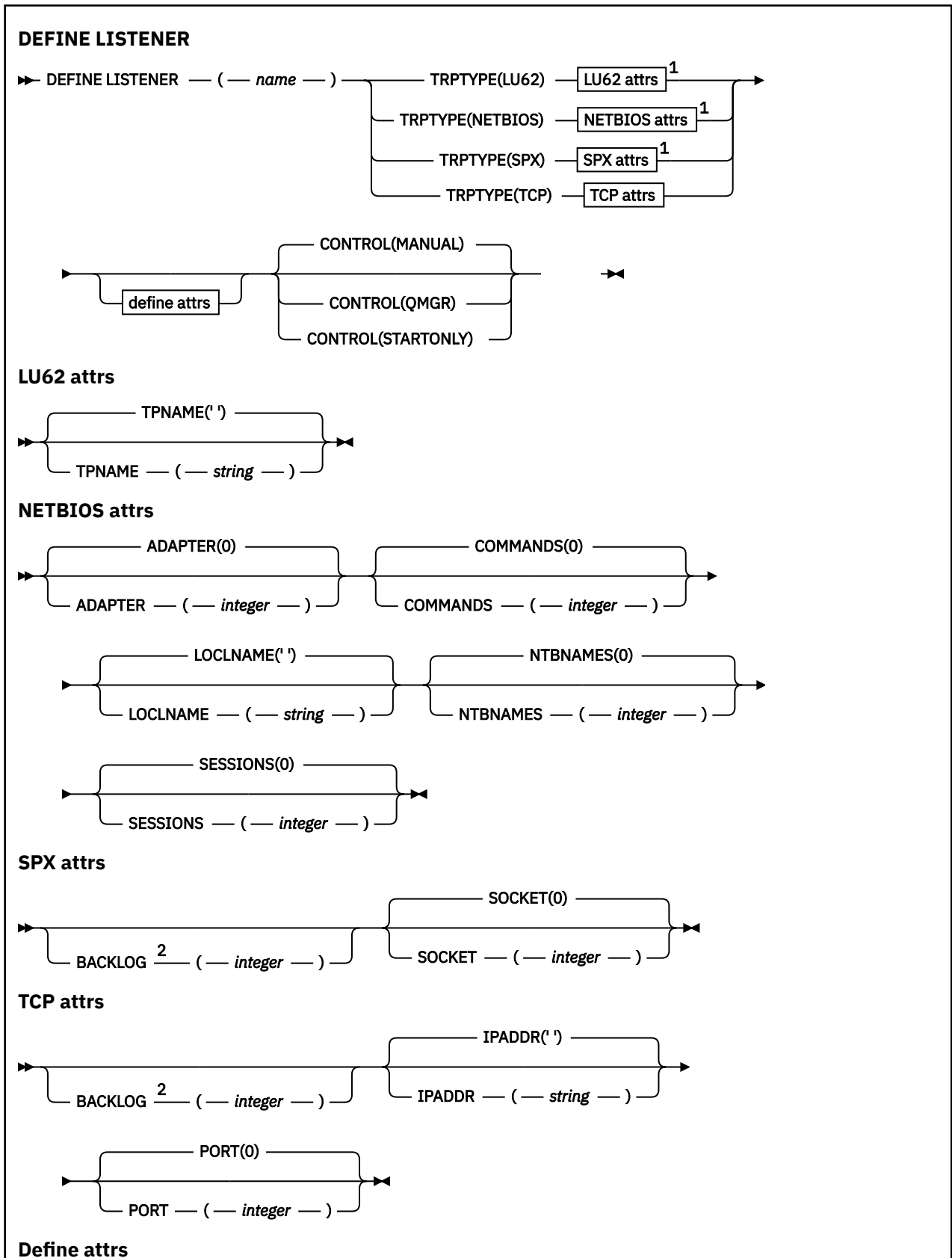
### **Using MQSC commands**

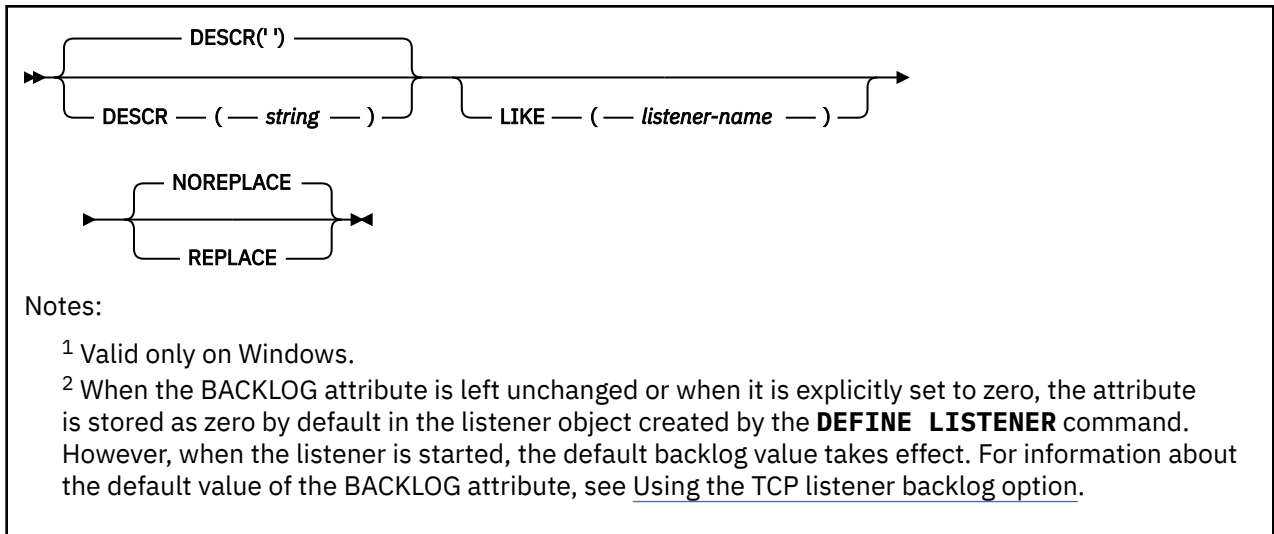
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DEFINE LISTENER” on page 532](#)

**Synonym:** DEF LSTR

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).





## Parameter descriptions for **DEFINE LISTENER**

### **(listener-name)**

Name of the IBM MQ listener definition (see [Rules for naming IBM MQ objects](#)). This is required.

The name must not be the same as any other listener definition currently defined on this queue manager (unless **REPLACE** is specified).

### **Windows ADAPTER(integer)**

The adapter number on which NetBIOS listens. This parameter is valid only on Windows when **TRPTYPE** is **NETBIOS**.

### **BACKLOG(integer)**

The number of concurrent connection requests that the listener supports.

### **Windows COMMANDS(integer)**

The number of commands that the listener can use. This parameter is valid only on Windows when **TRPTYPE** is **NETBIOS**.

### **CONTROL(string)**

Specifies how the listener is to be started and stopped.:

#### **MANUAL**

The listener is not to be started automatically or stopped automatically. It is to be controlled by use of the **START LISTENER** and **STOP LISTENER** commands.

#### **QMGR**

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

#### **STARTONLY**

The listener is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

### **DESCR(string)**

Plain-text comment. It provides descriptive information about the listener when an operator issues the **DISPLAY LISTENER** command (see [“DISPLAY LISTENER \(display listener information\) on Multiplatforms”](#) on page 742).

It should contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

**IPADDR(string)**

IP address for the listener specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric host name form. If you do not specify a value for this parameter, the listener listens on all configured IPv4 and IPv6 stacks.

**LIKE(listener-name)**

The name of a listener, with parameters that are used to model this definition.

This parameter applies only to the **DEFINE LISTENER** command.

If this field is not filled in, and you do not complete the parameter fields related to the command, the values are taken from the default definition for listeners on this queue manager. This is equivalent to specifying:

```
LIKE(SYSTEM.DEFAULT.LISTENER)
```

A default listener is provided but it can be altered by the installation of the default values required. See [Rules for naming IBM MQ objects](#).

**Windows LOCLNAME(string)**

The NetBIOS local name that the listener uses. This parameter is valid only on Windows when **TRPTYPE** is NETBIOS.

**Windows NTBNAMES(integer)**

The number of names that the listener can use. This parameter is valid only on Windows when **TRPTYPE** is NETBIOS.

**PORT(integer)**

The port number for TCP/IP. This is valid only when **TRPTYPE** is TCP. It must not exceed 65535.

**Windows SESSIONS(integer)**

The number of sessions that the listener can use. This parameter is valid only on Windows when **TRPTYPE** is NETBIOS.

**SOCKET(integer)**

The SPX socket on which to listen. This is valid only if **TRPTYPE** is SPX.

**Windows TPNAME(string)**

The LU 6.2 transaction program name (maximum length 64 characters). This parameter is valid only on Windows when **TRPTYPE** is LU62.

**TRPTYPE(string)**

The transmission protocol to be used:

**Windows LU62**

SNA LU 6.2. This is valid only on Windows.

**Windows NETBIOS**

NetBIOS. This is valid only on Windows.

**Windows SPX**

Sequenced packet exchange. This is valid only on Windows.

**TCP**

TCP/IP.



..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

**z/OS DEFINE MAXSMSGS (define maximum messages setting) on z/OS**

Use the MQSC command DEFINE MAXSMSGS to define the maximum number of messages that a task can get or put within a single unit of recovery.

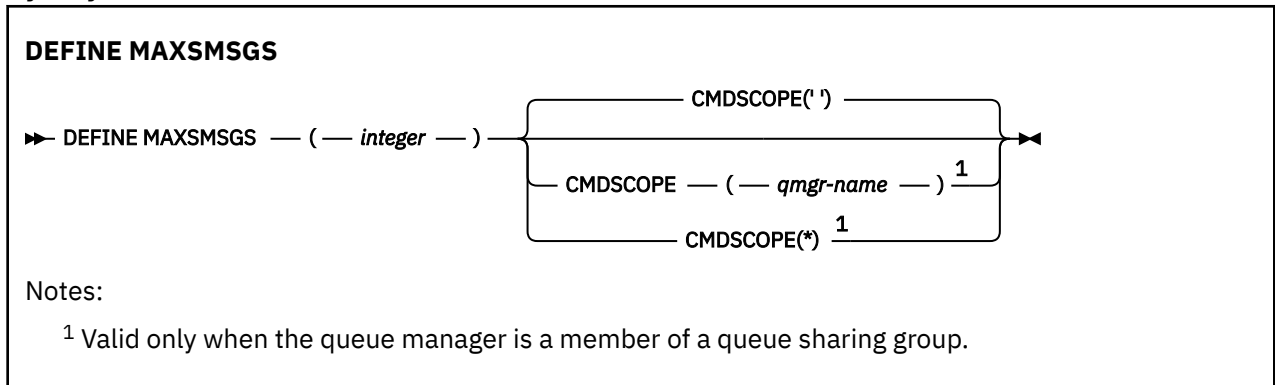
**Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 535](#)
- [“Parameter descriptions for DEFINE MAXSMSGS” on page 535](#)

**Synonym:** DEF MAXSM



**Usage notes**

1. This command is valid only on z/OS and is retained for compatibility with earlier releases, although it can no longer be issued from the CSQINP1 initialization input data set. You should use the MAXUMSGS parameter of the ALTER QMGR command instead.
2. You can issue the DEFINE MAXSMSGS command to change the number of messages allowed. Once a value is set, it is preserved during a queue manager restart.

**Parameter descriptions for DEFINE MAXSMSGS**

**(integer)**

The maximum number of messages that a task can get or put within a single unit of recovery. This value must be an integer in the range 1 through 999999999. The default value is 10000.

The number includes any trigger messages and report messages generated within the same unit of recovery.

## CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*


The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## DEFINE NAMELIST (define a list of names)

Use the MQSC command **DEFINE NAMELIST** to define a list of names. This is most commonly a list of cluster names or queue names.

### Using MQSC commands

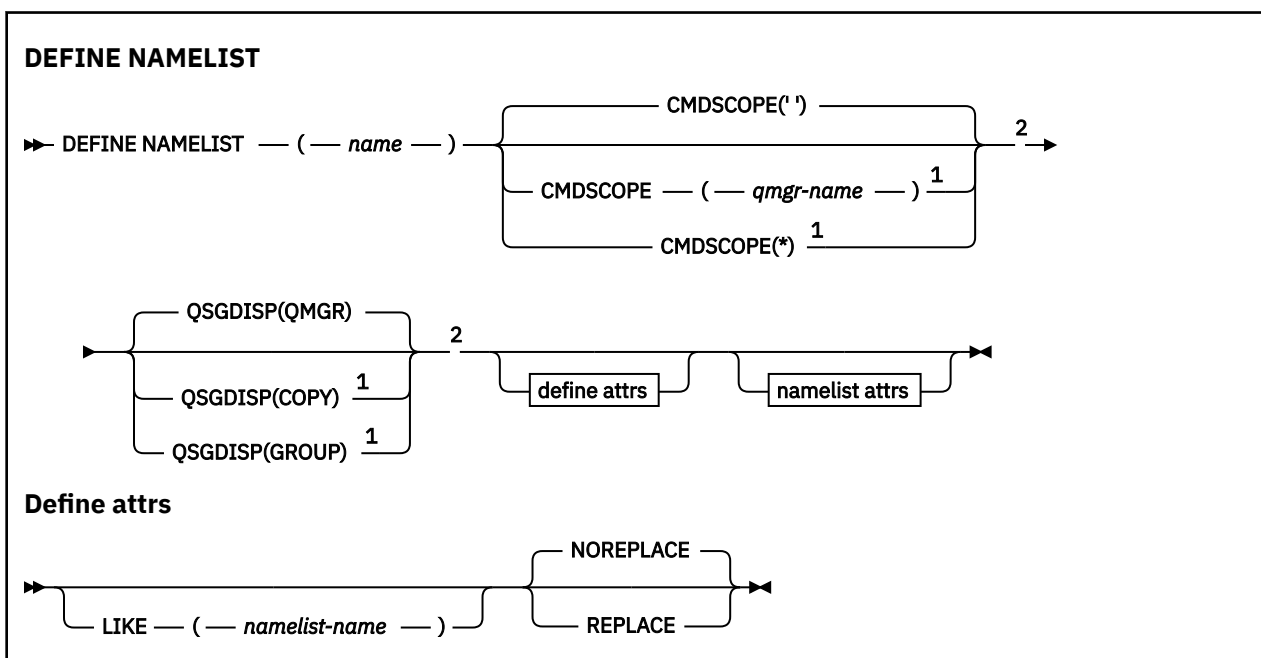
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

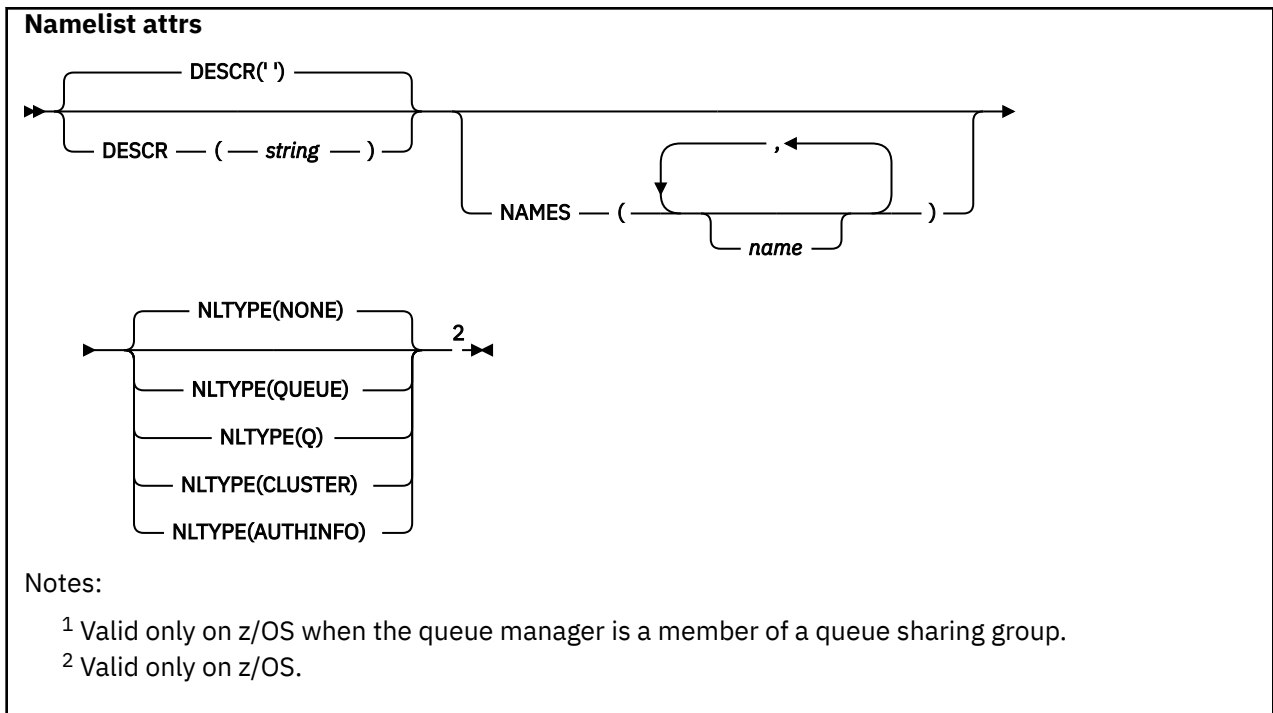
- [Syntax diagram](#)
- [“Usage notes” on page 537](#)
- [“Parameter descriptions for DEFINE NAMELIST” on page 537](#)

**Synonym:** DEF NL

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).







## Usage notes

Successful completion of the command does not mean that the action completed. To check for true completion, see the [DEFINE NAMESPACE](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for DEFINE NAMESPACE

### (name)

Name of the list.

The name must not be the same as any other namespace name currently defined on this queue manager (unless REPLACE or ALTER is specified). See [Rules for naming IBM MQ objects](#).

### z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of specifying \* is the same as entering the command on every queue manager in the queue sharing group.

**DESCR(string)**

Plain-text comment. It provides descriptive information about the namelist when an operator issues the **DISPLAY NAMELIST** command (see [“DISPLAY NAMELIST \(display a list of names\)”](#) on page 751).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

**LIKE(namelist-name)**

The name of a namelist, with parameters that are used to model this definition.

If this field is not completed and you do not complete the parameter fields related to the command, the values are taken from the default definition for namelists on this queue manager.

Not completing this parameter is equivalent to specifying:

```
LIKE(SYSTEM.DEFAULT.NAMELIST)
```

A default namelist definition is provided, but it can be altered by the installation to the default values required. See [Rules for naming IBM MQ objects](#).

**z/OS** On z/OS, the queue manager searches page set zero for an object with the name you specify and a disposition of QMGR or COPY. The disposition of the LIKE object is not copied to the object you are defining.

**Note:**

1. QSGDISP (GROUP) objects are not searched.
2. LIKE is ignored if QSGDISP(COPY) is specified.

**NAMES(name, ...)**

List of names.

The names can be of any type, but must conform to the rules for naming IBM MQ objects, with a maximum length of 48 characters.

An empty list is valid: specify NAMES(). The maximum number of names in the list is 256.

**z/OS NLTYPE**

Indicates the type of names in the namelist.

This parameter is valid only on z/OS.

**NONE**

The names are of no particular type.

**QUEUE or Q**

A namelist that holds a list of queue names.

**CLUSTER**

A namelist that is associated with clustering, containing a list of the cluster names.

**AUTHINFO**

This namelist is associated with TLS and contains a list of authentication information object names.

Namelists used for clustering must have NLTYPE(CLUSTER) or NLTYPE(NONE).

Namelists used for TLS must have NLTYPE(AUTHINFO).

**z/OS QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

<i>Table 158. Object dispositions for QSGDISP options</i>	
<b>QSGDISP</b>	<b>DEFINE</b>
COPY	<p>The object is defined on the page set of the queue manager that executes the command. It uses the <b>QSGDISP (GROUP)</b> object of the same name as the LIKE object.</p> <p>For example, if you issue the following command,</p> <pre>DEFINE NAMELIST(<i>namelist_name</i>) REPLACE QSGDISP(COPY)</pre> <p>the queue manager searches the shared configuration repository for a NAMELIST definition called <i>namelist_name</i>. If a matching NAMELIST definition is found, the queue manager creates a local copy of this definition on the queue manager page set.</p> <p>For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.</p>
GROUP	<p>The object definition resides in the shared configuration repository. <b>QSGDISP (GROUP)</b> is allowed only if the queue manager is in a queue sharing group.</p> <p>If the DEFINE for the <b>QSGDISP (GROUP)</b> object is successful, the DEFINE NAMELIST(<i>namelist_name</i>) REPLACE QSGDISP(COPY) command is generated and sent to all active queue managers in the queue sharing group to make or refresh local copies on page set zero.</p> <p>The DEFINE for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
PRIVATE	Not permitted.
QMGR	The object is defined on the page set of the queue manager that executes the command.

### **REPLACE and NOREPLACE**

Whether the existing definition (and on z/OS, with the same disposition) is to be replaced with this one. Any object with a different disposition is not changed.

#### **REPLACE**

The definition replaces any existing definition of the same name. If a definition does not exist, one is created.

#### **NOREPLACE**

The definition does not replace any existing definition of the same name.

### **Related concepts**

[Namelists](#)

### **Related tasks**

[Adding a new, interconnected cluster](#)

## DEFINE PROCESS (create a new process definition)

Use the MQSC command DEFINE PROCESS to define a new IBM MQ, process definition, and set its parameters.

### Using MQSC commands

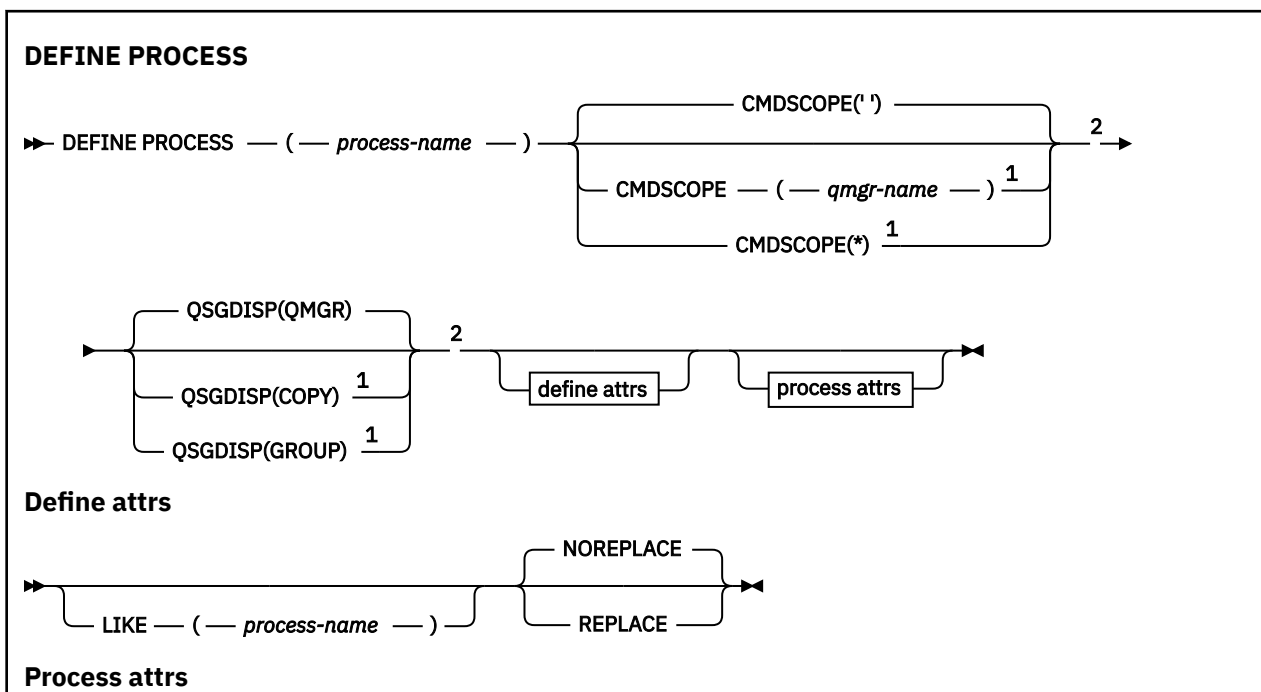
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

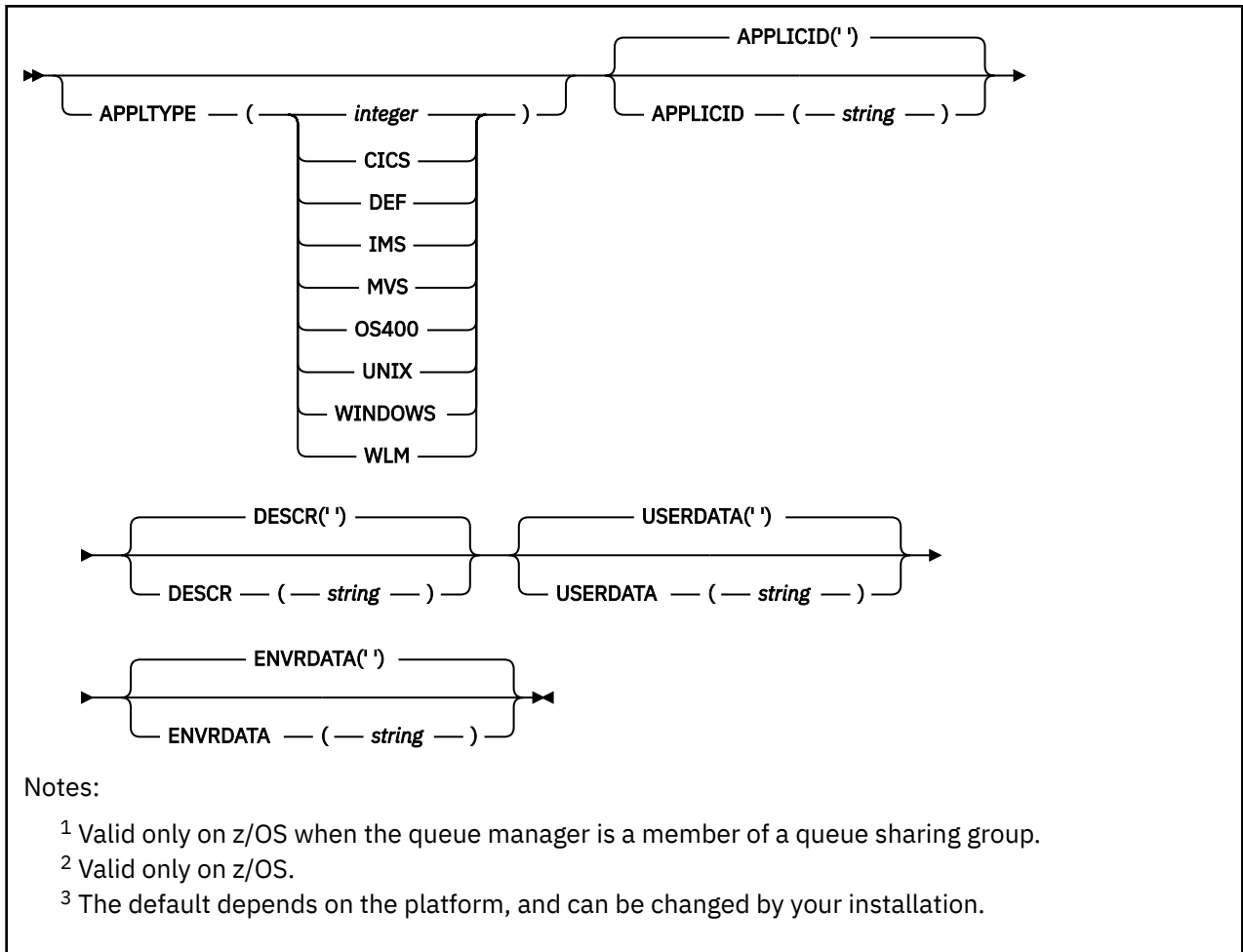
**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DEFINE PROCESS” on page 541](#)

**Synonym:** DEF PRO

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).





## Parameter descriptions for DEFINE PROCESS

### (*process-name*)

Name of the IBM MQ process definition (see [Rules for naming IBM MQ objects](#)). *process-name* is required.

The name must not be the same as any other process definition currently defined on this queue manager (unless REPLACE is specified).

### APPLICID(*string*)

The name of the application to be started. The name might typically be a fully qualified file name of an executable object. Qualifying the file name is particularly important if you have multiple IBM MQ installations, to ensure the correct version of the application is run. The maximum length is 256 characters.

For a CICS application the name is a CICS transaction ID.

**z/OS** For an IMS application, it is an IMS transaction ID.

**z/OS** On z/OS, for distributed queuing, it must be **CSQX START**.

### APPLTYPE(*string*)

The type of application to be started. Valid application types are:

#### integer

A system-defined application type in the range zero through 65 535 or a user-defined application type in the range 65 536 through 999 999 999.

For certain values in the system range, a parameter from the following list can be specified instead of a numeric value:

#### **CICS**

Represents a CICS transaction.

#### **z/OS IMS**

Represents an IMS transaction.

#### **z/OS MVS**

Represents a z/OS application (batch or TSO).

#### **IBM i OS400**

Represents an IBM i application.

#### **UNIX**

Represents a Linux or AIX application.

#### **WINDOWS**

Represents a Windows application.

#### **z/OS WLM**

Represents a z/OS workload manager application.

#### **DEF**

Specifying DEF causes the default application type for the platform at which the command is interpreted to be stored in the process definition. This default cannot be changed by the installation. If the platform supports clients, the default is interpreted as the default application type of the server.

Only use application types (other than user-defined types) that are supported on the platform at which the command is run:

- **z/OS** On z/OS, CICS, IMS, MVS, UNIX, WINDOWS, WLM, and DEF are supported.
- **IBM i** On IBM i, OS400, CICS, and DEF are supported.
- **Linux** **AIX** On AIX and Linux, UNIX, WINDOWS, CICS, and DEF are supported.
- **Windows** On Windows, WINDOWS, UNIX, CICS, and DEF are supported.

#### **z/OS CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

• •

The command runs on the queue manager on which it was entered.

#### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

In a shared queue environment, you can provide a different queue manager name from the one you are using to enter the command. The command server must be enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect is the same as entering the command on every queue manager in the queue sharing group.

#### **DESCR( string )**

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY PROCESS command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).



**Note:** Use characters from the coded character set identifier (CCSID) for this queue manager. Other characters might be translated incorrectly if the information is sent to another queue manager.

### **ENVRDATA( *string* )**

A character string that contains environment information pertaining to the application to be started. The maximum length is 128 characters.

The meaning of ENVRDATA is determined by the trigger-monitor application. The trigger monitor provided by IBM MQ appends ENVRDATA to the parameter list passed to the started application. The parameter list consists of the MQTMC2 structure, followed by one blank, followed by ENVRDATA with trailing blanks removed.

#### **Notes:**

1.  On z/OS, ENVRDATA is not used by the trigger-monitor applications provided by IBM MQ.
2.  On z/OS, if APPLTYPE is WLM, the default values for the ServiceName and ServiceStep fields in the work information header (MQWIH) can be supplied in ENVRDATA. The format must be:

```
SERVICENAME=servname, SERVICESTEP=stepname
```

where:

#### **SERVICENAME=**

is the first 12 characters of ENVRDATA.

#### **servname**

is a 32-character service name. It can contain embedded blanks or any other data, and have trailing blanks. It is copied to the MQWIH as is.

#### **SERVICESTEP=**

is the next 13 characters of ENVRDATA.

#### **stepname**

is a 1 - 8 character service step name. It is copied as-is to the MQWIH, and padded to eight characters with blanks.

If the format is incorrect, the fields in the MQWIH are set to blanks.

3. On AIX and Linux, ENVRDATA can be set to the ampersand character to make the started application run in the background.

### **LIKE( *process-name* )**


The name of an object of the same type, with parameters that are used to model this definition.

If this field is not provided, the values of fields you do not provide are taken from the default definition for this object.

Using LIKE is equivalent to specifying:

```
LIKE (SYSTEM.DEFAULT.PROCESS)
```

A default definition for each object type is provided. You can alter the provided defaults to the default values required. See [Rules for naming IBM MQ objects](#).

 On z/OS, the queue manager searches page set zero for an object with the name you specify and a disposition of QMGR or COPY. The disposition of the LIKE object is not copied to the object you are defining.

#### **Note:**

1. QSGDISP (GROUP) objects are not searched.
2. LIKE is ignored if QSGDISP(COPY) is specified.


## QSGDISP

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

<b>QSGDISP</b>	<b>DEFINE</b>
COPY	<p>The object is defined on the page set of the queue manager that executes the command. It uses the <b>QSGDISP (GROUP)</b> object of the same name as the LIKE object.</p> <p>For example, if you issue the following command,</p> <pre>DEFINE PROCESS(<i>process_name</i>) REPLACE QSGDISP(COPY)</pre> <p>the queue manager searches the shared configuration repository for a PROCESS definition called <i>process_name</i>. If a matching PROCESS definition is found, the queue manager creates a local copy of this definition on the queue manager page set.</p> <p>For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.</p>
GROUP	<p>The object definition resides in the shared configuration repository. <b>QSGDISP (GROUP)</b> is allowed only if the queue manager is in a queue sharing group.</p> <p>If the DEFINE for the <b>QSGDISP (GROUP)</b> object is successful, the DEFINE PROCESS(<i>process_name</i>) REPLACE QSGDISP(COPY) command is generated and sent to all active queue managers in the queue sharing group to make or refresh local copies on page set zero.</p> <p>The DEFINE for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
PRIVATE	Not permitted.
QMGR	The object is defined on the page set of the queue manager that executes the command.

### REPLACE and NOREPLACE

Whether the existing definition  (and on z/OS, with the same disposition) is to be replaced with this one. REPLACE is optional. Any object with a different disposition is not changed.

#### REPLACE

The definition replaces any existing definition of the same name. If a definition does not exist, one is created.

#### NOREPLACE

The definition does not replace any existing definition of the same name.



## USERDATA( *string* )

A character string that contains user information pertaining to the application defined in the APPLICID that is to be started. The maximum length is 128 characters.

The meaning of USERDATA is determined by the trigger-monitor application. The trigger monitor provided by IBM MQ simply passes USERDATA to the started application as part of the parameter list. The parameter list consists of the MQTMC2 structure (containing USERDATA), followed by one blank, followed by ENVRDATA with trailing blanks removed.

For IBM MQ message channel agents, the format of this field is a channel name of up to 20 characters. See [Managing objects for triggering](#) for information about what APPLICID to provide to message channel agents.

For Microsoft Windows, the character string must not contain double quotation marks if the process definition is going to be passed to `runmqtrm`.

## DEFINE PSID (define page set and buffer pool) on z/OS

Use the MQSC command DEFINE PSID to define a page set and associated buffer pool.

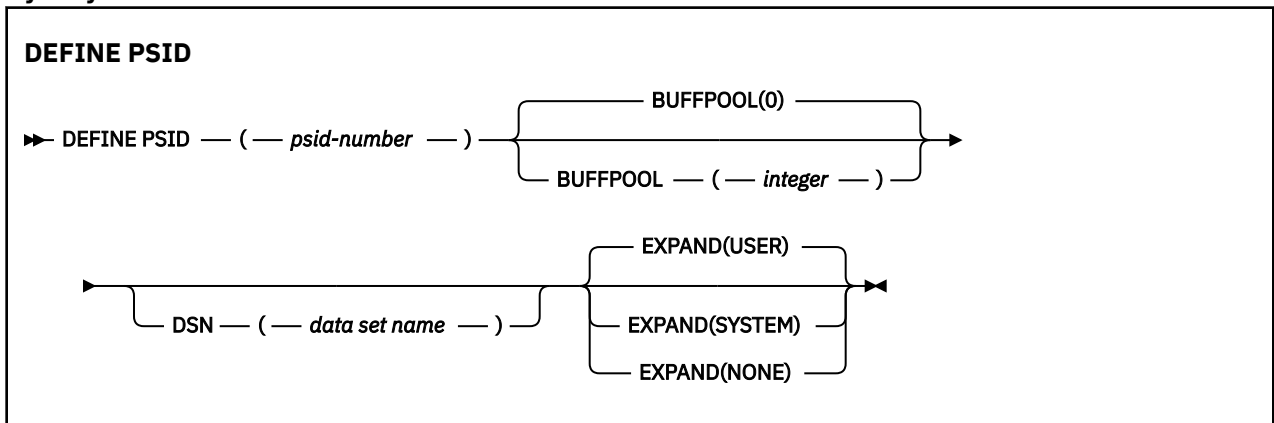
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 1CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DEFINE PSID” on page 545](#)
- [“Parameter descriptions for DEFINE PSID” on page 546](#)

**Synonym:** DEF PSID



### Usage notes for DEFINE PSID

The command can be used in two ways:

- 1. At restart, from the CSQINP1 initialization input data set, to specify your standard page sets:**
  - You cannot specify the DSN keyword if issuing the command from CSQINP1.
  - If more than one DEFINE PSID command is issued for the same page set, only the last one is processed.
- 2. While the queue manager is running, to dynamically add a page set:**
  - The command must specify the DSN keyword and can be issued from either of the following:
    - The z/OS console.

- The command server and command queue by means of CSQUTIL, CSQINPX, or applications.
- The page set identifier (that is the PSID number) may have previously been used by a queue manager. It should therefore be freshly formatted by a FORMAT(RECOVER) statement in CSQUTIL, or formatted by with a FORMAT(REPLACE) in CSQUTIL.
- You cannot dynamically add page set zero.
- The BUFFPOOL parameter can specify a currently unused buffer pool. If the buffer pool was defined in CSQINP1 but not used by any PSID, then the number of buffers specified there is created if the required virtual storage is available. If this is not available, or if the buffer pool was not defined in CSQINP1, the queue manager attempts to allocate 1000 buffers. If this is not possible, 100 buffers are allocated.
- You should update your queue manager started task procedure JCL and your CSQINP1 initialization input data set to include the new page set.

One of the messages [CSQP042I](#) or [CSQP041E](#) is output when the command is complete.

You must use the [ALTER PSID](#) command to dynamically change the expansion method. For example, to change the EXPAND parameter from USER to SYSTEM, issue the following command:

```
ALTER PSID(page set id) EXPAND(SYSTEM)
```

You can use the DISPLAY USAGE TYPE(PAGESET) command to display information about page sets (see [“DISPLAY USAGE \(display usage information\) on z/OS” on page 873](#) ).

## Parameter descriptions for DEFINE PSID

### **(psid-number)**

Identifier of the page set. This is required.

A one-to-one relationship exists between page sets and the VSAM data sets used to store the pages. The identifier consists of a number in the range 00 through 99. It is used to generate a *ddname*, which references the VSAM LDS data set, in the range CSQP0000 through CSQP0099.

The identifier must not be the same as any other page set identifier currently defined on this queue manager.

### **BUFFPOOL( integer )**

The buffer pool number is in the range zero through 99. This is optional. The default is zero.

If the buffer pool has not already been created by a DEFINE BUFFPOOL command, the buffer pool is created with 1000 buffers, and a LOCATION value of BELOW.

If the psid-number is zero, the buffer pool number must be in the range 0 to 15, otherwise the command fails, and the queue manager does not start.

### **DSN( data set name )**

The name of a cataloged VSAM LDS data set. This is optional. There is no default.

### **EXPAND**

Controls how the queue manager should expand a page set when it becomes nearly full, and further pages are required in a page set.

### **USER**

The secondary extent size that was specified when the page set was defined is used. If no secondary extent size was specified, or it was specified as zero, no dynamic page set expansion can take place if page set data set is non-striped.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set. Only one extent is required to reach this size.

## SYSTEM

A secondary extent size that is approximately 10 per cent of the current size of the page set is used. It can be rounded up depending on the characteristics of the DASD.

## NONE

No further page set expansion is to take place.

## DEFINE queues

Use the MQSC **DEFINE** command to define a local, model, or remote queue, or a queue alias, reply-to queue alias, or a queue manager alias.


## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

This section contains the following commands:


- “[DEFINE QALIAS \(define a new alias queue\)](#)” on page 572
- “[DEFINE QLOCAL \(define a new local queue\)](#)” on page 574
- “[DEFINE QMODEL \(define a new model queue\)](#)” on page 577
- “[DEFINE QREMOTE \(create local definition of a remote queue\)](#)” on page 580

Define a reply-to queue or queue manager alias with the “[DEFINE QREMOTE \(create local definition of a remote queue\)](#)” on page 580 command.

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

## Usage notes for DEFINE queues

- Successful completion of the command does not mean that the action completed. To check for true completion, see the [DEFINE queues](#) step in [Checking that async commands for distributed networks have finished](#).
- For local queues

1.  You can define a local queue with QSGDISP (SHARED) even though another queue manager in the queue sharing group already has a local version of the queue. However, when you try to access the locally defined queue, it fails with reason code MQRC\_OBJECT\_NOT\_UNIQUE (2343). A local version of the queue with the same name can be of type QLOCAL, QREMOTE, or QALIAS and has the disposition, QSGDISP (QMGR).

To resolve the conflict, you must delete one of the queues using the **DELETE** command. If the queue you want to delete contains messages, use the PURGE option or remove the messages first using the **MOVE** command.

For example, to delete the QSGDISP (LOCAL) version, which contains messages, and copy those messages to the QSGDISP (SHARED) version, then issue the following commands:

```
MOVE QLOCAL(Queue.1) QSGDISP(PRIVATE) TOQLOCAL(Queue.1) TYPE(ADD)
DELETE QLOCAL(Queue.1) QSGDISP(QMGR)
```

- For alias queues:
  1. `DEFINE QALIAS( aliasqueue ) TARGET( otherqname ) CLUSTER( c )` advertises the queue *otherqname* by the name *aliasqueue*.
  2. `DEFINE QALIAS( aliasqueue ) TARGET( otherqname )` allows a queue advertised by the name *otherqname* to be used on this queue manager by the name *aliasqueue*.

3. TARGTYPE and TARGET are not cluster attributes, that is, they are not shared in a cluster environment.
- For remote queues:
    - DEFINE QREMOTE( *rqueue* ) RNAME( *otherq* ) RQMNAME( *otherqm* ) CLUSTER( *cl* ) advertises this queue manager as a store and forward gateway to which messages for queue *rqueue* can be sent. It has no effect as a reply-to queue alias, except on the local queue manager.  
 DEFINE QREMOTE( *otherqm* ) RNAME( ) RQMNAME( *anotherqm* ) XMITQ( *xq* ) CLUSTER advertises this queue manager as a store and forward gateway to which messages for *anotherqm* can be sent.
    - RQMNAME can itself be the name of a cluster queue manager within the cluster. You can map the advertised queue manager name to another name locally. The pattern is the same as with QALIAS definitions.
    - It is possible for the values of RQMNAME and QREMOTE to be the same if RQMNAME is itself a cluster queue manager. If this definition is also advertised using a CLUSTER attribute, do not choose the local queue manager in the cluster workload exit. If you do so, a cyclic definition results.
    - Remote queues do not have to be defined locally. The advantage of doing so is that applications can refer to the queue by a simple, locally defined name. If you do then the queue name is qualified by the name of the queue manager on which the queue resides. Using a local definition means that applications do not need to be aware of the real location of the queue.
    - A remote queue definition can also be used as a mechanism for holding a queue manager alias definition, or a reply-to queue alias definition. The name of the definition in these cases is:
      - The queue manager name being used as the alias for another queue manager name (queue manager alias), or
      - The queue name being used as the alias for the reply-to queue (reply-to queue alias).

## Parameter descriptions for DEFINE QUEUE and ALTER QUEUE

Table 160 on page 548 shows the parameters that are relevant for each type of queue. There is a description of each parameter after the table.

Table 160. DEFINE and ALTER QUEUE parameters				
Parameter	Local queue	Model queue	Alias queue	Remote queue
ACCTQ	✓	✓		
BOQNAME	✓	✓		
BOTHRESH	✓	✓		
  CAEXPRY	✓	✓	✓	✓
 CFSTRUCT	✓	✓		
CLCHNAME	✓			
CLUSNL	✓		✓	✓
CLUSTER	✓		✓	✓
CLWLPRTY	✓		✓	✓

Table 160. DEFINE and ALTER QUEUE parameters (continued)

Parameter	Local queue	Model queue	Alias queue	Remote queue
<u>CLWLRANK</u>	✓		✓	✓
<u>CLWLUSEQ</u>	✓			
 <u>CMDSCOPE</u>	✓	✓	✓	✓
<u>CUSTOM</u>	✓	✓	✓	✓
<u>DEFBIND</u>	✓		✓	✓
<u>DEFPRESP</u>	✓	✓	✓	✓
<u>DEFPRTY</u>	✓	✓	✓	✓
<u>DEFPSIST</u>	✓	✓	✓	✓
<u>DEFREADA</u>	✓	✓	✓	
<u>DEFSOPT</u>	✓	✓		
<u>DEFTYPE</u>		✓		
<u>DESCR</u>	✓	✓	✓	✓
<u>DISTL</u>	✓	✓		
<u>FORCE</u>	✓		✓	✓
<u>GET</u>	✓	✓	✓	
<u>HARDENBO</u> or <u>NOHARDENBO</u>	✓	✓		
<u>IMGRCOVQ</u>	✓	✓		
<u>INDXTYPE</u>	✓	✓		
<u>INITQ</u>	✓	✓		
<u>LIKE</u>	✓	✓	✓	✓
<u>MAXDEPTH</u>	✓	✓		
<u>MAXFSIZE</u>	✓	✓		
<u>MAXMSGL</u>	✓	✓		
<u>MONQ</u>	✓	✓		
<u>MSGDLVSQ</u>	✓	✓		
<u>NOREPLACE</u>	✓	✓	✓	✓
<u>NPMCLASS</u>	✓	✓		

Table 160. DEFINE and ALTER QUEUE parameters (continued)





Parameter	Local queue	Model queue	Alias queue	Remote queue
<u>PROCESS</u>	✓	✓		
<u>PROPCTL</u>	✓	✓	✓	
<u>PUT</u>	✓	✓	✓	✓
<i>queue-name</i>	✓	✓	✓	✓
<u>QDEPTHHI</u>	✓	✓		
<u>QDEPTHLO</u>	✓	✓		
<u>QDPHIEV</u>	✓	✓		
<u>QDPLOEV</u>	✓	✓		
<u>QDPMAXEV</u>	✓	✓		
 <u>QSGDISP</u>	✓	✓	✓	✓
<u>QSVCI EV</u>	✓	✓		
<u>QSVCI NT</u>	✓	✓		
<u>REPLACE</u>	✓	✓	✓	✓
<u>RETINTVL</u>	✓	✓		
<u>RNAME</u>				✓
<u>RQMNAME</u>				✓
<u>SCOPE</u>	✓		✓	✓
<u>SHARE</u> or <u>NOSHARE</u>	✓	✓		
<u>STATQ</u>	✓	✓		
 <u>STGCLASS</u>	✓	✓		
 <u>STREAMQ</u>	✓	✓		
 <u>STRMQOS</u>	✓	✓		
<u>TARGET</u>			✓	
<u>TARGQ</u>			✓	
<u>TARGETTYPE</u>			✓	

Table 160. DEFINE and ALTER QUEUE parameters (continued)

Parameter	Local queue	Model queue	Alias queue	Remote queue
<u>TRIGDATA</u>	✓	✓		
<u>TRIGDPTH</u>	✓	✓		
<u>TRIGGER or NOTRIGGER</u>	✓	✓		
<u>TRIGMPRI</u>	✓	✓		
<u>TRIGTYPE</u>	✓	✓		
<u>USAGE</u>	✓	✓		
<u>XMITQ</u>				✓

**queue-name**

Local name of the queue, except the remote queue where it is the local definition of the remote queue.

See [Rules for naming IBM MQ objects](#).

**ACCTQ**

Specifies whether accounting data collection is to be enabled for the queue. On z/OS, the data collected is class 3 accounting data (thread-level and queue-level accounting). In order for accounting data to be collected for this queue, accounting data for this connection must also be enabled. Turn on accounting data collection by setting either the **ACCTQ** queue manager attribute, or the options field in the MQCNO structure on the MQCONN call.

**QMGR**

The collection of accounting data is based on the setting of the **ACCTQ** parameter on the queue manager definition.

**ON**

Accounting data collection is enabled for the queue unless the **ACCTQ** queue manager parameter has a value of NONE.



On z/OS systems, you must enable class 3 accounting using the **START TRACE** command.

**OFF**

Accounting data collection is disabled for the queue.

**BOQNAME (queue-name)**

The excessive backout requeue name.

This parameter is supported only on local and model queues.

Use this parameter to set or change the back out queue name attribute of a local or model queue. Apart from allowing its value to be queried, the queue manager does nothing based on the value of this attribute. IBM MQ classes for JMS transfers a message that is backed out the maximum number of times to this queue. The maximum is specified by the **BOTHRESH** attribute.

**BOTHRESH(integer)**

The backout threshold.

This parameter is supported only on local and model queues.

Use this parameter to set or change the value of the back out threshold attribute of a local or model queue. Apart from allowing its value to be queried, the queue manager does nothing based on the value of this attribute. IBM MQ classes for JMS use the attribute to determine how many times to

allow a message to be backed out. When the value is exceeded, the message is transferred to the queue named by the **BOQNAME** attribute.

Specify a value in the range 0 - 999,999,999.

### V 9.3.1 Multi **CAEXPRY(integer)**

The maximum time, expressed in tenths of a second, until a message put using an object handle with this object in the resolution path, becomes eligible for expiry processing.

**Important:** **z/OS** You cannot use the **CAEXPRY** attribute introduced in IBM MQ 9.3.1 with a cluster if the full repository is on z/OS.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

#### *integer*

The value must be in the range one through to 999 999 999.

#### **NOLIMIT**

There is no limit on the expiry time of messages put using this object. This is the default value.

If you set **CAEXPRY** to **NOLIMIT**, you can set the **CUSTOM CAEXPRY** attribute as well.

Note that existing messages in the queue, prior to a change in **CAEXPRY**, are not affected by the change (that is, their expiry time remains intact). Only new messages that are put into the queue after the change in **CAEXPRY** have the new expiry time.

### **z/OS** **CFSTRUCT(structure-name)**

Specifies the name of the coupling facility structure where you want messages stored when you use shared queues.

This parameter is supported only on z/OS for local and model queues.

The name:

- Cannot have more than 12 characters
- Must start with an uppercase letter (A - Z)
- Can include only the characters A - Z and 0 - 9

The name of the queue sharing group to which the queue manager is connected is prefixed to the name you supply. The name of the queue sharing group is always four characters, padded with @ symbols if necessary. For example, if you use a queue sharing group named NY03 and you supply the name PRODUCT7, the resultant coupling facility structure name is NY03PRODUCT7. The administrative structure for the queue sharing group (in this case NY03CSQ\_ADMIN) cannot be used for storing messages.

For **ALTER QLOCAL**, **ALTER QMODEL**, **DEFINE QLOCAL** with **REPLACE**, and **DEFINE QMODEL** with **REPLACE** the following rules apply:

- On a local queue with **QSGDISP(SHARED)**, **CFSTRUCT** cannot change.
- If you change either the **CFSTRUCT** or **QSGDISP** value you must delete and redefine the queue. To preserve any of the messages on the queue you must offload the messages before you delete the queue. Reload the messages after you redefine the queue, or move the messages to another queue.
- On a model queue with **DEFTYPE(SHAREDYN)**, **CFSTRUCT** cannot be blank.
- On a local queue with a **QSGDISP** other than **SHARED**, or a model queue with a **DEFTYPE** other than **SHAREDYN**, the value of **CFSTRUCT** does not matter.

For **DEFINE QLOCAL** with **NOREPLACE** and **DEFINE QMODEL** with **NOREPLACE**, the coupling facility structure:

- On a local queue with **QSGDISP(SHARED)** or a model queue with a **DEFTYPE(SHAREDYN)**, **CFSTRUCT** cannot be blank.
- On a local queue with a **QSGDISP** other than **SHARED**, or a model queue with a **DEFTYPE** other than **SHAREDYN**, the value of **CFSTRUCT** does not matter.



**Note:** Before you can use the queue, the structure must be defined in the coupling facility Resource Management (CFRM) policy data set.

### **CLCHNAME(channel name)**

This parameter is supported only on transmission queues.

**CLCHNAME** is the generic name of the cluster-sender channels that use this queue as a transmission queue. The attribute specifies which cluster-sender channels send messages to a cluster-receiver channel from this cluster transmission queue.

You can also set the transmission queue attribute **CLCHNAME** attribute to a cluster-sender channel manually. Messages that are destined for the queue manager connected by the cluster-sender channel are stored in the transmission queue that identifies the cluster-sender channel. They are not stored in the default cluster transmission queue. If you set the **CLCHNAME** attribute to blanks, the channel switches to the default cluster transmission queue when the channel restarts. The default queue is either `SYSTEM.CLUSTER.TRANSMIT.ChannelName` or `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, depending on the value of the queue manager **DEFCLXQ** attribute.

By specifying asterisks, "\*" in **CLCHNAME**, you can associate a transmission queue with a set of cluster-sender channels. The asterisks can be at the beginning, end, or any number of places in the middle of the channel name string. **CLCHNAME** is limited to a length of 48 characters, `MQ_OBJECT_NAME_LENGTH`. A channel name is limited to 20 characters: `MQ_CHANNEL_NAME_LENGTH`. If you specify an asterisk you must also set the `SHARE` attribute so that multiple channels can concurrently access the transmission queue.

**z/OS** If you specify a "\*" in **CLCHNAME**, to obtain a channel profile name, you must specify the channel profile name within quotation marks. If you do not specify the generic channel name within quotation marks you receive message CSQ9030E.

The default queue manager configuration is for all cluster-sender channels to send messages from a single transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. The default configuration can be modified by changing the queue manager attribute, **DEFCLXQ**. The default value of the attribute is `SCTQ`. You can change the value to `CHANNEL`. If you set the **DEFCLXQ** attribute to `CHANNEL`, each cluster-sender channel defaults to using a specific cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.ChannelName`.

**z/OS** On z/OS, if this parameter is set, the queue:

- Must be shareable, by specifying the queue attribute `SHARE`.
- Must be indexed on the correlation ID by specifying `INDXTYPE(CORRELID)`.
- Must not be a dynamic or a shared queue.

### **z/OS ALW CLUSNL(namelist name)**

The name of the namelist that specifies a list of clusters to which the queue belongs.

This parameter is supported only on alias, local, and remote queues.

Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of **CLUSNL** or **CLUSTER** can be non-blank; you cannot specify a value for both.

On local queues, this parameter cannot be set for the following queues:

- Transmission queues
- `SYSTEM.CHANNEL.xx` queues
- `SYSTEM.CLUSTER.xx` queues
- `SYSTEM.COMMAND.xx` queues
- **z/OS** On z/OS only, `SYSTEM.QSG.xx` queues

This parameter is valid only on the following platforms:

- AIX, Linux, and Windows
- z/OS

**z/OS** **ALW** **CLUSTER(cluster name)**

The name of the cluster to which the queue belongs.

This parameter is supported only on alias, local, and remote queues.

The maximum length is 48 characters conforming to the rules for naming IBM MQ objects. Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of **CLUSNL** or **CLUSTER** can be non-blank; you cannot specify a value for both.

On local queues, this parameter cannot be set for the following queues:

- Transmission queues
- SYSTEM.CHANNEL.xxx queues
- SYSTEM.CLUSTER.xxx queues
- SYSTEM.COMMAND.xxx queues
- **z/OS** On z/OS only, SYSTEM.QSG.xxx queues

This parameter is valid only on the following platforms:

- AIX, Linux, and Windows
- z/OS

**CLWLPRTY(integer)**

Specifies the priority of the queue for the purposes of cluster workload distribution. This parameter is valid only for local, remote, and alias queues. The value must be in the range zero through 9 where zero is the lowest priority and 9 is the highest. For more information about this attribute, see [CLWLPRTY queue attribute](#).

**CLWLRANK (integer)**

Specifies the rank of the queue for the purposes of cluster workload distribution. This parameter is valid only for local, remote, and alias queues. The value must be in the range zero through 9 where zero is the lowest rank and 9 is the highest. For more information about this attribute, see [CLWLRANK queue attribute](#).

**CLWLUSEQ**

Specifies the behavior of an MQPUT operation when the target queue has a local instance and at least one remote cluster instance. The parameter has no effect when the MQPUT originates from a cluster channel. This parameter is valid only for local queues.

**QMGR**

The behavior is as specified by the **CLWLUSEQ** parameter of the queue manager definition.

**ANY**

The queue manager is to treat the local queue as another instance of the cluster queue for the purposes of workload distribution.

**LOCAL**

The local queue is the only target of the MQPUT operation.

**z/OS** **CMDSCOPE**

This parameter applies to z/OS only. It specifies where the command is run when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP or SHARED.

''

The command runs on the queue manager on which it was entered.

### ***QmgrName***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered. You can specify another name, only if you are using a queue sharing group environment and if the command server is enabled.

**\***

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of **\*** is the same as entering the command on every queue manager in the queue sharing group.

### **CUSTOM(*string*)**

The custom attribute for new features.

This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE).

The maximum length is defined by the IBM MQ constant MQ\_CUSTOM\_LENGTH and is currently set to 128 on all platforms.

The CUSTOM attribute is intended to be used with the following IBM MQ attribute.

### **LTS CAEXPRY(*integer*)**

**Note:** **V 9.3.1** The CAEXPRY queue attribute introduced at IBM MQ 9.3.1 replaces the use of the CAEXPRY option in the CUSTOM field. It is not possible to set the CAEXPRY attribute if the CUSTOM field has a CAEXPRY attribute defined in it already. You should alter existing queues to set the new CAEXPRY field and unset the CAEXPRY attribute from the CUSTOM field. For example:

```
ALTER QL(Q1) CAEXPRY(1000) CAEXPRY('')
```

The maximum time, expressed in tenths of a second, until a message put using an object handle with this object in the resolution path, becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

### ***integer***

The value must be in the range one through to 999 999 999.

### **NOLIMIT**

There is no limit on the expiry time of messages put using this object. This is the default value.

Note that existing messages in the queue, prior to a change in **CAEXPRY**, are not affected by the change (that is, their expiry time remains intact). Only new messages that are put into the queue after the change in **CAEXPRY** have the new expiry time.

### **DEFBIND**

Specifies the binding to be used when the application specifies MQ00\_BIND\_AS\_Q\_DEF on the MQOPEN call, and the queue is a cluster queue.

### **OPEN**

The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

### **NOTFIXED**

The queue handle is not bound to any instance of the cluster queue. The queue manager selects a specific queue instance when the message is put using MQPUT. It changes that selection later, if the need arises.

### **GROUP**

Allows an application to request that a group of messages is allocated to the same destination instance.

Multiple queues with the same name can be advertised in a queue manager cluster. An application can send all messages to a single instance, MQ00\_BIND\_ON\_OPEN. It can allow a

workload management algorithm to select the most suitable destination on a per message basis, MQOO\_BIND\_NOT\_FIXED. It can allow an application to request that a group of messages be all allocated to the same destination instance. The workload balancing reselects a destination between groups of messages, without requiring an MQCLOSE and MQOPEN of the queue.

The MQPUT1 call always behaves as if NOTFIXED is specified.

This parameter is valid on all platforms.

#### **DEFPRESP**

Specifies the behavior to be used by applications when the put response type, within the MQPMO options, is set to MQPMO\_RESPONSE\_AS\_Q\_DEF.

##### **SYNC**

Put operations to the queue specifying MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE is specified instead.

##### **ASYNC**

Put operations to the queue specifying MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_ASYNC\_RESPONSE is specified instead; see [MQPMO options \(MQLONG\)](#).

#### **DEFPRTY(*integer*)**

The default priority of messages put on the queue. The value must be in the range 0 - 9. Zero is the lowest priority, through to the **MAXPRTY** queue manager parameter. The default value of **MAXPRTY** is 9.

#### **DEFPSIST**


Specifies the message persistence to be used when applications specify the MQPER\_PERSISTENCE\_AS\_Q\_DEF option.

##### **NO**

Messages on this queue are lost across a restart of the queue manager.

##### **YES**

Messages on this queue survive a restart of the queue manager.

 On z/OS, N and Y are accepted as synonyms of NO and YES.

#### **DEFREADA**

Specifies the default read ahead behavior for non-persistent messages delivered to the client. Enabling read ahead can improve the performance of client applications consuming non-persistent messages.

##### **NO**

Non-persistent messages are not read ahead unless the client application is configured to request read ahead.

##### **YES**

Non-persistent messages are sent to the client before an application requests them. Non-persistent messages can be lost if the client ends abnormally or if the client does not delete all the messages it is sent.

##### **DISABLED**


Read ahead of non-persistent messages is not enabled for this queue. Messages are not sent ahead to the client regardless of whether read ahead is requested by the client application.

#### **DEFSOPT**

The default share option for applications opening this queue for input:

##### **EXCL**

The open request is for exclusive input from the queue.

 On z/OS, EXCL is the default value.

##### **SHARED**

The open request is for shared input from the queue.

 On Multiplatforms, SHARED is the default value.

## DEFTYPE

Queue definition type.

This parameter is supported only on model queues.

## PERMDYN

A permanent dynamic queue is created when an application issues an MQOPEN MQI call with the name of this model queue specified in the object descriptor (MQOD).

▶ **z/OS** On z/OS, the dynamic queue has a disposition of QMGR.

## ▶ **z/OS** SHAREDYN

This option is available on z/OS only.

A permanent dynamic queue is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor (MQOD).

The dynamic queue has a disposition of SHARED.

## TEMPDYN

A temporary dynamic queue is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor (MQOD).

▶ **z/OS** On z/OS, the dynamic queue has a disposition of QMGR.

▶ **z/OS** Do not specify this value for a model queue definition with a **DEFPSIST** parameter of YES.

▶ **z/OS** If you specify this option, do not specify **INDXTYPE**(MSGTOKEN).

## DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the **DISPLAY QUEUE** command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** Use characters that are in the coded character set identifier (CCSID) of this queue manager. If you do not do so and if the information is sent to another queue manager, they might be translated incorrectly.

## ▶ **ALW** DISTL

Sets whether distribution lists are supported by the partner queue manager.

### YES

Distribution lists are supported by the partner queue manager.

### NO

Distribution lists are not supported by the partner queue manager.

**Note:** You do not normally change this parameter, because it is set by the MCA. However you can set this parameter when defining a transmission queue if the distribution list capability of the destination queue manager is known.

This parameter is valid only on AIX, Linux, and Windows.

## FORCE

This parameter applies only to the **ALTER** command on alias, local and remote queues.

Specify this parameter to force completion of the command in the following circumstances.

For an alias queue, if both of the following statements are true:

- The **TARGET** parameter specifies a queue
- An application has this alias queue open

For a local queue, if both of the following statements are true:

- The **NOSHARE** parameter is specified

- More than one application has the queue open for input

**FORCE** is also needed if both of the following statements are true:

- The **USAGE** parameter is changed
- Either one or more messages are on the queue, or one or more applications have the queue open

Do not change the **USAGE** parameter while there are messages on the queue; the format of messages changes when they are put on a transmission queue.

For a remote queue, if both of the following statements are true:

- The **XMITQ** parameter is changed
- One or more applications has this queue open as a remote queue

**FORCE** is also needed if both of the following statements are true:

- Any of the **RNAME**, **RQMNAME**, or **XMITQ** parameters are changed
- One or more applications has a queue open that resolved through this definition as a queue manager alias

**Note:** **FORCE** is not required if this definition is in use as a reply-to queue alias only.

If **FORCE** is not specified in the circumstances described, the command is unsuccessful.

## GET

Specifies whether applications are to be permitted to get messages from this queue:

### ENABLED

Messages can be retrieved from the queue, by suitably authorized applications.

### DISABLED

Applications cannot retrieve messages from the queue.

This parameter can also be changed using the MQSET API call.

## HARDENBO & NOHARDENBO

Specifies whether the count of the number of times that a message was backed out is hardened. When the count is hardened, the value of the **BackoutCount** field of the message descriptor is written to the log before the message is returned by an MQGET operation. Writing the value to the log ensures that the value is accurate across restarts of the queue manager.

This parameter is supported only on local and model queues.


When the backout count is hardened, the performance of MQGET operations for persistent messages on this queue is impacted.

### HARDENBO

The message backout count for messages on this queue is hardened to ensure that the count is accurate.

### NOHARDENBO

The message backout count for messages on this queue is not hardened and might not be accurate over queue manager restarts.

**Note:**  This parameter affects only z/OS. You can set this parameter on Multiplatforms but it is ineffective.

## IMGRCOVQ

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used. Possible values are:

### YES

These queue objects are recoverable.

**NO**

The “rcdmqimg (record media image)” on page 139 and “rcrmqobj (re-create object)” on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

**QMGR**

If you specify QMGR, and the **IMGRCOVQ** attribute for the queue manager specifies YES, these queue objects are recoverable.

If you specify QMGR and the **IMGRCOVQ** attribute for the queue manager specifies NO, the “rcdmqimg (record media image)” on page 139 and “rcrmqobj (re-create object)” on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

QMGR is the default value.

This parameter is not valid on z/OS.

**z/OS INDXTYPE**

The type of index maintained by the queue manager to expedite MQGET operations on the queue. For shared queues, the type of index determines the type of MQGET operations that can be used.

This parameter is supported only on z/OS.

This parameter is supported only on local and model queues.

Messages can be retrieved using a selection criterion only if an appropriate index type is maintained, as the following table shows:

Retrieval selection criterion	Index type required	
	Shared queue	Other queue
None (sequential retrieval)	Any	Any
Message identifier	MSGID or NONE	Any
Correlation identifier	CORRELID	Any
Message and correlation identifiers	MSGID or CORRELID	Any
Group identifier	GROUPID	Any
Grouping	GROUPID	GROUPID
Message token	Not allowed	MSGTOKEN

where the value of **INDXTYPE** parameter has the following values:

**NONE**

No index is maintained. Use NONE when messages are typically retrieved sequentially or use both the message identifier and the correlation identifier as a selection criterion on the MQGET call.

**MSGID**

An index of message identifiers is maintained. Use MSGID when messages are typically retrieved using the message identifier as a selection criterion on the MQGET call with the correlation identifier set to NULL.

**CORRELID**

An index of correlation identifiers is maintained. Use CORRELID when messages are typically retrieved using the correlation identifier as a selection criterion on the MQGET call with the message identifier set to NULL.

**GROUPID**

An index of group identifiers is maintained. Use GROUPID when messages are retrieved using message grouping selection criteria.

**Note:**

1. You cannot set **INDXTYPE** to GROUPID if the queue is a transmission queue.
2. The queue must use a CF structure at CFLEVEL (3), to specify a shared queue with **INDXTYPE(GROUPID)**.

**z/OS MSGTOKEN**

An index of message tokens is maintained. Use MSGTOKEN when the queue is a WLM-managed queue that you are using with the Workload Manager functions of z/OS.

**Note:** You cannot set **INDXTYPE** to MSGTOKEN if:

- The queue is a model queue with a definition type of SHAREDYN
- The queue is a temporary dynamic queue
- The queue is a transmission queue
- You specify **QSGDISP(SHARED)**

For queues that are not shared and do not use grouping or message tokens, the index type does not restrict the type of retrieval selection. However, the index is used to expedite **GET** operations on the queue, so choose the type that corresponds to the most common retrieval selection.

If you are altering or replacing an existing local queue, you can change the **INDXTYPE** parameter only in the cases indicated in the following table:

*Table 162. Index type change permitted depending upon queue-sharing and presence of messages in the queue*

Queue type		NON-SHARED			SHARED	
Queue state		Uncommitted activity	No uncommitted activity, messages present	No uncommitted activity, and empty	Open or messages present	Not open, and empty
Change <b>INDXTYPE</b> from:	To:	Change allowed?				
NONE	MSGID	No	Yes	Yes	No	Yes
NONE	CORRELID	No	Yes	Yes	No	Yes
NONE	MSGTOKEN	No	No	Yes	-	-
NONE	GROUPID	No	No	Yes	No	Yes
MSGID	NONE	No	Yes	Yes	No	Yes
MSGID	CORRELID	No	Yes	Yes	No	Yes
MSGID	MSGTOKEN	No	No	Yes	-	-
MSGID	GROUPID	No	No	Yes	No	Yes
CORRELID	NONE	No	Yes	Yes	No	Yes
CORRELID	MSGID	No	Yes	Yes	No	Yes
CORRELID	MSGTOKEN	No	No	Yes	-	-
CORRELID	GROUPID	No	No	Yes	No	Yes
MSGTOKEN	NONE	No	Yes	Yes	-	-
MSGTOKEN	MSGID	No	Yes	Yes	-	-



Table 162. Index type change permitted depending upon queue-sharing and presence of messages in the queue (continued)

Queue type		NON-SHARED			SHARED	
MSGTOKEN	CORRELID	No	Yes	Yes	-	-
MSGTOKEN	GROUPLD	No	No	Yes	-	-
GROUPLD	NONE	No	No	Yes	No	Yes
GROUPLD	MSGID	No	No	Yes	No	Yes
GROUPLD	CORRELID	No	No	Yes	No	Yes
GROUPLD	MSGTOKEN	No	No	Yes	-	-

### INITQ(string)

The local name of the initiation queue on this queue manager, to which trigger messages relating to this queue are written. See Rules for naming IBM MQ objects.

This parameter is supported only on local and model queues.

### LIKE(qtype-name)

The name of a queue, with parameters that are used to model this definition.

If this field is not completed, the values of undefined parameter fields are taken from one of the following definitions. The choice depends on the queue type:

Table 163. Queue types and their corresponding definitions

Queue type	Definition
Alias queue	SYSTEM.DEFAULT.ALIAS.QUEUE
Local queue	SYSTEM.DEFAULT.LOCAL.QUEUE
Model queue	SYSTEM.DEFAULT.MODEL.QUEUE
Remote queue	SYSTEM.DEFAULT.REMOTE.QUEUE

For example, not completing this parameter is equivalent to defining the following value of **LIKE** for an alias queue:

```
LIKE(SYSTEM.DEFAULT.ALIAS.QUEUE)
```

If you require different default definitions for all queues, alter the default queue definitions instead of using the **LIKE** parameter.

**z/OS** On z/OS, the queue manager searches for an object with the name and queue type you specify with a disposition of QMGR, COPY, or SHARED. The disposition of the **LIKE** object is not copied to the object you are defining.

### Note:

1. **QSGDISP**(GROUP) objects are not searched.
2. **LIKE** is ignored if **QSGDISP**(COPY) is specified.

### **z/OS** **ALW** **MAXDEPTH(integer)**

The maximum number of messages allowed on the queue.

This parameter is supported only on local and model queues.

On the following platforms, specify a value in the range zero through 999999999:

- **ALW** AIX, Linux, and Windows

- **z/OS** z/OS

On any other IBM MQ platform, specify a value in the range zero through 640000.

Other factors can still cause the queue to be treated as full, for example, if there is no further hard disk space available.

If this value is reduced, any messages that are already on the queue that exceed the new maximum remain intact.

### **Multi** **MAXFSIZE**

The maximum size, in megabytes, that a queue file can grow to. It is possible for a queue file to exceed this size if you have configured the value to be lower than the current queue file size.

If that happens the queue file no longer accepts new messages, but allows existing messages to be consumed. When the queue file size has dropped below the configured value, new messages can be put to the queue.

**Note:** This figure can differ from the value of the attribute configured on the queue, because internally the queue manager might need to use a larger block size to reach the chosen size. See [Modifying IBM MQ queue files](#) for more information on changing the size of queue files and block size and granularity.

When the granularity needs changing because this attribute has been increased, warning message AMQ7493W Granularity changed is written to the AMQERR logs. This gives you an indication that you need to plan for the queue to be emptied, in order for IBM MQ to adopt the new granularity.

Specify a value greater than or equal to 20, and less than or equal to 267,386,880.

The default value for this attribute is *DEFAULT*, which equates to a hard-coded value of 2,088,960 MB, the maximum for a queue in versions of IBM MQ prior to IBM MQ 9.1.5.

### **MAXMSGL(integer)**

The maximum length (in bytes) of messages on this queue.

This parameter is supported only on local and model queues.

**ALW** On AIX, Linux, and Windows, specify a value in the range zero to the maximum message length for the queue manager. See the **MAXMSGL** parameter of the ALTER QMGR command, [ALTER QMGR MAXMSGL](#).

**z/OS** On z/OS, specify a value in the range zero through 100 MB (104 857 600 bytes).

Message length includes the length of user data and the length of headers. For messages put on the transmission queue, there are additional transmission headers. Allow an additional 4000 bytes for all the message headers.

If this value is reduced, any messages that are already on the queue with length that exceeds the new maximum are not affected.

Applications can use this parameter to determine the size of buffer for retrieving messages from the queue. Therefore, the value can be reduced only if it is known that this reduction does not cause an application to operate incorrectly.

Note that by adding the digital signature and key to the message, [Advanced Message Security](#) increases the length of the message.

### **MONQ**

Controls the collection of online monitoring data for queues.

This parameter is supported only on local and model queues.

#### **QMGR**

Collect monitoring data according to the setting of the queue manager parameter **MONQ**.

#### **OFF**

Online monitoring data collection is turned off for this queue.

#### **LOW**

If the value of the **MONQ** parameter of the queue manager is not **NONE**, online monitoring data collection is turned on for this queue.

## **MEDIUM**

If the value of the **MONQ** parameter of the queue manager is not NONE, online monitoring data collection is turned on for this queue.

## **HIGH**

If the value of the **MONQ** parameter of the queue manager is not NONE, online monitoring data collection is turned on for this queue.

There is no distinction between the values LOW, MEDIUM, and HIGH. These values all turn data collection on, but do not affect the rate of collection.

When this parameter is used in an **ALTER** queue command, the change is effective only when the queue is next opened.

## **MSGDLVSQ**

Message delivery sequence.

This parameter is supported only on local and model queues.

## **PRIORITY**


Messages are delivered (in response to MQGET API calls) in first-in-first-out (FIFO) order within priority.

## **FIFO**

Messages are delivered (in response to MQGET API calls) in FIFO order. Priority is ignored for messages on this queue.

The message delivery sequence parameter can be changed from PRIORITY to FIFO while there are messages on the queue. The order of the messages already on the queue is not changed. Messages added to the queue later take the default priority of the queue, and so might be processed before some of the existing messages.

If the message delivery sequence is changed from FIFO to PRIORITY, the messages put on the queue while the queue was set to FIFO take the default priority.

**Note:**  If **INDXTYPE**(GROUPID) is specified with **MSGDLVSQ**(PRIORITY), the priority in which groups are retrieved is based on the priority of the first message within each group. The priorities 0 and 1 are used by the queue manager to optimize the retrieval of messages in logical order. The first message in each group must not use these priorities. If it does, the message is stored as if it was priority two.

## **NPMCLASS**


The level of reliability to be assigned to non-persistent messages that are put to the queue:

### **NORMAL**

Non-persistent messages are lost after a failure, or queue manager shutdown. These messages are discarded on a queue manager restart.

### **HIGH**

The queue manager attempts to retain non-persistent messages on this queue over a queue manager restart or switch over.

 You cannot set this parameter on z/OS.

## **PROCESS(string)**

The local name of the IBM MQ process.

This parameter is supported only on local and model queues.

This parameter is the name of a process instance that identifies the application started by the queue manager when a trigger event occurs; see [Rules for naming IBM MQ objects](#).

The process definition is not checked when the local queue is defined, but it must be available for a trigger event to occur.

If the queue is a transmission queue, the process definition contains the name of the channel to be started. This parameter is optional for transmission queues on the following platforms:

- **IBM i** IBM i
- **ALW** AIX, Linux, and Windows
- **z/OS** z/OS

If you do not specify it, the channel name is taken from the value specified for the **TRIGDATA** parameter.

### **PROPCTL**

Property control attribute. The attribute is optional. It is applicable to local, alias, and model queues.

**Note:** If your application is opening an alias queue you must set this value on both the alias and target queues.

**PROPCTL** options are as follows. The options do not affect message properties in the MQMD or MQMD extension.

#### **ALL**

Set ALL so that an application can read all the properties of the message either in MQRFH2 headers, or as properties of the message handle.

The ALL option enables applications that cannot be changed to access all the message properties from MQRFH2 headers. Applications that can be changed, can access all the properties of the message as properties of the message handle.

In some cases, the format of data in MQRFH2 headers in the received message might be different to the format in the message when it was sent.

#### **COMPAT**

Set COMPAT so that unmodified applications that expect JMS-related properties to be in an MQRFH2 header in the message data continue to work as before. Applications that can be changed, can access all the properties of the message as properties of the message handle.

If the message contains a property with a prefix of `mcd.`, `jms.`, `usr.`, or `mqext.`, all message properties are delivered to the application. If no message handle is supplied, properties are returned in an MQRFH2 header. If a message handle is supplied, all properties are returned in the message handle.

If the message does not contain a property with one of those prefixes, and the application does not provide a message handle, no message properties are returned to the application. If a message handle is supplied, all properties are returned in the message handle.

In some cases, the format of data in MQRFH2 headers in the received message might be different to the format in the message when it was sent.

#### **FORCE**

Force all applications to read message properties from MQRFH2 headers.

Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

A valid message handle supplied in the `MsgHandle` field of the `MQGMO` structure on the `MQGET` call is ignored. Properties of the message are not accessible using the message handle.

In some cases, the format of data in MQRFH2 headers in the received message might be different to the format in the message when it was sent.

#### **NONE**

If a message handle is supplied, all the properties are returned in the message handle.

All message properties are removed from the message body before it is delivered to the application.

#### **PUT**

Specifies whether messages can be put on the queue.

**ENABLED**

Messages can be added to the queue (by suitably authorized applications).

**DISABLED**


Messages cannot be added to the queue.

This parameter can also be changed using the MQSET API call.

**QDEPTHHI(integer)**

The threshold against which the queue depth is compared to generate a Queue Depth High event.

This parameter is supported only on local and model queues.

 For more information about the effect that shared queues on z/OS have on this event; see [Shared queues and queue depth events on z/OS](#).


This event indicates that an application put a message on a queue resulting in the number of messages on the queue becoming greater than or equal to the queue depth high threshold. See the **QDPHIEV** parameter.

The value is expressed as a percentage of the maximum queue depth (**MAXDEPTH** parameter), and must be in the range zero through 100 and no less than **QDEPTHLO**.

**QDEPTHLO(integer)**

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

This parameter is supported only on local and model queues.

 For more information about the effect that shared queues on z/OS have on this event; see [Shared queues and queue depth events on z/OS](#).

This event indicates that an application retrieved a message from a queue resulting in the number of messages on the queue becoming less than or equal to the queue depth low threshold. See the **QDPLOEV** parameter.

The value is expressed as a percentage of the maximum queue depth (**MAXDEPTH** parameter), and must be in the range zero through 100 and no greater than **QDEPTHHI**.

**QDPHIEV**

Controls whether Queue Depth High events are generated.

This parameter is supported only on local and model queues.

A Queue Depth High event indicates that an application put a message on a queue resulting in the number of messages on the queue becoming greater than or equal to the queue depth high threshold. See the **QDEPTHHI** parameter.

**ENABLED**

Queue Depth High events are generated.

**DISABLED**

Queue Depth High events are not generated.

**Note:** The value of this parameter can change implicitly.

 On z/OS, shared queues affect the event.

For more information about this event, see [Queue Depth High](#).

**QDPLOEV**

Controls whether Queue Depth Low events are generated.

This parameter is supported only on local and model queues.

A Queue Depth Low event indicates that an application retrieved a message from a queue resulting in the number of messages on the queue becoming less than or equal to the queue depth low threshold. See the **QDEPTHLO** parameter.


**ENABLED**

Queue Depth Low events are generated.

**DISABLED**

Queue Depth Low events are not generated.

**Note:** The value of this parameter can change implicitly.

 On z/OS, shared queues affect the event.

For more information about this event, see [Queue Depth Low](#).

**QDPMAXEV**

Controls whether Queue Full events are generated.

This parameter is supported only on local and model queues.

A Queue Full event indicates that a put to a queue was rejected because the queue is full. The queue depth reached its maximum value.


**ENABLED**

Queue Full events are generated.

**DISABLED**

Queue Full events are not generated.

**Note:** The value of this parameter can change implicitly.

 On z/OS, shared queues affect the event.

For more information about this event, see [Queue Full](#).

 **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object within the group.

<i>Table 164. Object dispositions for <b>QSGDISP</b> options</i>	
<b>QSGDISP</b>	<b>DEFINE</b>
COPY	<p>The object is defined on the page set of the queue manager that executes the command. It uses the <b>QSGDISP (GROUP)</b> object of the same name as the LIKE object.</p> <p>For example, if you issue the following command,</p> <pre>DEFINE QUEUE(<i>q_name</i>) REPLACE QSGDISP(COPY)</pre> <p>the queue manager searches the shared configuration repository for a QUEUE definition called <i>q_name</i>. If a matching QUEUE definition is found, the queue manager creates a local copy of this definition on the queue manager page set.</p> <p>For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.</p>
GROUP	<p>The object definition resides in the shared configuration repository. <b>QSGDISP (GROUP)</b> is allowed only if the queue manager is in a queue sharing group.</p> <p>If the DEFINE for the <b>QSGDISP (GROUP)</b> object is successful, the DEFINE QUEUE(<i>q_name</i>) REPLACE QSGDISP(COPY) command is generated and sent to all active queue managers in the queue sharing group to make or refresh local copies on page set zero.</p> <p>The DEFINE for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>

Table 164. Object dispositions for **QSGDISP** options (continued)

<b>QSGDISP</b>	<b>DEFINE</b>
PRIVATE	Not permitted.
QMGR	The object is defined on the page set of the queue manager that executes the command.
SHARED	<p>This option applies only to local queues. The object is defined in the shared repository. Messages are stored in the coupling facility and are available to any queue manager in the queue sharing group. You can specify SHARED only if:</p> <ul style="list-style-type: none"> <li>• CFSTRUCT is nonblank</li> <li>• INDXTYPE is not MSGTOKEN</li> <li>• The queue is not: <ul style="list-style-type: none"> <li>– SYSTEM.CHANNEL.INITQ</li> <li>– SYSTEM.COMMAND.INPUT</li> </ul> </li> </ul> <p>If the queue is clustered, a command is generated. The command is sent to all active queue managers in the queue sharing group to notify them of this clustered, shared queue.</p>

### **QSVCI EV**

Controls whether Service Interval High or Service Interval OK events are generated.

This parameter is supported only on local and model queues and is ineffective if it is specified on a shared queue.

A Service Interval High event is generated when a check indicates that no messages were retrieved from the queue for at least the time indicated by the **QSVCI NT** parameter.

A Service Interval OK event is generated when a check indicates that messages were retrieved from the queue within the time indicated by the **QSVCI NT** parameter.

**Note:** The value of this parameter can change implicitly. For more information, see the description of the Service Interval High and Service Interval OK events in [Queue Service Interval High](#) and [Queue Service Interval OK](#).

#### **HIGH**

Service Interval High events are generated

#### **OK**

Service Interval OK events are generated

#### **NONE**

No service interval events are generated

### **QSVCI NT(integer)**

The service interval used for comparison to generate Service Interval High and Service Interval OK events.


This parameter is supported only on local and model queues and is ineffective if it is specified on a shared queue.

See the **QSVCI EV** parameter.

The value is in units of milliseconds, and must be in the range zero through 999999999.

### **REPLACE & NOREPLACE**

This option controls whether any existing definition is to be replaced with this one.

**Note:**  On IBM MQ for z/OS, an existing definition is replaced only if it is of the same disposition. Any object with a different disposition is not changed.

## REPLACE

If the object does exist, the effect is like issuing the **ALTER** command without the **FORCE** parameter and with all the other parameters specified. In particular, note that any messages that are on the existing queue are retained.

There is a difference between the **ALTER** command without the **FORCE** parameter, and the **DEFINE** command with the **REPLACE** parameter. The difference is that **ALTER** does not change unspecified parameters, but **DEFINE** with **REPLACE** sets all the parameters. If you use **REPLACE**, unspecified parameters are taken either from the object named on the **LIKE** parameter, or from the default definition, and the parameters of the object being replaced, if one exists, are ignored.

The command fails if both of the following statements are true:

- The command sets parameters that would require the use of the **FORCE** parameter if you were using the **ALTER** command
- The object is open

The **ALTER** command with the **FORCE** parameter succeeds in this situation.

**ALW** If **SCOPE(CELL)** is specified on AIX, Linux, and Windows, and there is already a queue with the same name in the cell directory, the command fails, even if **REPLACE** is specified.

## NOREPLACE

The definition must not replace any existing definition of the object.

## RETINTVL(*integer*)

The number of hours from when the queue was defined, after which the queue is no longer needed. The value must be in the range 0 - 999,999,999.

This parameter is supported only on local and model queues.

The **CRDATE** and **CRTIME** can be displayed using the **DISPLAY QUEUE** command.

This information is available for use by an operator or a housekeeping application to delete queues that are no longer required.

**Note:** The queue manager does not delete queues based on this value, nor does it prevent queues from being deleted if their retention interval is not expired. It is the responsibility of the user to take any required action.

## RNAME(*string*)

Name of remote queue. This parameter is the local name of the queue as defined on the queue manager specified by **RQMNAME**.

This parameter is supported only on remote queues.

- If this definition is used for a local definition of a remote queue, **RNAME** must not be blank when the open occurs.
- If this definition is used for a queue manager alias definition, **RNAME** must be blank when the open occurs.

In a queue manager cluster, this definition applies only to the queue manager that made it. To advertise the alias to the whole cluster, add the **CLUSTER** attribute to the remote queue definition.

- If this definition is used for a reply-to queue alias, this name is the name of the queue that is to be the reply-to queue.

The name is not checked to ensure that it contains only those characters normally allowed for queue names; see [Rules for naming IBM MQ objects](#).

## RQMNAME(*string*)

The name of the remote queue manager on which the queue **RNAME** is defined.

This parameter is supported only on remote queues.

- If an application opens the local definition of a remote queue, **RQMNAME** must not be blank or the name of the local queue manager. When the open occurs, if **XMITQ** is blank there must be a local queue of this name, which is to be used as the transmission queue.



- If this definition is used for a queue manager alias, **RQMNAME** is the name of the queue manager that is being aliased. It can be the name of the local queue manager. Otherwise, if **XMITQ** is blank, when the open occurs there must be a local queue of this name, which is to be used as the transmission queue.
- If **RQMNAME** is used for a reply-to queue alias, **RQMNAME** is the name of the queue manager that is to be the reply-to queue manager.

The name is not checked to ensure that it contains only those characters normally allowed for IBM MQ object names; see [Rules for naming IBM MQ objects](#).

## ALW SCOPE

Specifies the scope of the queue definition.

This parameter is supported only on alias, local, and remote queues.

### QMGR

The queue definition has queue manager scope. This means that the definition of the queue does not extend beyond the queue manager that owns it. You can open a queue for output that is owned by another queue manager in either of two ways:

1. Specify the name of the owning queue manager.
2. Open a local definition of the queue on the other queue manager.

### CELL

The queue definition has cell scope. Cell scope means that the queue is known to all the queue managers in the cell. A queue with cell scope can be opened for output merely by specifying the name of the queue. The name of the queue manager that owns the queue need not be specified.

If there is already a queue with the same name in the cell directory, the command fails. The **REPLACE** option does not affect this situation.

This value is valid only if a name service supporting a cell directory is configured.

**Restriction:** The DCE name service is no longer supported.

This parameter is valid only on AIX, Linux, and Windows.

## SHARE and NOSHARE

Specifies whether multiple applications can get messages from this queue.

This parameter is supported only on local and model queues.

### SHARE

More than one application instance can get messages from the queue.

### NOSHARE

Only a single application instance can get messages from the queue.

## V 9.3.0 V 9.3.0 STATQ

Specifies whether statistics data collection is enabled:

### QMGR

Statistics data collection is based on the setting of the **STATQ** parameter of the queue manager.

### ON

If the value of the **STATQ** parameter of the queue manager is not NONE, statistics data collection for the queue is enabled.

**z/OS** On z/OS systems, you must enable class 5 statistics using the START TRACE command.

### OFF

Statistics data collection for the queue is disabled.

If this parameter is used in an **ALTER** queue command, the change is effective only for connections to the queue manager made after the change to the parameter.

**z/OS** **STGCLASS(string)**

The name of the storage class.

This parameter is supported only on local and model queues.

**Note:** You can change this parameter only if the queue is empty and closed.

This parameter is an installation-defined name. The first character of the name must be uppercase A through Z, and subsequent characters either uppercase A through Z or numeric 0 through 9.

This parameter is valid only on z/OS; see [Storage classes](#).

**V 9.3.0** **STREAMQ**

The name of a secondary queue where a copy of each message is put.



**Attention:** If the user setting the **STREAMQ** attribute does not have the correct authority on the chosen stream queue, the command fails with error message AMQ8135E, or the equivalent message CSQ9016E on z/OS.

In addition, if the stream queue does not exist, error message AMQ8135E (CSQ9016E on z/OS) is returned instead of AMQ8147E, or the equivalent message CSQM125I on z/OS.

**V 9.3.1** **z/OS** See [Streaming queue restrictions](#) for information on when you can set **STREAMQ**.

**V 9.3.0** **STRMQOS**

The quality of service to use when delivering messages to the streaming queue.

The value can be one of:

**BESTEF**

If the original message can be delivered, but the streamed message cannot, the original message is still delivered to its queue.

This is the default value.

**MUSTDUP**

The queue manager ensures that both the original message and the streamed message are successfully delivered to their queues.

If, for some reason, the streamed message cannot be delivered to its queue, then the original message is not delivered to its queue either. The putting application receives an error reason code and must try to put the message again.

**TARGET(string)**

The name of the queue or topic object being aliased; See [Rules for naming IBM MQ objects](#). The object can be a queue or a topic as defined by **TARGETYPE**. The maximum length is 48 characters.

This parameter is supported only on alias queues.

This object needs to be defined only when an application process opens the alias queue.

The **TARGQ** parameter, defined in IBM WebSphere MQ 6.0, is renamed to **TARGET** from version 7.0 and generalized to allow you to specify the name of either a queue or a topic. The default value for **TARGET** is a queue, therefore **TARGET(my\_queue\_name)** is the same as **TARGQ(my\_queue\_name)**. The **TARGQ** attribute is retained for compatibility with your existing programs. If you specify **TARGET**, you cannot also specify **TARGQ**.

**TARGETYPE(string)**

The type of object to which the alias resolves.

**QUEUE (default)**

The alias resolves to a queue.

**TOPIC**

The alias resolves to a topic.

**TRIGDATA(string)**

The data that is inserted in the trigger message. The maximum length of the string is 64 bytes.

This parameter is supported only on local and model queues.

For a transmission queue, you can use this parameter to specify the name of the channel to be started.

This parameter can also be changed using the MQSET API call.

#### **TRIGDPTH(*integer*)**

The number of messages that have to be on the queue before a trigger message is written, if **TRIGTYPE** is DEPTH. The value must be in the range 1 - 999,999,999. The default value is 1.

This parameter is supported only on local and model queues.

This parameter can also be changed using the MQSET API call.

#### **TRIGGER & NOTRIGGER**

Specifies whether trigger messages are written to the initiation queue, named by the **INITQ** parameter, to trigger the application, named by the **PROCESS** parameter:

##### **TRIGGER**

Triggering is active, and trigger messages are written to the initiation queue.

##### **NOTRIGGER**

Triggering is not active, and trigger messages are not written to the initiation queue. This is the default value.

This parameter is supported only on local and model queues.

This parameter can also be changed using the MQSET API call.

#### **TRIGMPRI(*integer*)**

The message priority number that triggers this queue. The value must be in the range zero through to the **MAXPRTY** queue manager parameter; see [“DISPLAY QMGR \(display queue manager settings\)”](#) on page 765 for details. The default value is zero.

This parameter can also be changed using the MQSET API call.

#### **TRIGTYPE**

Specifies whether and under what conditions a trigger message is written to the initiation queue. The initiation queue is (named by the **INITQ** parameter).

This parameter is supported only on local and model queues.

##### **FIRST**

Whenever the first message of priority equal to or greater than the priority specified by the **TRIGMPRI** parameter of the queue arrives on the queue. This is the default value.

##### **EVERY**

Every time a message arrives on the queue with priority equal to or greater than the priority specified by the **TRIGMPRI** parameter of the queue.

##### **DEPTH**

When the number of messages with priority equal to or greater than the priority specified by **TRIGMPRI** is equal to the number indicated by the **TRIGDPTH** parameter.

##### **NONE**

No trigger messages are written.

This parameter can also be changed using the MQSET API call.

#### **USAGE**

Queue usage.

This parameter is supported only on local and model queues.

##### **NORMAL**

The queue is not a transmission queue.

##### **XMITQ**

The queue is a transmission queue, which is used to hold messages that are destined for a remote queue manager. When an application puts a message to a remote queue, the message is stored

on the appropriate transmission queue. It stays there, awaiting transmission to the remote queue manager.

If you specify this option, do not specify values for **CLUSTER** and **CLUSNL**.

**z/OS** Additionally, on z/OS, do not specify **INDXTYPE**(MSGTOKEN) or **INDXTYPE**(GROUPID).

### XMITQ(string)

The name of the transmission queue to be used for forwarding messages to the remote queue. **XMITQ** is used with either remote queue or queue manager alias definitions.

This parameter is supported only on remote queues.

If **XMITQ** is blank, a queue with the same name as **RQMNAME** is used as the transmission queue.

This parameter is ignored if the definition is being used as a queue manager alias and **RQMNAME** is the name of the local queue manager.

It is also ignored if the definition is used as a reply-to queue alias definition.

### Related tasks

[Copying a local queue definition](#)

### DEFINE QALIAS (define a new alias queue)

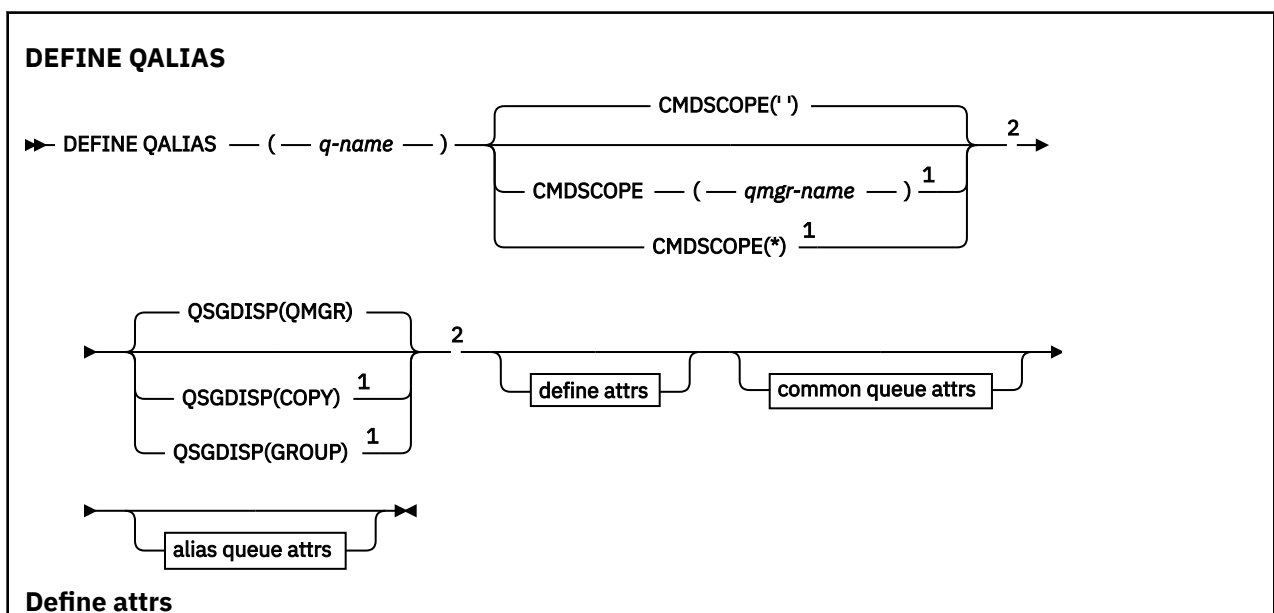
Use **DEFINE QALIAS** to define a new alias queue, and set its parameters.

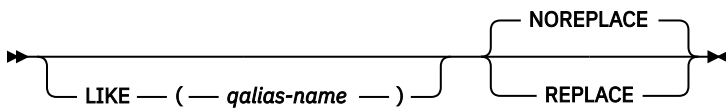
**Note:** An alias queue provides a level of indirection to another queue or a topic object. If the alias refers to a queue, it must be another local or remote queue, defined at this queue manager, or a clustered alias queue defined on another queue manager. It cannot be another alias queue on this queue manager. If the alias refers to a topic, it must be a topic object defined at this queue manager.

- [Syntax diagram](#)
- [“Usage notes for DEFINE queues” on page 547](#)
- [“Parameter descriptions for DEFINE QUEUE and ALTER QUEUE” on page 548](#)

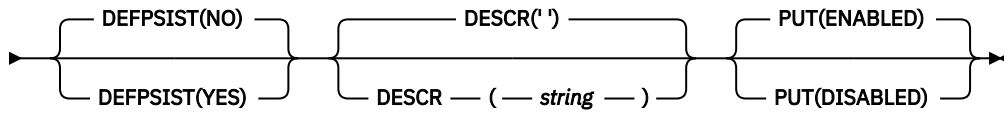
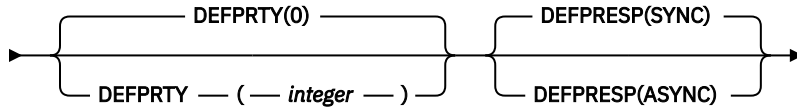
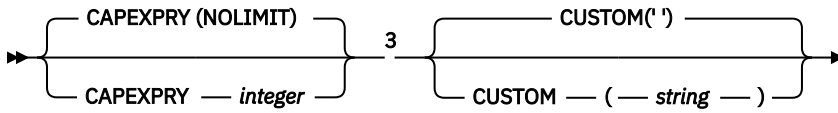
**Synonym:** DEF QA

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).

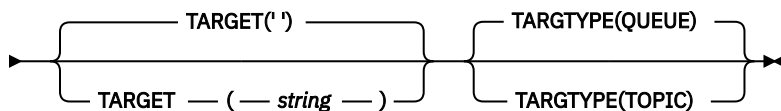
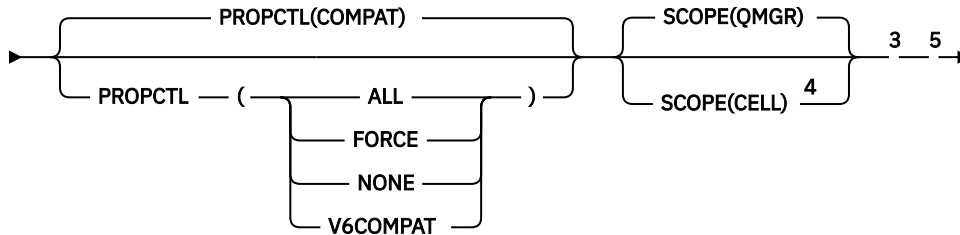
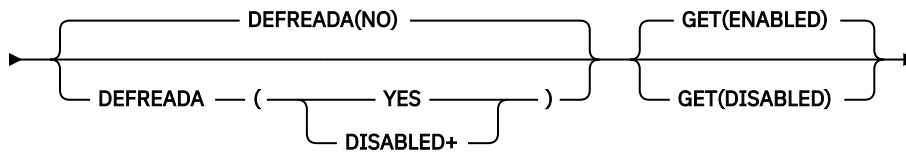
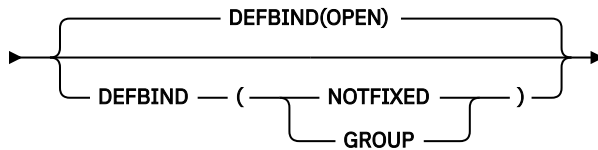
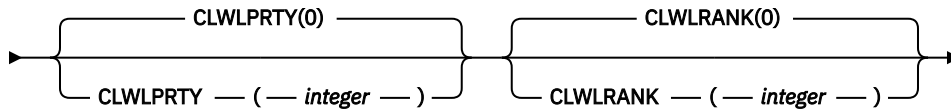
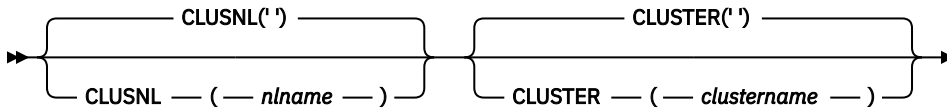




**Common queue attrs**



**Alias queue attrs**



Notes:

<sup>1</sup> Valid only on z/OS.

- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Not valid on z/OS.
- <sup>4</sup> Valid only on AIX, Linux, and Windows.
- <sup>5</sup> The TARGQ parameter, defined in IBM WebSphere MQ 6.0, is renamed to TARGET from version 7.0 and generalized to allow you to specify the name of either a queue or a topic. The default value for TARGET is a queue, therefore TARGET(my\_queue\_name) is the same as TARGQ(my\_queue\_name). The TARGQ attribute is retained for compatibility with your existing programs.

**Related concepts**

[Working with alias queues](#)

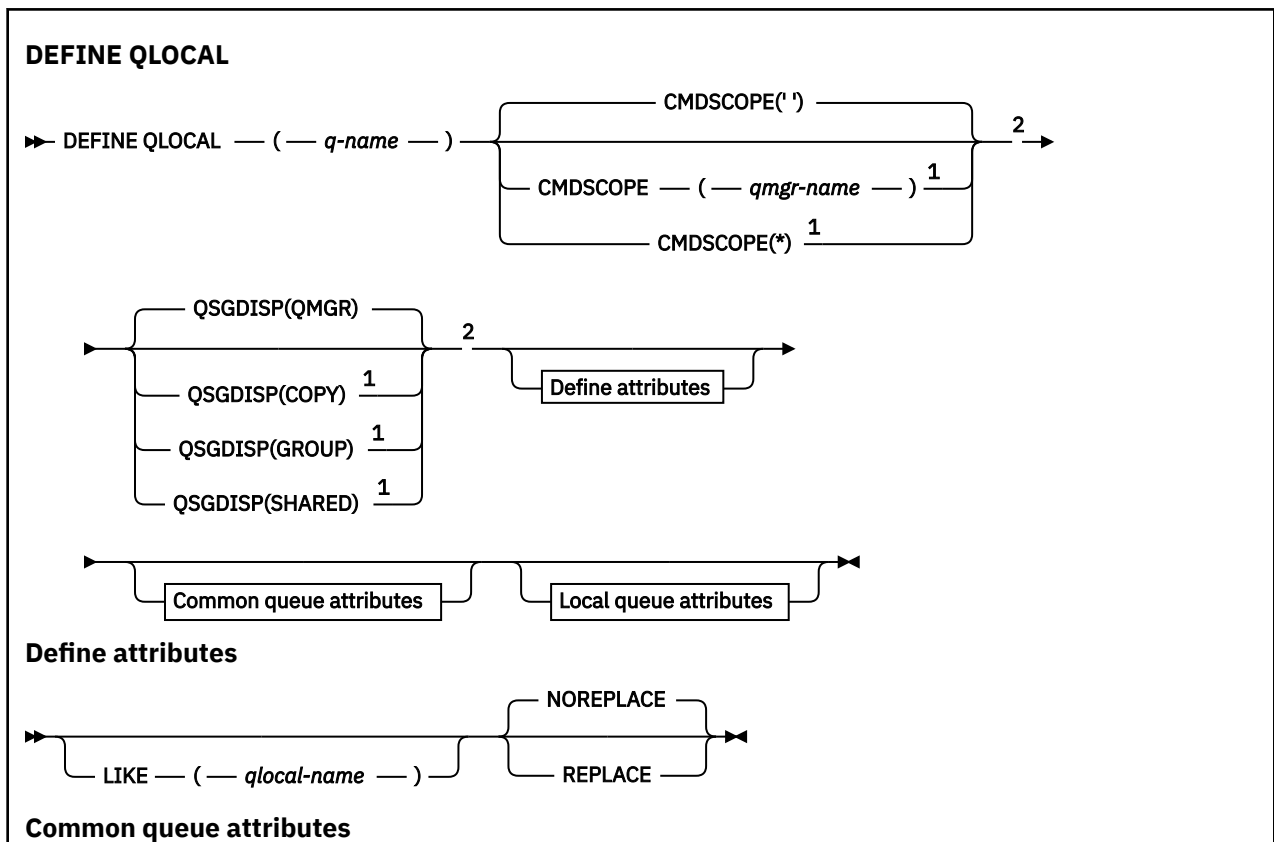
**DEFINE QLOCAL (define a new local queue)**

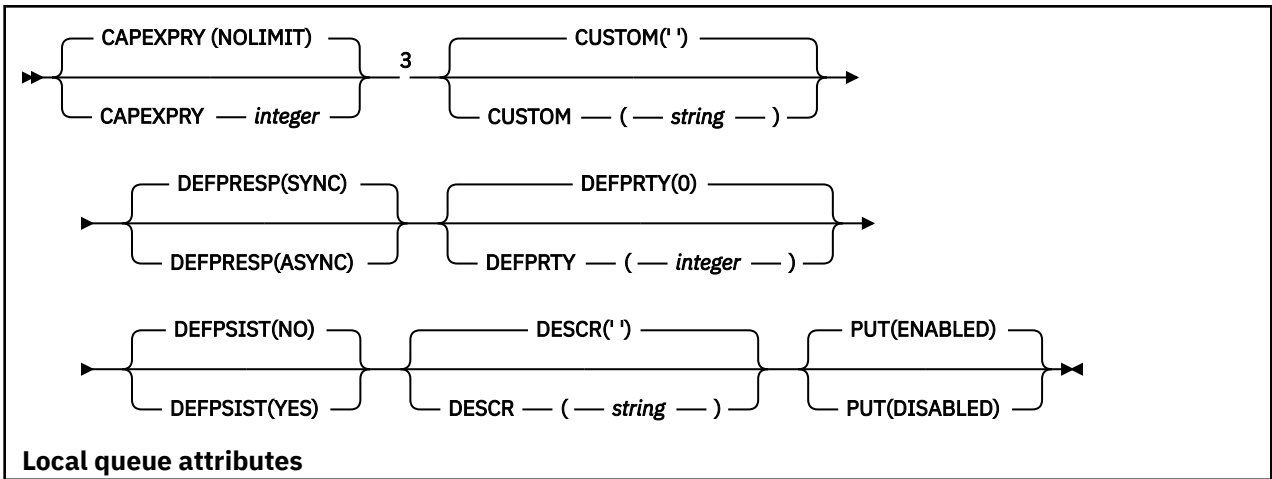
Use **DEFINE QLOCAL** to define a new local queue, and set its parameters.

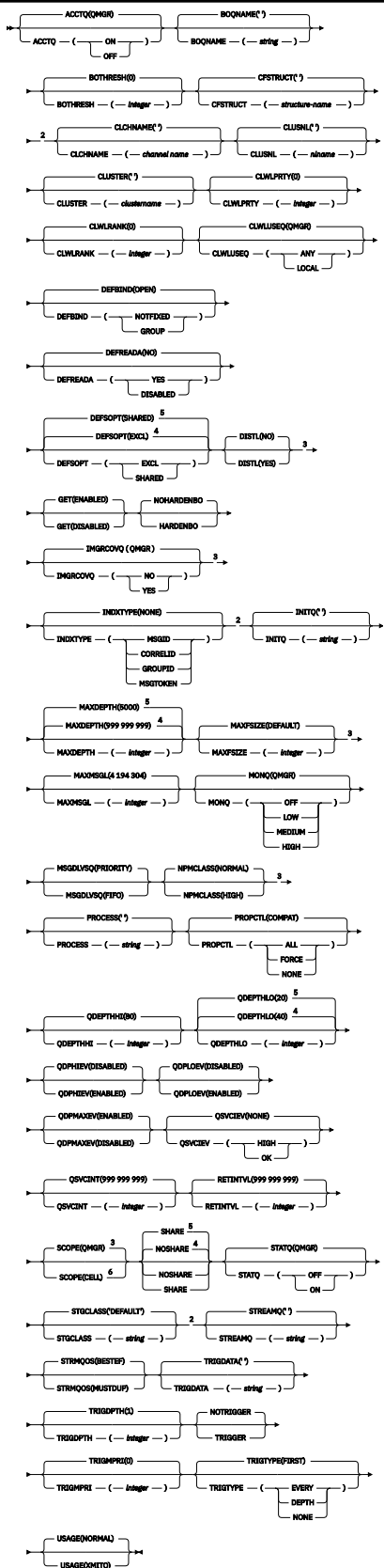
- [Syntax diagram](#)
- [“Usage notes for DEFINE queues” on page 547](#)
- [“Parameter descriptions for DEFINE QUEUE and ALTER QUEUE” on page 548](#)

**Synonym: DEF QL**

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).







Notes:

- 1 Valid only on z/OS and when the queue manager is a member of a queue sharing group.
- 2 Valid only on z/OS.



- <sup>3</sup> Not valid on z/OS.
- <sup>4</sup> Default for z/OS.
- <sup>5</sup> Default for Multiplatforms.
- <sup>6</sup> Valid only on AIX, Linux, and Windows systems.

**Related tasks**

[Defining a local queue](#)

[Changing local queue attributes](#)

**DEFINE QMODEL (define a new model queue)**

Use **DEFINE QMODEL** to define a new model queue, and set its parameters.

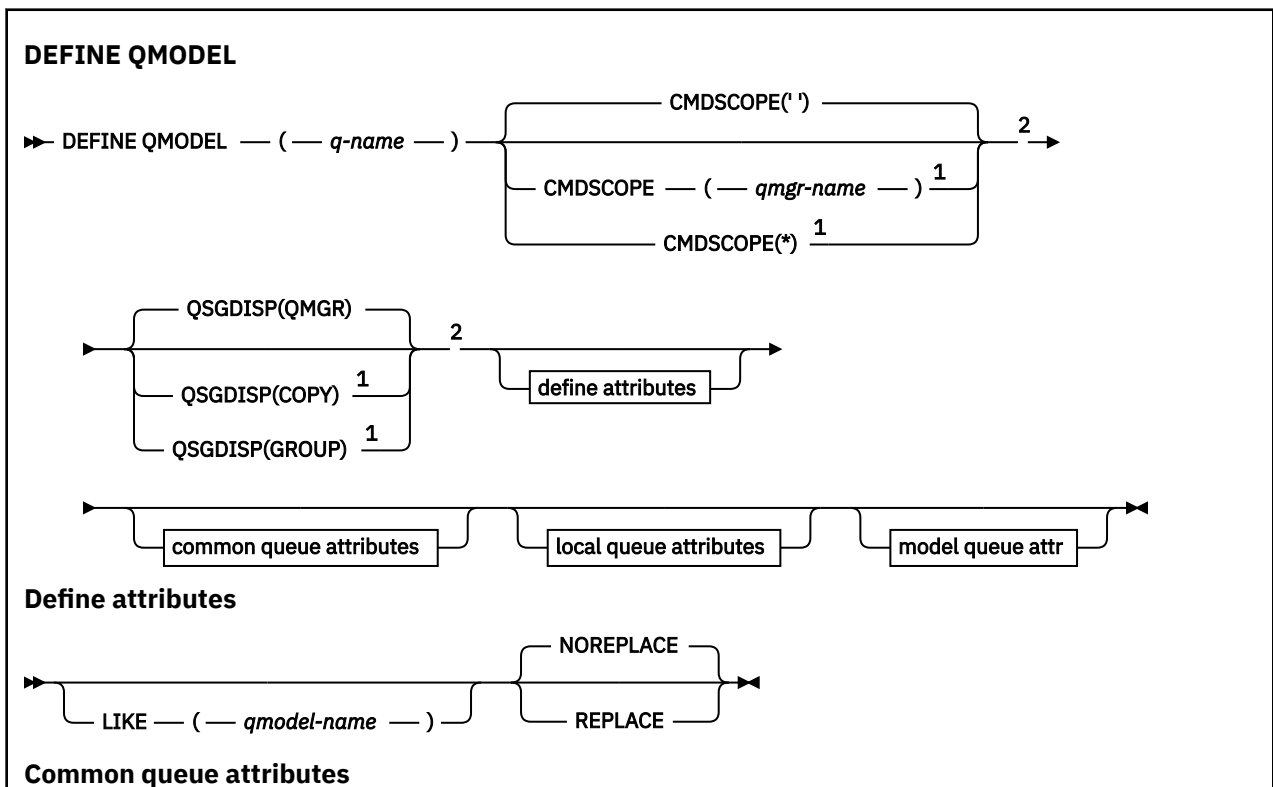
A model queue is not a real queue, but a collection of attributes that you can use when creating dynamic queues with the MQOPEN API call.

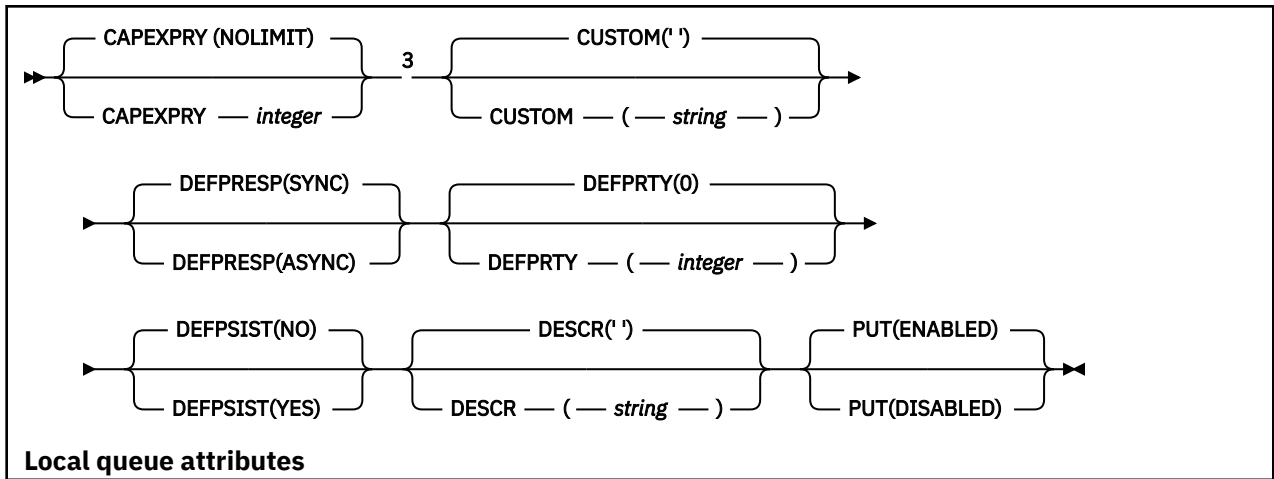
When it has been defined, a model queue (like any other queue) has a complete set of applicable attributes, even if some of these are defaults.

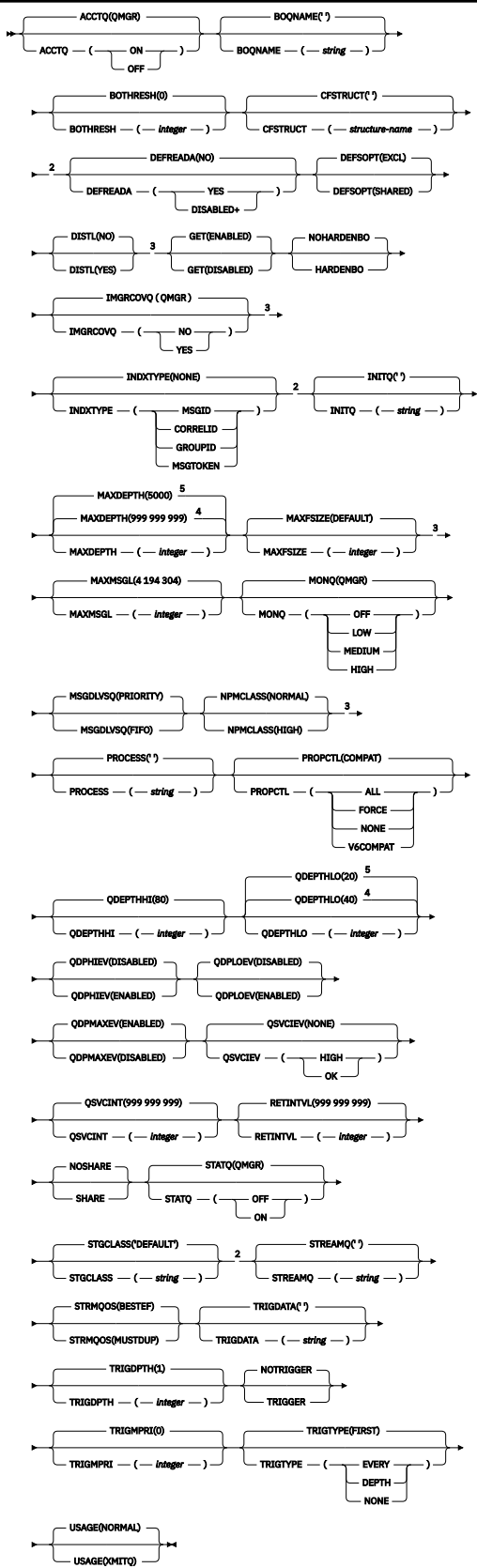
- [Syntax diagram](#)
- [“Usage notes for DEFINE queues” on page 547](#)
- [“Parameter descriptions for DEFINE QUEUE and ALTER QUEUE” on page 548](#)

**Synonym: DEF QM**

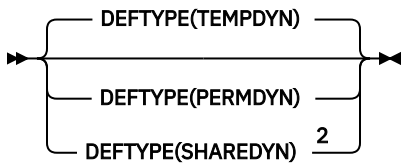
Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).







## Model queue attr



**Notes:**

- <sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>2</sup> Used only on z/OS.
- <sup>3</sup> Not valid on z/OS.
- <sup>4</sup> Default for z/OS.
- <sup>5</sup> Default for Multiplatforms.

**Related concepts**

[Working with model queues](#)

**DEFINE QREMOTE (create local definition of a remote queue)**

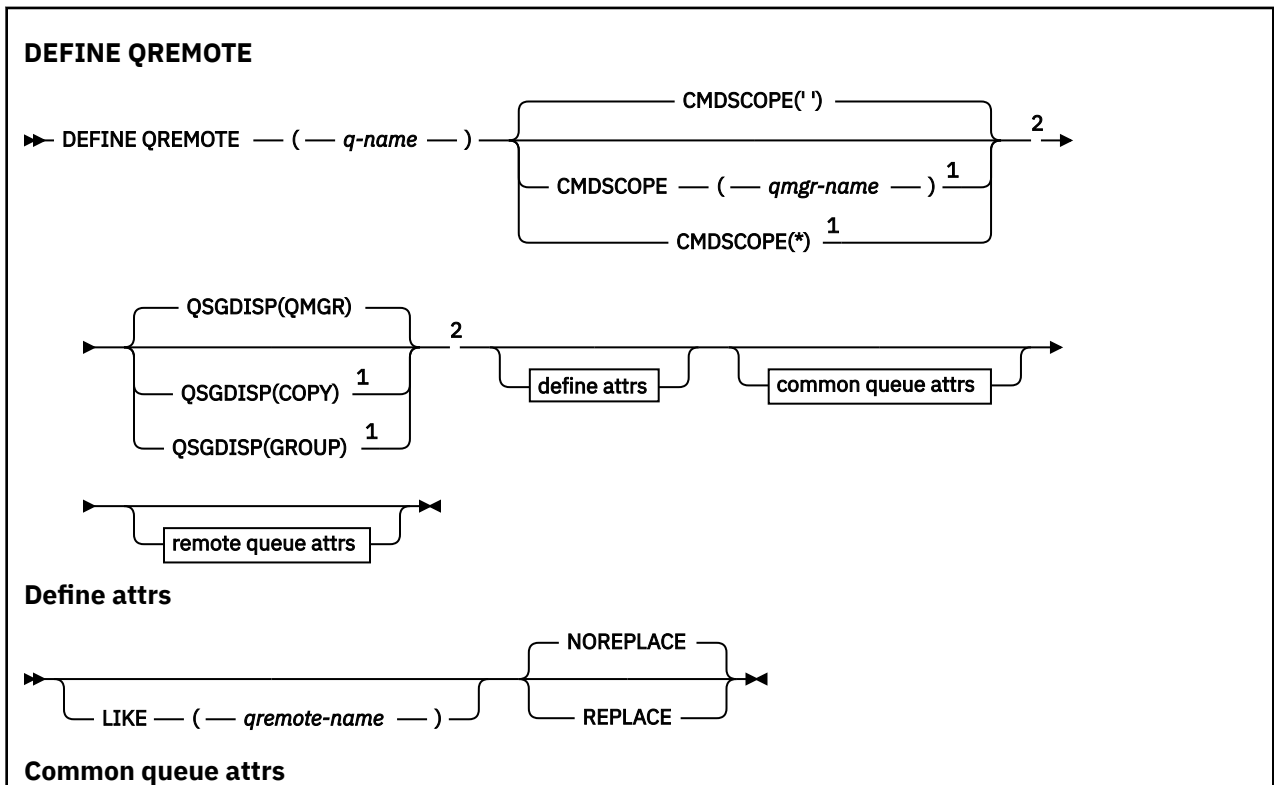
Use DEFINE QREMOTE to define a new local definition of a remote queue, a queue manager alias, or a reply-to queue alias, and to set its parameters.

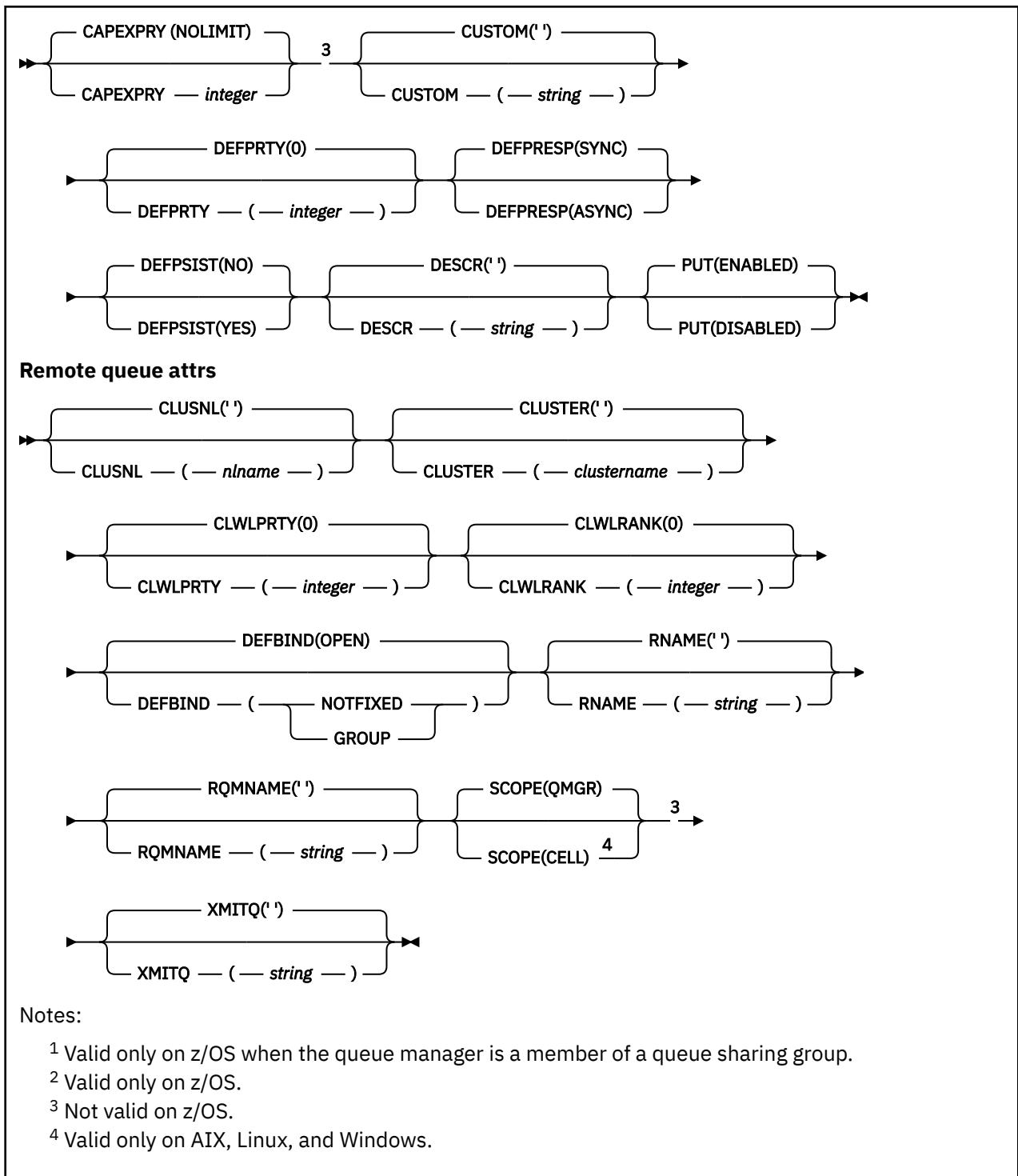
A remote queue is one that is owned by another queue manager that application processes connected to this queue manager need to access.

- [Syntax diagram](#)
- [“Usage notes for DEFINE queues” on page 547](#)
- [“Parameter descriptions for DEFINE QUEUE and ALTER QUEUE” on page 548](#)

**Synonym:** DEF QR

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).





## Multi **DEFINE SERVICE** (create a new service definition) on Multiplatforms

Use the MQSC command **DEFINE SERVICE** to define a new IBM MQ service definition, and set its parameters.

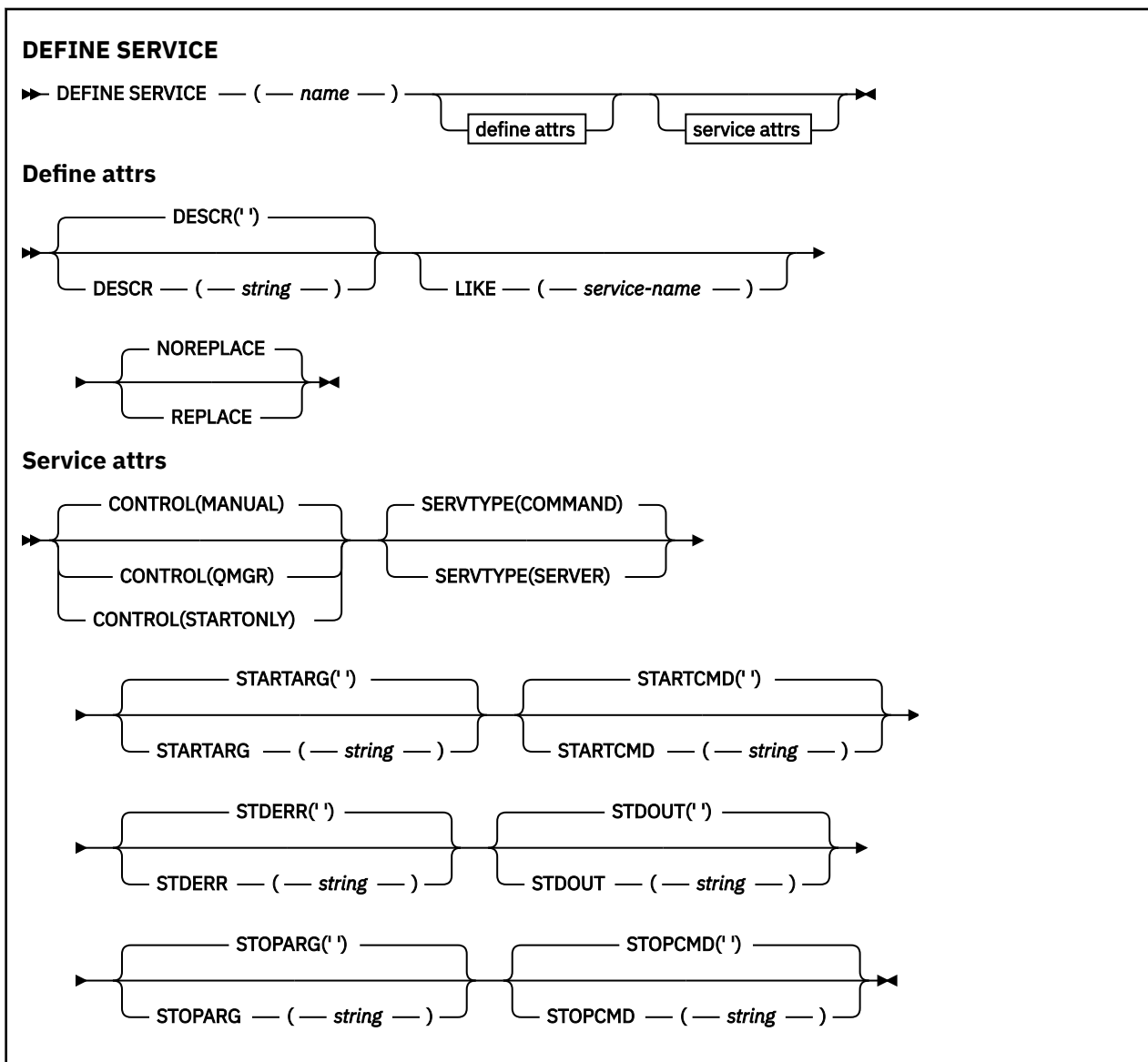
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)

- “Usage notes” on page 582
- “Parameter descriptions for DEFINE SERVICE” on page 583

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



## Usage notes

A service is used to define the user programs that are to be started and stopped when the queue manager is started and stopped. You can also start and stop these programs by issuing the **START SERVICE** and **STOP SERVICE** commands.



**Attention:** This command allows a user to run an arbitrary command with mqm authority. If granted rights to use this command, a malicious or careless user could define a service which damages your systems or data, for example, by deleting essential files.

For more information about services, see [Services](#).

## Parameter descriptions for DEFINE SERVICE

The parameter descriptions apply to the **ALTER SERVICE** and **DEFINE SERVICE** commands, with the following exceptions:

- The **LIKE** parameter applies only to the **DEFINE SERVICE** command.
- The **NOREPLACE** and **REPLACE** parameter applies only to the **DEFINE SERVICE** command.

### **(service-name)**

Name of the IBM MQ service definition (see [Rules for naming IBM MQ objects](#) ).

The name must not be the same as any other service definition currently defined on this queue manager (unless **REPLACE** is specified).

### **CONTROL(string)**

Specifies how the service is to be started and stopped:

#### **MANUAL**

The service is not to be started automatically or stopped automatically. It is to be controlled by use of the **START SERVICE** and **STOP SERVICE** commands.

#### **QMGR**

The service being defined is to be started and stopped at the same time as the queue manager is started and stopped.

#### **STARTONLY**

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

### **DESCR(string)**

Plain-text comment. It provides descriptive information about the service when an operator issues the **DISPLAY SERVICE** command (see [“DISPLAY SERVICE \(display service information\) on Multiplatforms”](#) on page 823).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

### **LIKE(service-name)**

The name of a service the parameters of which are used to model this definition.

This parameter applies only to the **DEFINE SERVICE** command.

If this field is not completed, and you do not complete the parameter fields related to the command, the values are taken from the default definition for services on this queue manager. Not completing this parameter is equivalent to specifying:

```
LIKE(SYSTEM.DEFAULT.SERVICE)
```

A default service is provided but it can be altered by the installation of the default values required. See [Rules for naming IBM MQ objects](#).

### **REPLACE and NOREPLACE**

Whether the existing definition is to be replaced with this one.

This parameter applies only to the **DEFINE SERVICE** command.

#### **REPLACE**

The definition must replace any existing definition of the same name. If a definition does not exist, one is created.

#### **NOREPLACE**

The definition should not replace any existing definition of the same name.

**SERVTYPE**

Specifies the mode in which the service is to run:

**COMMAND**

A command service object. Multiple instances of a command service object can be executed concurrently. You cannot monitor the status of command service objects.

**SERVER**

A server service object. Only one instance of a server service object can be executed at a time. The status of server service objects can be monitored using the **DISPLAY SVSTATUS** command.

**STARTARG(string)**

Specifies the arguments to be passed to the user program at queue manager startup.

**STARTCMD(string)**

Specifies the name of the program which is to run. You must specify a fully qualified path name to the executable program.

**STDERR(string)**

Specifies the path to a file to which the standard error (stderr) of the service program is redirected. If the file does not exist when the service program is started, the file is created. If this value is blank then any data written to stderr by the service program is discarded.

**STDOUT(string)**

Specifies the path to a file to which the standard output (stdout) of the service program is redirected. If the file does not exist when the service program is started, the file is created. If this value is blank then any data written to stdout by the service program is discarded.

**STOPARG(string)**

Specifies the arguments to be passed to the stop program when instructed to stop the service.

**STOPCMD(string)**

Specifies the name of the executable program to run when the service is requested to stop. You must specify a fully qualified path name to the executable program.

Replaceable inserts can be used for any of the **STARTCMD**, **STARTARG**, **STOPCMD**, **STOPARG**, **STDOUT** or **STDERR** strings, for more information, see [Replaceable inserts on service definitions](#).

**Related concepts**

[Working with services](#)

**Related tasks**

[Defining a service object](#)

[Using a server service object](#)

[Using a command service object](#)

**Related reference**

[“ALTER SERVICE \(alter a service definition\) on Multiplatforms” on page 415](#)

Use the MQSC command **ALTER SERVICE** to alter the parameters of an existing IBM MQ service definition.

[“DISPLAY SVSTATUS \(display services status\) on Multiplatforms” on page 843](#)

Use the MQSC command **DISPLAY SVSTATUS** to display status information for one or more services. Only services with a **SERVTYPE** of SERVER are displayed.

[“START SERVICE \(start a service\) on Multiplatforms” on page 955](#)

Use the MQSC command **START SERVICE** to start a service. The identified service definition is started within the queue manager and inherits the environment and security variables of the queue manager.

[“STOP SERVICE \(stop a service\) on Multiplatforms” on page 975](#)



Use the MQSC command **STOP SERVICE** to stop a service.

## **z/OS** DEFINE STGCLASS (define storage class to page set mapping) on z/OS

Use the MQSC command DEFINE STGCLASS to define a storage class to page set mapping.

### Using MQSC commands on z/OS

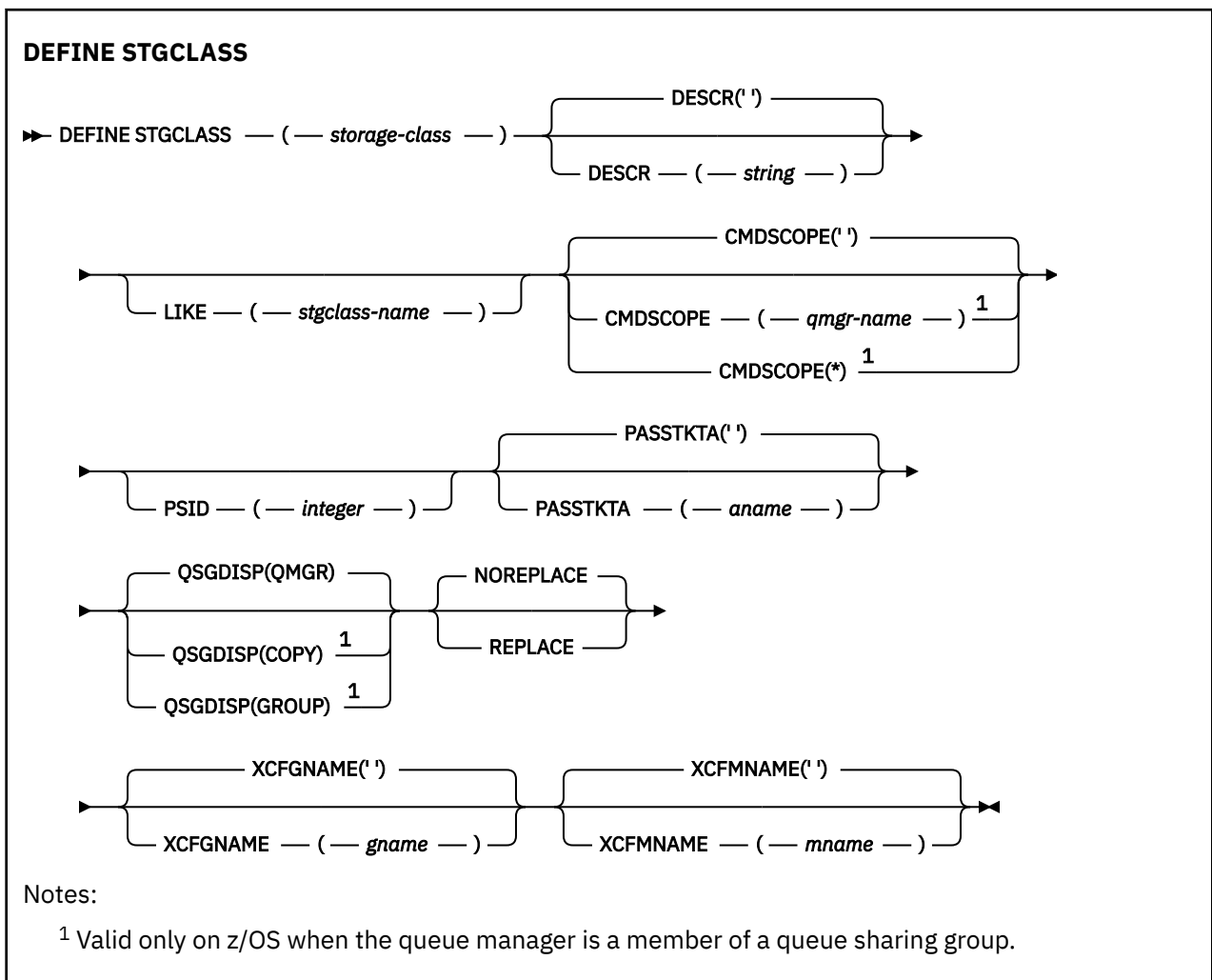
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DEFINE STGCLASS” on page 586](#)
- [“Parameter descriptions for DEFINE STGCLASS” on page 586](#)

**Synonym:** DEF STC

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



## Usage notes for DEFINE STGCLASS

1. The resultant values of XCFGNAME and XCFMNAME must either both be blank or both be nonblank.
2. You can change a storage class only if it is not being used by any queues. To determine whether any queues are using the storage class, you can use the following command:

```
DISPLAY QUEUE(*) STGCLASS(ABC) PSID(n)
```

where 'ABC' is the name of the storage class, and *n* is the identifier of the page set that the storage class is associated with.

This command gives a list of all queues that reference the storage class, and have an active association to page set *n*, and therefore identifies the queues that are actually preventing the change to the storage class. If you do not specify the PSID, you just get a list of queues that are potentially stopping the change.

See the [DISPLAY QUEUE PSID](#) command for more information about active association of a queue to a page set.

## Parameter descriptions for DEFINE STGCLASS

### **(storage-class)**

Name of the storage class.

This name is one to 8 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 through 9.

**Note:** Exceptionally, certain all numeric storage class names are allowed, but are reserved for the use of IBM service personnel.

The storage class must not be the same as any other storage class currently defined on this queue manager.

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

''

The command runs on the queue manager on which it was entered.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

### **DESCR( description )**

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY STGCLASS command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager

**LIKE( *stgclass-name* )**

The name of an object of the same type, with parameters that are used to model this definition.

If this field is not completed, and you do not complete the parameter fields related to the command, the values are taken from the default definition for this object.

Not completing this parameter is equivalent to specifying:

```
LIKE(SYSTEMST)
```

This default storage class definition can be altered by your installation to the default values required.

The queue manager searches for an object with the name you specify and a disposition of QMGR or COPY. The disposition of the LIKE object is not copied to the object you are defining.

**Note:**

1. QSGDISP (GROUP) objects are not searched.
2. LIKE is ignored if QSGDISP(COPY) is specified.

**PASSTKTA( *application name* )**

The application name that is passed to RACF when authenticating the PassTicket specified in the MQIIH header.

**PSID( *integer* )**

The page set identifier that this storage class is to be associated with.

**Note:** No check is made that the page set has been defined; an error is raised only when you try to put a message to a queue that specifies this storage class (MQRC\_PAGESET\_ERROR).

The string consists of two numeric characters, in the range 00 through 99. See [“DEFINE PSID \(define page set and buffer pool\) on z/OS”](#) on page 545.

**QSGDISP**

Specifies the disposition of the object in the group.

<i>Table 165. Object dispositions for QSGDISP options</i>	
<b>QSGDISP</b>	<b>DEFINE</b>
COPY	<p>The object is defined on the page set of the queue manager that executes the command. It uses the <b>QSGDISP (GROUP)</b> object of the same name as the LIKE object.</p> <p>For example, if you issue the following command,</p> <pre>DEFINE STGCLASS(<i>storage_class_name</i>) REPLACE QSGDISP(COPY)</pre> <p>the queue manager searches the shared configuration repository for an STGCLASS definition called <i>storage_class_name</i>. If a matching STGCLASS definition is found, the queue manager creates a local copy of this definition on the queue manager page set.</p> <p>For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.</p>

Table 165. Object dispositions for **QSGDISP** options (continued)

<b>QSGDISP</b>	<b>DEFINE</b>
<b>GROUP</b>	<p>The object definition resides in the shared configuration repository. <b>QSGDISP (GROUP)</b> is allowed only if the queue manager is in a queue sharing group.</p> <p>If the DEFINE for the <b>QSGDISP (GROUP)</b> object is successful, the DEFINE <code>STGCLASS(<i>storage_class_name</i>) REPLACE QSGDISP(COPY)</code> command is generated and sent to all active queue managers in the queue sharing group to make or refresh local copies on page set zero.</p> <p>The DEFINE for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
<b>PRIVATE</b>	Not permitted.
<b>QMGR</b>	The object is defined on the page set of the queue manager that executes the command.

### **REPLACE and NOREPLACE**

Whether the existing definition, and with the same disposition, is to be replaced with this one. Any object with a different disposition is not changed.

#### **REPLACE**

The definition replaces any existing definition of the same name. If a definition does not exist, one is created.

If you use the REPLACE option, all queues that use this storage class must be temporarily altered to use another storage class while the command is issued.

#### **NOREPLACE**

The definition does not replace any existing definition of the same name.

### **XCFGNAME( *group name* )**

If you are using the IMS bridge, this name is the name of the XCF group to which the IMS system belongs. (This name is the group name specified in the IMS parameter list.)

This name is 1 - 8 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 - 9.

### **XCFMNAME( *member name* )**

If you are using the IMS bridge, this name is the XCF member name of the IMS system within the XCF group specified in XCFGNAME. (This name is the member name specified in the IMS parameter list.)


This name is 1 - 16 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 - 9.

## **DEFINE SUB (create a durable subscription)**

Use **DEFINE SUB** to allow an existing application to participate in a publish/subscribe application by allowing the administrative creation of a durable subscription.

### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DEFINE SUB” on page 591](#)

- [“Parameter descriptions for DEFINE SUB” on page 591](#)

**Synonym: DEF SUB**

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).



<sup>2</sup> Valid only on z/OS.

<sup>3</sup> At least one of **TOPICSTR** and **TOPICOBJ** must be present on **DEFINE**.

## Usage notes for **DEFINE SUB**

- You must provide the following information when you define a subscription:
  - The **SUBNAME**
  - A destination for messages
  - The topic to which the subscription applies
- You can provide the topic name in the following ways:

### **TOPICSTR**

The topic is fully specified as the **TOPICSTR** attribute.

### **TOPICOBJ**

The topic is obtained from the **TOPICSTR** attribute of the named topic object. The named topic object is retained as the **TOPICOBJ** attribute of the new subscription. This method is provided to help you enter long topic strings through an object definition.

### **TOPICSTR and TOPICOBJ**

The topic is obtained by the concatenation of the **TOPICSTR** attribute of the named topic object and the value of **TOPICSTR** (see the MQSUB API specification for concatenation rules). The named topic object is retained as the **TOPICOBJ** attribute of the new subscription.

- If you specify **TOPICOBJ**, the parameter must name an IBM MQ topic object. The existence of the named topic object is checked at the time the command processes.
- You can explicitly specify the destination for messages through the use of the **DEST** and **DESTQMGR** keywords.

You must provide the **DEST** keyword for the default option of **DESTCLAS (PROVIDED)**; if you specify **DESTCLAS (MANAGED)**, a managed destination is created on the local queue manager, so you cannot specify either the **DEST** or **DESTQMGR** attribute. For more information, see [Managed queues and publish/subscribe](#).

-  On z/OS only, at the time the **DEF SUB** command processes, no check is performed that the named **DEST** or **DESTQMGR** exists.

These names are used at publishing time as the `ObjectName` and `ObjectQMgrName` for an MQOPEN call. These names are resolved according to the IBM MQ name resolution rules.

- When a subscription is defined administratively using MQSC or PCF commands, the selector is not validated for invalid syntax. The **DEFINE SUB** command has no equivalent to the MQRC\_SELECTION\_NOT\_AVAILABLE reason code that can be returned by the MQSUB API call.
- **TOPICOBJ**, **TOPICSTR**, **WSHEMA**, **SELECTOR**, **SUBSCOPE**, **SUBLEVEL**, and **DESTCLAS** cannot be changed with **DEFINE REPLACE**.
- When a publication has been retained, it is no longer available to subscribers at higher levels because it is republished at PubLevel 1.
- Successful completion of the command does not mean that the action completed. To check for true completion, see the [DEFINE SUB step in Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for **DEFINE SUB**

*(string)*

A mandatory parameter. Specifies the unique name for this subscription, see **SUBNAME** property.





If this field is not supplied, and you do not complete the parameter fields related to the command, the values are taken from the default definition for subscriptions on this queue manager. Not completing this parameter is equivalent to specifying:

```
LIKE (SYSTEM.DEFAULT.SUB)
```

**PSPROP**

The manner in which publish subscribe related message properties are added to messages sent to this subscription.

**NONE**

Do not add publish subscribe properties to the message.

**COMPAT**

Publish/subscribe properties are added within an MQRFH version 1 header unless the message was published in PCF format.

**MSGPROP**

Publish/subscribe properties are added as message properties.

**RFH2**

Publish/subscribe properties are added within an MQRFH version 2 header.

**PUBACCT(*string*)**

Accounting token passed by the subscriber, for propagation into messages published to this subscription in the AccountingToken field of the MQMD.

If this byte string is enclosed in quotation marks, characters in the range A-F must be specified in uppercase.

**PUBAPPID(*string*)**

Identity data passed by the subscriber, for propagation into messages published to this subscription in the AppIdentityData field of the MQMD.

**PUBPRTY**

The priority of the message sent to this subscription.

**AS PUB**

Priority of the message sent to this subscription is taken from the priority supplied in the published message.

**AS QDEF**

Priority of the message sent to this subscription is taken from the default priority of the queue defined as a destination.

**(*integer*)**

An integer providing an explicit priority for messages published to this subscription.

**REPLACE and NOREPLACE**

This parameter controls whether any existing definition is to be replaced with this one.

**REPLACE**

The definition replaces any existing definition of the same name. If a definition does not exist, one is created.

You cannot change **TOPICOBJ**, **TOPICSTR**, **WSHEMA**, **SELECTOR**, **SUBSCOPE**, or **DESTCLAS** with **DEFINE REPLACE**.

**NOREPLACE**

The definition does not replace any existing definition of the same name.

**REQONLY**

Indicates whether the subscriber polls for updates using the MQSUBRQ API call, or whether all publications are delivered to this subscription.

**NO**

All publications on the topic are delivered to this subscription. This is the default value.

**YES**

Publications are only delivered to this subscription in response to an MQSUBRQ API call.

This parameter is equivalent to the subscribe option MQSO\_PUBLICATIONS\_ON\_REQUEST.

**SELECTOR(string)**

A selector that is applied to messages published to the topic.

**SUBLEVEL(integer)**

The level within the subscription hierarchy at which this subscription is made. The range is zero through 9.

**SUBSCOPE**

Determines whether this subscription is forwarded to other queue managers, so that the subscriber receives messages published at those other queue managers.

**ALL**

The subscription is forwarded to all queue managers directly connected through a publish/subscribe collective or hierarchy.

**QMGR**

The subscription forwards messages published on the topic only within this queue manager.

**Note:** Individual subscribers can only restrict **SUBSCOPE**. If the parameter is set to ALL at topic level, then an individual subscriber can restrict it to QMGR for this subscription. However, if the parameter is set to QMGR at topic level, then setting an individual subscriber to ALL has no effect.

**SUBNAME**

The application's unique subscription name that is associated with the handle. This parameter is relevant only for handles of subscriptions to topics. It is not returned for other handles. Not all subscriptions will have a subscription name.

**SUBUSER(string)**

Specifies the user ID that is used for security checks that are performed to ensure that publications can be put to the destination queue associated with the subscription. This ID is either the user ID associated with the creator of the subscription or, if subscription takeover is permitted, the user ID that last took over the subscription. The length of this parameter must not exceed 12 characters.

**TOPICOBJ(string)**

The name of a topic object used by this subscription.

**TOPICSTR(string)**

Specifies a fully qualified topic name, or a topic set using wildcard characters for the subscription.

**USERDATA(string)**

Specifies the user data associated with the subscription. The string is a variable length value that can be retrieved by the application on an MQSUB API call and passed in a message sent to this subscription as a message property. The **USERDATA** is stored in the RFH2 header in the mqps folder with the key Sud.

An IBM MQ classes for JMS application can retrieve the subscription user data from the message by using the constant JMS\_IBM\_SUBSCRIPTION\_USER\_DATA. For more information, see [Retrieval of user subscription data](#).

**VARUSER**

Specifies whether a user other than the subscription creator can connect to and take over ownership of the subscription.

**ANY**

Any user can connect to and takeover ownership of the subscription.

**FIXED**

Takeover by another USERID is not permitted.

**WSHEMA**

The schema to be used when interpreting any wildcard characters in the topic string.

**CHAR**

Wildcard characters represent portions of strings.

## TOPIC

Wildcard characters represent portions of the topic hierarchy.

### Related tasks

[Defining an administrative subscription](#)

[Changing local subscription attributes](#)


[Copying a local subscription definition](#)

## DEFINE TOPIC (define a new administrative topic)

Use **DEFINE TOPIC** to define a new IBM MQ administrative topic in a topic tree, and set its parameters.

### Using MQSC commands

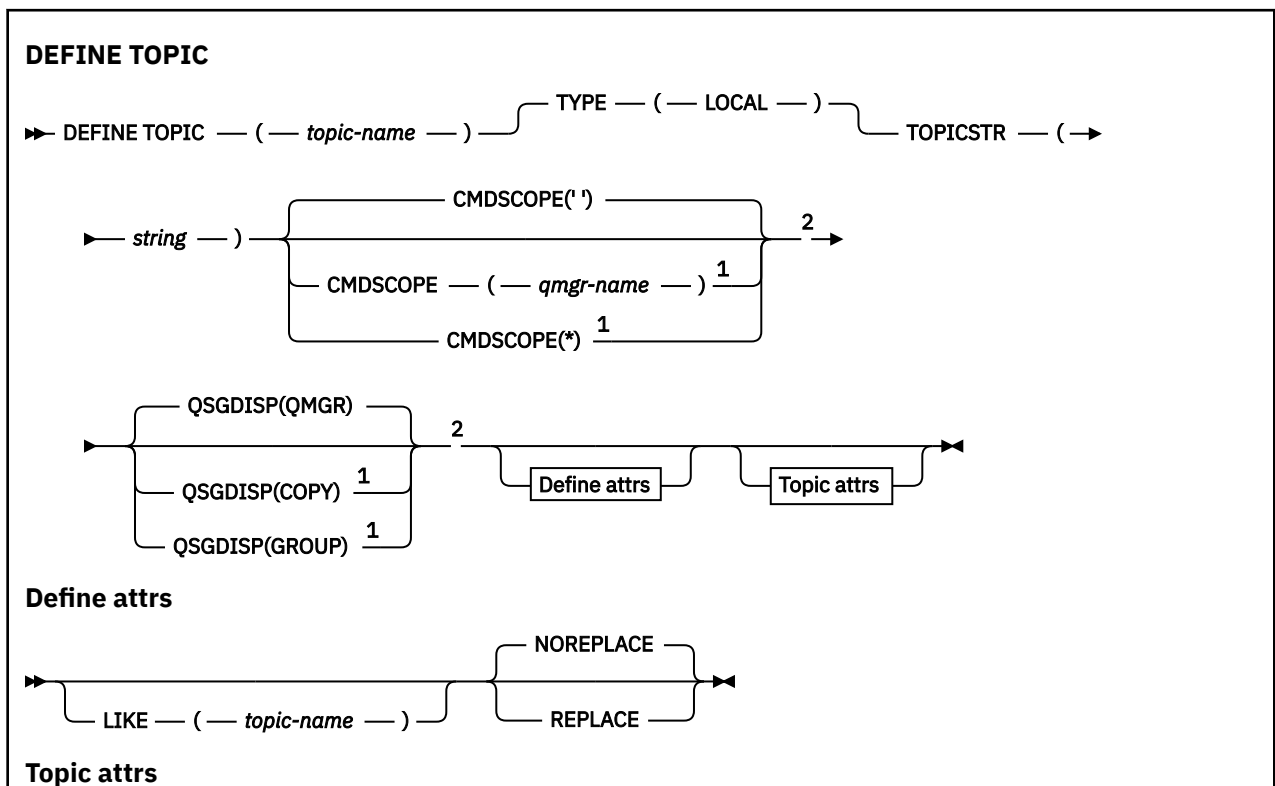
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

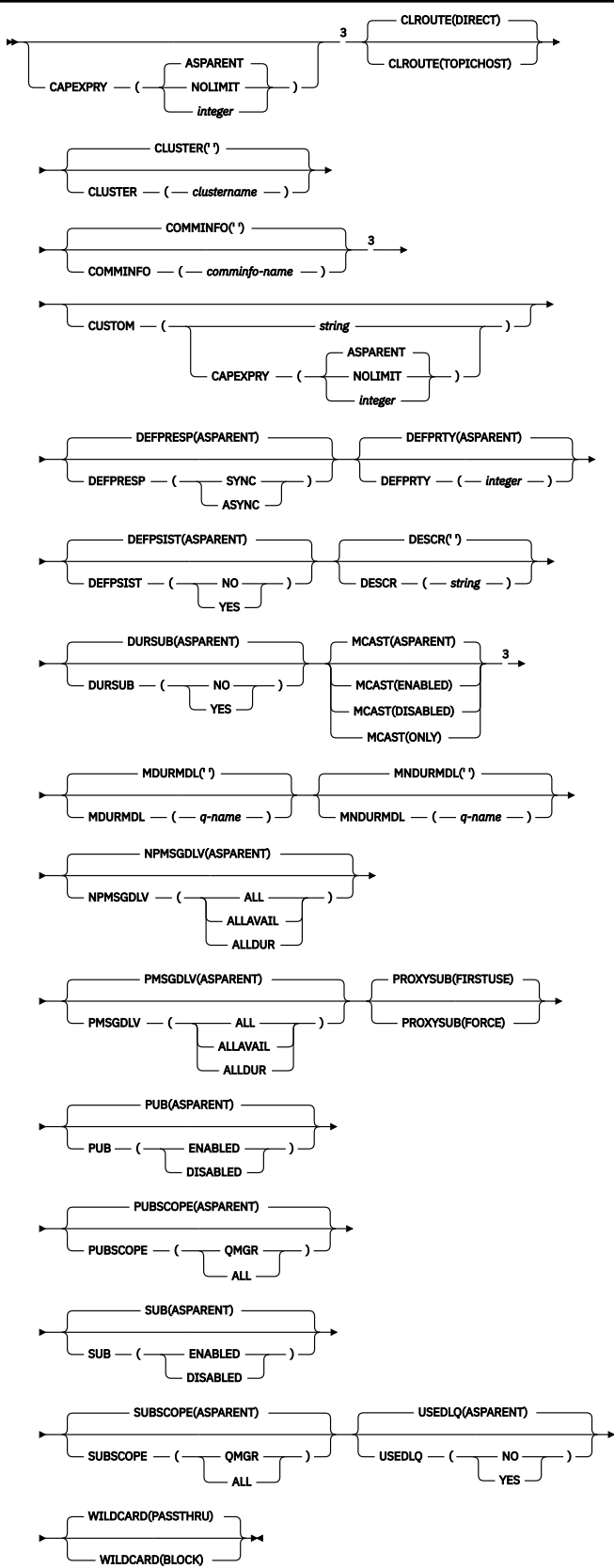
 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DEFINE TOPIC” on page 597](#)
- [“Parameter descriptions for DEFINE TOPIC” on page 597](#)

**Synonym:** DEF TOPIC

Values shown above the main line in the syntax diagram are the defaults supplied with IBM MQ, but your installation might have changed them. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).





Notes:

- 1 Valid only on z/OS when the queue manager is a member of a queue sharing group.
- 2 Valid only on z/OS.

<sup>3</sup> Not valid on z/OS.

## Usage notes for DEFINE TOPIC

- When an attribute has the value ASPARENT, the value is taken from the setting of the first parent administrative node that is found in the topic tree. Administered nodes are based on either locally defined topic objects or remotely defined cluster topics when participating in a publish/subscribe cluster. If the first parent topic object also has the value ASPARENT, the next object is looked for. If every object that is found, when looking up the tree, uses ASPARENT, the values are taken from the SYSTEM.BASE.TOPIC, if it exists. If SYSTEM.BASE.TOPIC does not exist, the values are the same as the values supplied with IBM MQ in the definition of the SYSTEM.BASE.TOPIC.
- The ASPARENT attribute is applied at each queue manager in the cluster collective by inspecting the set of local definitions and cluster definitions that is visible in the queue manager at the time.
- When a publication is sent to multiple subscribers, the attributes used from the topic object are used consistently for all subscribers that receive the publication. For example, inhibiting publication on a topic is applied for the next application MQPUT to the topic. A publication that is in progress to multiple subscribers completes to all subscribers. This publication does not take note of a change that has happened, part of the way through, to any attribute on the topic.
- Successful completion of the command does not mean that the action completed. To check for true completion, see the [DEFINE TOPIC](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for DEFINE TOPIC

### *(topic-name)*

Name of the IBM MQ topic definition (see [Rules for naming IBM MQ objects](#)). The maximum length is 48 characters.

The name must not be the same as any other topic definition currently defined on this queue manager (unless REPLACE is specified).

V 9.3.1

Multi

### **CAEXPRY***(integer)*

The maximum time, expressed in tenths of a second, until a message published to a topic which inherits properties from this object, remains in the system until it becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

### *integer*

The value must be in the range one through to 999 999 999.

### **NOLIMIT**

There is no limit on the expiry time of messages put to this topic.

### **ASPARENT**

The maximum message expiry time is based on the setting of the closest parent administrative topic object in the topic tree. This is the default value.

### **CLROUTE**

The routing behavior to use for topics in the cluster defined by the **CLUSTER** parameter.

### **DIRECT**

When you configure a direct routed clustered topic on a queue manager, all queue managers in the cluster become aware of all other queue managers in the cluster. When performing publish and subscribe operations, each queue manager can connect direct to any other queue manager in the cluster.

### **TOPICHOST**

When you use topic host routing, all queue managers in the cluster become aware of the cluster queue managers that host the routed topic definition (that is, the queue managers on which

you have defined the topic object). When performing publish and subscribe operations, queue managers in the cluster connect only to these topic host queue managers, and not directly to each other. The topic host queue managers are responsible for routing publications from queue managers on which publications are published to queue managers with matching subscriptions.

After a topic object has been clustered (through setting the **CLUSTER** property) you cannot change the value of the **CLROUTE** property. The object must be un-clustered (**CLUSTER** set to ' ') before you can change the value. Un-clustering a topic converts the topic definition to a local topic, which results in a period during which publications are not delivered to subscriptions on remote queue managers; this should be considered when performing this change. See [The effect of defining a non-cluster topic with the same name as a cluster topic from another queue manager](#). If you try to change the value of the **CLROUTE** property while it is clustered, the system generates an MQRCCF\_CLROUTE\_NOT\_ALTERABLE exception.

See also [Routing for publish/subscribe clusters: Notes on behavior](#) and [Designing publish/subscribe clusters](#).

## CLUSTER

The name of the cluster to which this topic belongs. Setting this parameter to a cluster that this queue manager is a member of makes all queue managers in the cluster aware of this topic. Any publication to this topic or a topic string below it put to any queue manager in the cluster is propagated to subscriptions on any other queue manager in the cluster. For more details, see [Distributed publish/subscribe networks](#).

..

If no topic object above this topic in the topic tree has set this parameter to a cluster name, then this topic does not belong to a cluster. Publications and subscriptions for this topic are not propagated to publish/subscribe cluster-connected queue managers. If a topic node higher in the topic tree has a cluster name set, publications and subscriptions to this topic are also propagated throughout the cluster.

### *string*

The topic belongs to this cluster. It is not recommended that this is set to a different cluster from a topic object above this topic object in the topic tree. Other queue managers in the cluster will honor this object's definition unless a local definition of the same name exists on those queue managers.

To prevent all subscriptions and publications being propagated throughout a cluster, leave this parameter blank on the system topics SYSTEM.BASE.TOPIC and SYSTEM.DEFAULT.TOPIC, except in special circumstances, for example to support migration.

## CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

## COMMINFO( *comminfo-name* )

The name of the Multicast communication information object associated with this topic object.

**CUSTOM(string)**

The custom attribute for new features.

This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE).

**CAEXPRY(integer)**

**Note:** **V 9.3.1** The CAEXPRY queue attribute introduced at IBM MQ 9.3.1 replaces the use of the CAEXPRY option in the CUSTOM field. It is not possible to set the CAEXPRY attribute if the CUSTOM field has a CAEXPRY attribute defined in it already. You should alter existing topics to set the new CAEXPRY field and unset the CAEXPRY attribute from the CUSTOM field. For example:

```
DEFINE TOPIC(Q1) CAEXPRY(1000) CAEXPRY('')
```

The maximum time, expressed in tenths of a second, until a message published to a topic which inherits properties from this object, remains in the system until it becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

**integer**

The value must be in the range one through to 999 999 999.

**NOLIMIT**

There is no limit on the expiry time of messages put to this topic.

**ASPARENT**

The maximum message expiry time is based on the setting of the closest parent administrative topic object in the topic tree. This is the default value.

Specifying a value for CAEXPRY that is not valid, does not cause the command to fail. Instead, the default value is used.

**DEFPRESP**

Specifies the put response to be used when applications specify the MQPMO\_RESPONSE\_AS\_DEF option.

**ASPARENT**

The default put response is based on the setting of the closest parent administrative topic object in the topic tree.

**SYNC**

Put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application.

**ASYNC**

Put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application; but an improvement in performance might be seen for messages put in a transaction and any non-persistent messages

**DEFPRTY( integer )**

The default priority of messages published to the topic.

**( integer )**

The value must be in the range zero (the lowest priority), through to the MAXPRTY queue manager parameter (MAXPRTY is 9).

**ASPARENT**

The default priority is based on the setting of the closest parent administrative topic object in the topic tree.

**DEFPSIST**

Specifies the message persistence to be used when applications specify the MQPER\_PERSISTENCE\_AS\_TOPIC\_DEF option.

**ASPARENT**

The default persistence is based on the setting of the closest parent administrative topic object in the topic tree.

**NO**

Messages on this queue are lost during a restart of the queue manager.

**YES**

Messages on this queue survive a restart of the queue manager.

On z/OS, N and Y are accepted as synonyms of NO and YES.

**DESCR( *string* )**

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY TOPIC command.

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

**DURSUB**

Specifies whether applications are permitted to make durable subscriptions on this topic.

**ASPARENT**

Whether durable subscriptions can be made on this topic is based on the setting of the closest parent administrative topic object in the topic tree.

**NO**

Durable subscriptions cannot be made on this topic.

**YES**

Durable subscriptions can be made on this topic.

**LIKE( *topic-name* )**


The name of a topic. The topic parameters are used to model this definition.

If this field is not completed, and you do not complete the parameter fields related to the command, the values are taken from the default definition for topics on this queue manager.

Not completing this field is equivalent to specifying:

```
LIKE(SYSTEM.DEFAULT.TOPIC)
```

A default topic definition is provided, but it can be altered by the installation to the default values required. See [Rules for naming IBM MQ objects](#).

 On z/OS, the queue manager searches page set zero for an object with the name you specify and a disposition of QMGR or COPY. The disposition of the LIKE object is not copied to the object you are defining.

**Note:**

1. QSGDISP (GROUP) objects are not searched.
2. LIKE is ignored if QSGDISP(COPY) is specified.

**MCAST**

Specifies whether multicast is allowable in the topic tree. The values are:

**ASPARENT**

The multicast attribute of the topic is inherited from the parent.



**DISABLED**

No multicast traffic is allowed at this node.

**ENABLED**

Multicast traffic is allowed at this node.

**ONLY**

Only subscriptions from a multicast capable client are allowed.

**MDURMDL(string)**

The name of the model queue to be used for durable subscriptions that request that the queue manager manages the destination of its publications (see [Rules for naming IBM MQ objects](#)). The maximum length is 48 characters.

If **MDURMDL** is blank, it operates in the same way as **ASPARENT** values on other attributes. The name of the model queue to be used is based on the closest parent administrative topic object in the topic tree with a value set for **MDURMDL**.

If you use **MDURMDL** to specify a model queue for a clustered topic, you must ensure that the queue is defined on every queue manager in the cluster where a durable subscription using this topic can be made.

The dynamic queue created from this model has a prefix of SYSTEM.MANAGED.DURABLE

**MNDURMDL( string )**

The name of the model queue to be used for non-durable subscriptions that request that the queue manager manages the destination of its publications (see [Rules for naming IBM MQ objects](#)). The maximum length is 48 characters.

If **MNDURMDL** is blank, it operates in the same way as **ASPARENT** values on other attributes. The name of the model queue to be used is based on the closest parent administrative topic object in the topic tree with a value set for **MNDURMDL**.

If you use **MNDURMDL** to specify a model queue for a clustered topic, you must ensure that the queue is defined on every queue manager in the cluster where a non-durable subscription using this topic can be made.

The dynamic queue created from this model has a prefix of SYSTEM.MANAGED.NDURABLE.

**NPMGDLV**

The delivery mechanism for non-persistent messages published to this topic:

**ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**ALL**

Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**ALLAVAIL**

Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**ALLDUR**

Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a non-persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT calls fails.

**PMSGDLV**

The delivery mechanism for persistent messages published to this topic:

**ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**ALL**

Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**ALLAVAIL**

Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**ALLDUR**

Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT calls fails.

**PROXYSUB**

Controls when a proxy subscription is sent for this topic, or topic strings below this topic, to neighboring queue managers when in a publish/subscribe cluster or hierarchy. For more details, see [Subscription performance in publish/subscribe networks](#).

**FIRSTUSE**

For each unique topic string at or below this topic object, a proxy subscription is asynchronously sent to all neighboring queue managers in the following scenarios:

- When a local subscription is created.
- When a proxy subscription is received that must be propagated to further directly connected queue managers.

**FORCE**

A wildcard proxy subscription that matches all topic strings at and below this point in the topic tree is sent to neighboring queue managers even if no local subscriptions exist.

**Note:** The proxy subscription is sent when this value is set on DEFINE or ALTER. When set on a clustered topic, all queue managers in the cluster issue the wildcard proxy subscription to all other queue managers in the cluster.

**PUB**

Controls whether messages can be published to this topic.

**ASPARENT**

Whether messages can be published to the topic is based on the setting of the closest parent administrative topic object in the topic tree.

**ENABLED**

Messages can be published to the topic (by suitably authorized applications).

**DISABLED**

Messages cannot be published to the topic.

See also [Special handling for the PUB parameter](#).

**PUBSCOPE**

Determines whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster.

**Note:** You can restrict the behavior on a publication-by-publication basis, using MQPMO\_SCOPE\_QMGR on the Put Message options.

**ASPARENT**

Determines whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster. This is based on the setting of the first parent administrative node found in the topic tree that relates to this topic.

**QMGR**

Publications for this topic are not propagated to connected queue managers.

**ALL**

Publications for this topic are propagated to hierarchically connected queue managers and to publish/subscribe cluster-connected queue managers.

**z/OS**

**QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object within the group.

<i>Table 166. Object dispositions for QSGDISP options</i>	
<b>QSGDISP</b>	<b>DEFINE</b>
COPY	<p>The object is defined on the page set of the queue manager that executes the command. It uses the <b>QSGDISP (GROUP)</b> object of the same name as the LIKE object.</p> <p>For example, if you issue the following command,</p> <pre>DEFINE TOPIC(topic_name) REPLACE QSGDISP(COPY)</pre> <p>the queue manager searches the shared configuration repository for a TOPIC definition called <i>topic_name</i>. If a matching TOPIC definition is found, the queue manager creates a local copy of this definition on the queue manager page set.</p> <p>For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.</p>
GROUP	<p>The object definition resides in the shared configuration repository. <b>QSGDISP (GROUP)</b> is allowed only if the queue manager is in a queue sharing group.</p> <p>If the DEFINE for the <b>QSGDISP (GROUP)</b> object is successful, the DEFINE TOPIC(<i>topic_name</i>) REPLACE QSGDISP(COPY) command is generated and sent to all active queue managers in the queue sharing group to make or refresh local copies on page set zero.</p> <p>The DEFINE for the group object takes effect regardless of whether the generated command with <b>QSGDISP (COPY)</b> fails.</p>
PRIVATE	Not permitted.
QMGR	The object is defined on the page set of the queue manager that executes the command.

**REPLACE and NOREPLACE**

Determines whether the existing definition (and on z/OS, with the same disposition) is to be replaced with this one. Any object with a different disposition is not changed.

**REPLACE**

If the object does exist, the effect is like issuing the **ALTER** command without the **FORCE** option and with *all* the other parameters specified.

(The difference between the **ALTER** command without the **FORCE** option, and the **DEFINE** command with the **REPLACE** option, is that **ALTER** does not change unspecified parameters, but **DEFINE** with **REPLACE** sets *all* the parameters. When you use **REPLACE**, unspecified parameters are taken either from the object named on the **LIKE** option, or from the default definition, and the parameters of the object being replaced, if one exists, are ignored.)

The command fails if both of the following statements are true:

- The command sets parameters that would require the use of the **FORCE** option if you were using the **ALTER** command.
- The object is open.

The ALTER command with the FORCE option succeeds in this situation.

**Note:** The REPLACE option does not replace the TOPICSTR properties of a topic. TOPICSTR is a property that is usefully varied in the example to test different topic trees. To change topics, delete the topic first.

#### **NOREPLACE**

The definition must not replace any existing definition of the object.

#### **SUB**

Controls whether applications are to be permitted to subscribe to this topic.

#### **ASPARENT**

Whether applications can subscribe to the topic is based on the setting of the closest parent administrative topic object in the topic tree.

#### **ENABLED**

Subscriptions can be made to the topic (by suitably authorized applications).

#### **DISABLED**

Applications cannot subscribe to the topic.

#### **SUBSCOPE**

Determines whether this queue manager subscribes to publications in this queue manager or in the network of connected queue managers. If subscribing to all queue managers, the queue manager propagates subscriptions to them as part of a hierarchy or as part of a publish/subscribe cluster.

**Note:** You can restrict the behavior on a subscription-by-subscription basis, using **MQPMO\_SCOPE\_QMGR** on the Subscription Descriptor or **SUBSCOPE(QMGR)** on **DEFINE SUB**. Individual subscribers can override the **SUBSCOPE** setting of ALL by specifying the **MQSO\_SCOPE\_QMGR** subscription option when creating a subscription.

#### **ASPARENT**

Whether this queue manager subscribes to publications in the same way as the setting of the first parent administrative node found in the topic tree relating to this topic.

#### **QMGR**

Only publications that are published on this queue manager reach the subscriber.

#### **ALL**

A publication made on this queue manager or on another queue manager reaches the subscriber. Subscriptions for this topic are propagated to hierarchically connected queue managers and to publish/subscribe cluster-connected queue managers.

#### **TOPICSTR(string)**


The topic string represented by this topic object definition. This parameter is required and cannot contain the empty string.

The topic string must not be the same as any other topic string already represented by a topic object definition.

The maximum length of the string is 10,240 characters.

**Note:** The REPLACE option does not replace the TOPICSTR properties of a topic. TOPICSTR is a property that is usefully varied in the example to test different topic trees. To change topics, delete the topic first.

#### **TYPE(topic-type)**

If this parameter is used it must follow immediately after the *topic-name* parameter on all platforms  except z/OS.

#### **LOCAL**

A local topic object.

## USEDLQ

Determines whether the dead-letter queue is used when publication messages cannot be delivered to their correct subscriber queue.

## ASPARENT

Determines whether to use the dead-letter queue using the setting of the closest administrative topic object in the topic tree. This value is the default supplied with IBM MQ, but your installation might have changed it.

## NO

Publication messages that cannot be delivered to their correct subscriber queue are treated as a failure to put the message. The MQPUT of an application to a topic fails in accordance with the settings of **NPMGDLV** and **PMSGDLV**.

## YES

When the **DEADQ** queue manager attribute provides the name of a dead-letter queue, then it is used. If the queue manager does not provide the name of a dead-letter queue, then the behavior is as for NO.

## WILDCARD

The behavior of wildcard subscriptions with respect to this topic.

## PASSTHRU

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object receive publications made to this topic and to topic strings more specific than this topic.

## BLOCK

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object do not receive publications made to this topic or to topic strings more specific than this topic.

The value of this attribute is used when subscriptions are defined. If you alter this attribute, the set of topics covered by existing subscriptions is not affected by the modification. This scenario applies also if the topology is changed when topic objects are created or deleted; the set of topics matching subscriptions created following the modification of the **WILDCARD** attribute is created using the modified topology. If you want to force the matching set of topics to be re-evaluated for existing subscriptions, you must restart the queue manager.

## Related tasks


[Defining an administrative topic](#)

## DELETE AUTHINFO (delete authentication information)

Use MQSC command **DELETE AUTHINFO** to delete an authentication information object.

## Using MQSC commands

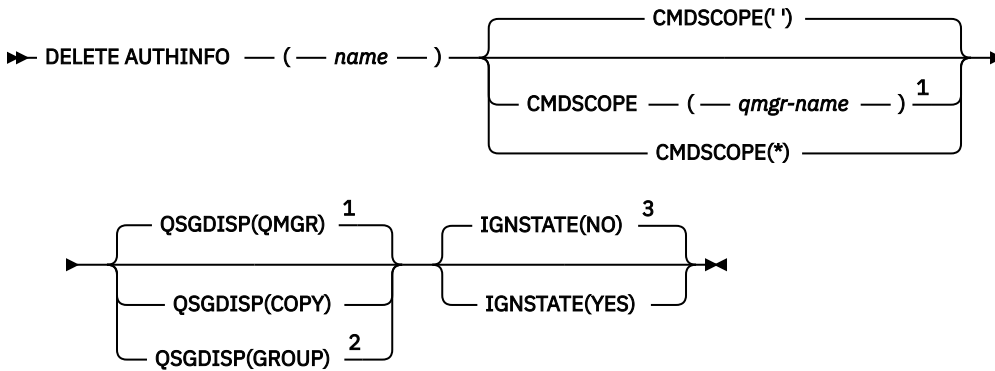
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DELETE AUTHINFO” on page 606](#)

**Synonym:** None

## DELETE AUTHINFO



### Notes:

- <sup>1</sup> Valid only on z/OS.
- <sup>2</sup> Valid only when the queue manager is a member of a queue sharing group. You can use queue sharing groups only on IBM MQ for z/OS.
- <sup>3</sup> Valid only on AIX, Linux, and Windows.

## Parameter descriptions for DELETE AUTHINFO

### (name)

Name of the authentication information object. This is required.

The name must be that of an existing authentication information object.

### z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered. This is the default value.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### z/OS QSGDISP

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

### COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters **QSGDISP (COPY)**. Any object residing in the shared repository, or any object defined using a command that had the parameters **QSGDISP(QMGR)**, is not affected by this command.

## GROUP

The object definition resides in the shared repository. The object was defined using a command that had the parameters **QSGDISP (GROUP)**. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE AUTHINFO(name) QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with **QSGDISP (COPY)** fails.

## QMGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters **QSGDISP (QMGR)**. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

## V 9.3.0 ALW IGNSTATE

Specifies whether the command fails if the object does not exist. The value can be one of the following values:

### NO

When the command is issued from **runmqsc** started with the **-n** parameter to run without connecting to a queue manager, the command succeeds regardless of whether the object exists.

In all other environments, the command fails if the object does not exist.

This is the default value.

### YES

The command succeeds regardless of whether the object exists.

## Multi DELETED AUTHREC (delete authority records) on Multiplatforms

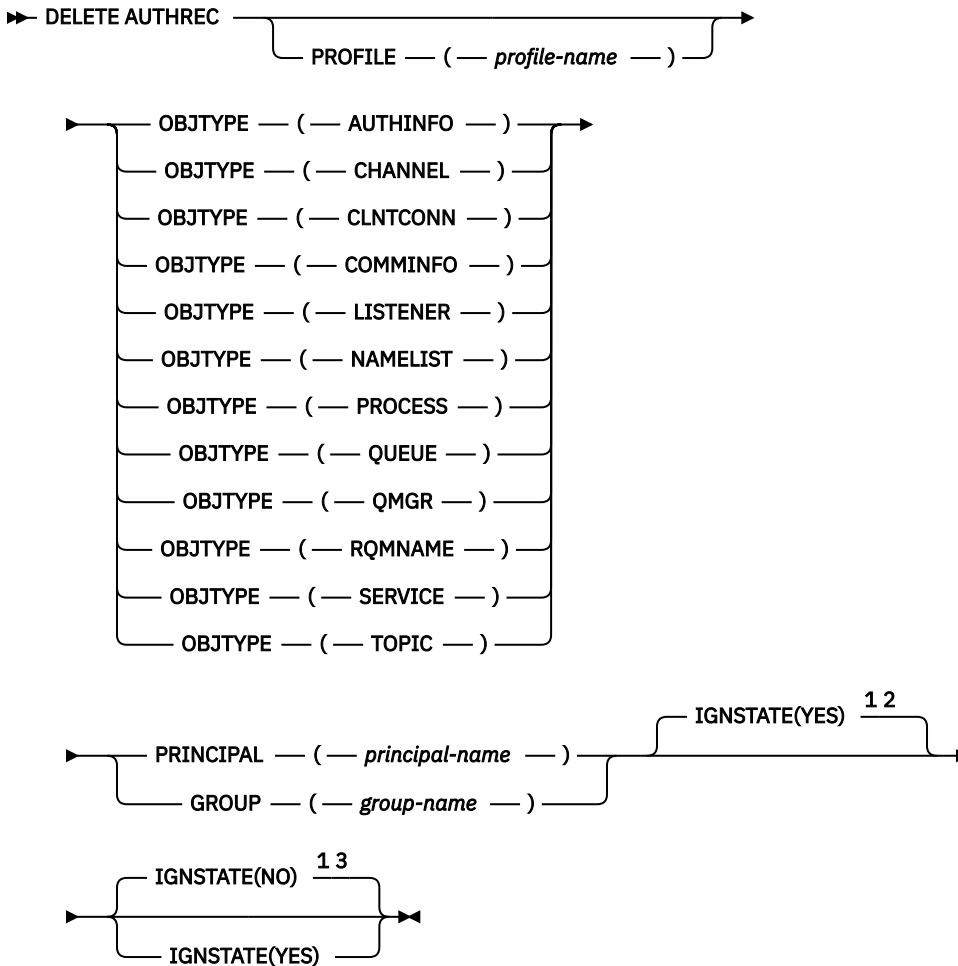
Use the MQSC command **DELETE AUTHREC** to delete authority records associated with a profile name.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions” on page 608](#)

## DELETE AUTHREC



### Notes:

- <sup>1</sup> Valid only on AIX, Linux, and Windows.
- <sup>2</sup> Default value is YES for QMGR, TOPIC, RQMNAME and QUEUE objects when set using the OBJTYPE parameter. Setting IGNSSTATE to NO is not valid for these objects.
- <sup>3</sup> Default value is NO for objects apart from QMGR, TOPIC, RQMNAME and QUEUE.

## Parameter descriptions

### PROFILE(*profile-name*)

The name of the object or generic profile for which to remove the authority record. This parameter is required unless the **OBJTYPE** parameter is QMGR, in which case it can be omitted.

### OBJTYPE

The type of object referred to by the profile. Specify one of the following values:

#### AUTHINFO

Authentication information record

#### CHANNEL

Channel

#### CLNTCONN

Client connection channel

#### COMMINFO

Communication information object



**LISTENER**

Listener

**NAMELIST**

Namelist

**PROCESS**

Process

**QUEUE**

Queue

**QMGR**

Queue manager

**RQMNAME**

Remote queue manager

**SERVICE**

Service

**TOPIC**

Topic

**PRINCIPAL(*principal-name*)**

A principal name. This is the name of a user for whom to remove authority records for the specified profile. On IBM MQ for Windows, the name of the principal can optionally include a domain name, specified in this format: user@domain.

You must specify either PRINCIPAL or GROUP.

**GROUP(*group-name*)**

A group name. This is the name of the user group for which to remove authority records for the specified profile. You can specify one name only and it must be the name of an existing user group.

**Windows** For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following formats:

```
GroupName@domain
domain\GroupName
```

You must specify either PRINCIPAL or GROUP.

**V9.3.0** **ALW** **IGNSTATE**

Specifies whether the command fails if the authority record does not exist. The value can be one of the following values:

**NO**

The command fails if the authority record does not exist.

This value is not valid for objects of type QUEUE, QMGR, RQMNAME, and TOPIC. This is the default value for all other object types.

**YES**

The command succeeds regardless of whether the authority record exists.

This is the default value for objects of type QUEUE, QMGR, RQMNAME, and TOPIC.

**z/OS** **DELETE BUFFPOOL (delete a buffer pool) on z/OS**

Use the MQSC command DELETE BUFFPOOL to delete a buffer pool that is used for holding messages in main storage.

**Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage note for DELETE BUFFPOOL” on page 610](#)
- [“Parameter descriptions for DELETE BUFFPOOL” on page 610](#)

**Synonym:** DEL BP

### DELETE BUFFPOOL

► DELETE BUFFPOOL — ( — *integer* — ) ◄

### Usage note for DELETE BUFFPOOL

- Ensure there are no current page set definitions using the named buffer pool, otherwise the command will fail.
- DELETE BUFFPOOL cannot be issued from CSQINPT.

### Parameter descriptions for DELETE BUFFPOOL

(*integer*)

This is the number of the buffer pool to be deleted. The value is an integer in the range zero through 99.

### z/OS DELETE CFSTRUCT (delete CF application structure) on z/OS

Use the MQSC command DELETE CFSTRUCT to delete a CF application structure definition.

### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DELETE CFSTRUCT” on page 610](#)
- [“Keyword and parameter descriptions for DELETE CFSTRUCT” on page 611](#)

**Synonym:** None

### DELETE CFSTRUCT

► DELETE CFSTRUCT — ( — *structure-name* — ) ◄

### Usage notes for DELETE CFSTRUCT

1. This command is valid only z/OS when the queue manager is a member of a queue sharing group.
2. The command fails if there are any queues in existence that reference this CF structure name that are not both empty and closed.
3. The command cannot specify the CF administration structure (CSQ\_ADMIN).
4. The command deletes the Db2 CF structure record only. It does **not** delete the CF structure definition from the CFRM policy data set.

- 5. CF structures at CFLEVEL(1) are automatically deleted when the last queue on that structure is deleted.

## Keyword and parameter descriptions for DELETE CFSTRUCT

**(structure-name)**

The name of the CF structure definition to be deleted. The name must be defined within the queue sharing group.

## DELETE CHANNEL (delete a channel)

Use the MQSC command DELETE CHANNEL to delete a channel definition.

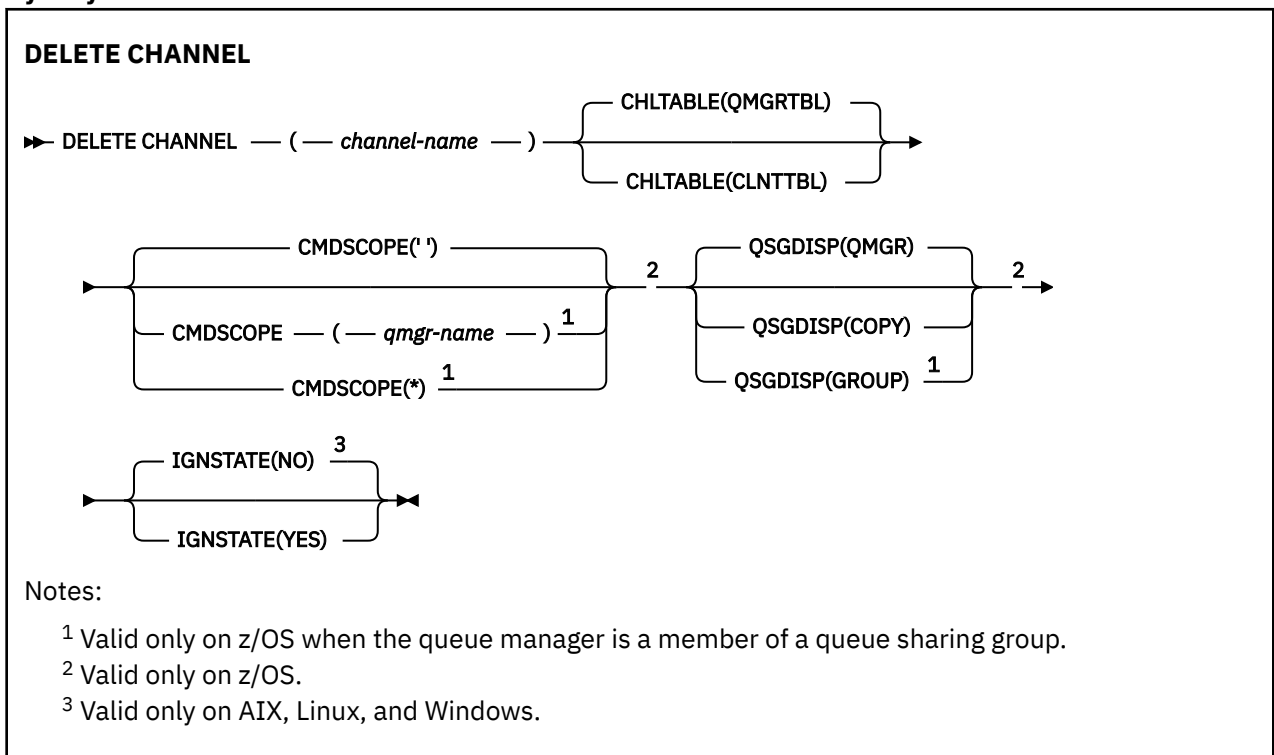
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 611](#)
- [“Parameter descriptions” on page 612](#)

**Synonym:** DELETE CHL



### Usage notes

- Successful completion of the command does not mean that the action completed. To check for true completion, see the [DELETE CHANNEL](#) step in [Checking that async commands for distributed networks have finished](#).

- ▶ **z/OS** On z/OS systems, the command fails if the channel initiator and command server have not been started, or the channel status is RUNNING, except client-connection channels, which can be deleted without the channel initiator or command server running.
- ▶ **z/OS** On z/OS systems, you can only delete cluster-sender channels that have been created manually.

## Parameter descriptions

### **(channel-name)**

The name of the channel definition to be deleted. This is required. The name must be that of an existing channel.

### **CHLTABLE**

Specifies the channel definition table that contains the channel to be deleted. This is optional.

### **QMGRTBL**

The channel table is that associated with the target queue manager. This table does not contain any channels of type CLNTCONN. This is the default.

### **CLNTTBL**

The channel table for CLNTCONN channels. On z/OS, this is associated with the target queue manager, but separate from the main channel table. On all other platforms, this channel table is normally associated with a queue manager, but can be a system-wide, queue manager independent channel table if you set up a number of environment variables. For more information about setting up environment variables, see [Using IBM MQ environment variables](#).

## ▶ **z/OS** **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

''

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## ▶ **z/OS** **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

### **COPY**

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(COPY). Any object residing in the shared repository, or any object defined using a command that had the parameters QSGDISP(QMGR), is not affected by this command.

## GROUP

The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(GROUP). Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE CHANNEL(channel-name) QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

## QMGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

## V9.3.0 ALW IGNSTATE

Specifies whether the command fails if the channel does not exist. The value can be one of the following values:

### NO

When the command is issued from **runmqsc** started with the **-n** parameter to run without connecting to a queue manager, the command succeeds regardless of whether the channel exists.

In all other environments, the command fails if the channel does not exist.

This is the default value.

### YES

The command succeeds regardless of whether the channel exists.

## ALW DELETE CHANNEL (delete a channel) MQTT

Use the MQSC command DELETE CHANNEL to delete an MQ Telemetry channel definition.

## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

The DELETE CHANNEL (MQTT) command is only valid for MQ Telemetry channels.

**Synonym:** DELETE CHL

### DELETE CHANNEL

```
▶▶ DELETE CHANNEL — ( — channel-name — ) — CHLTYPE — ( — MQTT — ) ▶▶
```

## Parameter descriptions

### (*channel-name*)

The name of the channel definition to be deleted. This is required. The name must be that of an existing channel.

### CHLTYPE

This parameter is required. There is only one possible value: MQTT.

## Multi **DELETE COMMINFO (delete communications information) on Multiplatforms**

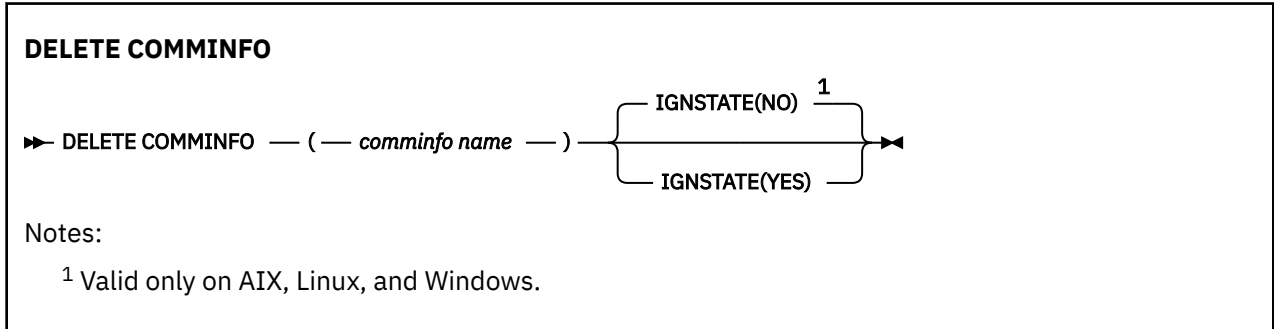
Use the MQSC command DELETE COMMINFO to delete a communication information object.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DELETE COMMINFO” on page 614](#)

**Synonym:** DEL COMMINFO



### Parameter descriptions for DELETE COMMINFO

#### (*comminfo name*)

The name of the communications information object to be deleted. This parameter is required.

#### V 9.3.0 ALW **IGNSTATE**

Specifies whether the command fails if the object does not exist. The value can be one of the following values:

#### **NO**

The command fails if the object does not exist. This is the default value.

#### **YES**

The command succeeds regardless of whether the object exists.

## Multi **DELETE LISTENER (delete a listener) on Multiplatforms**

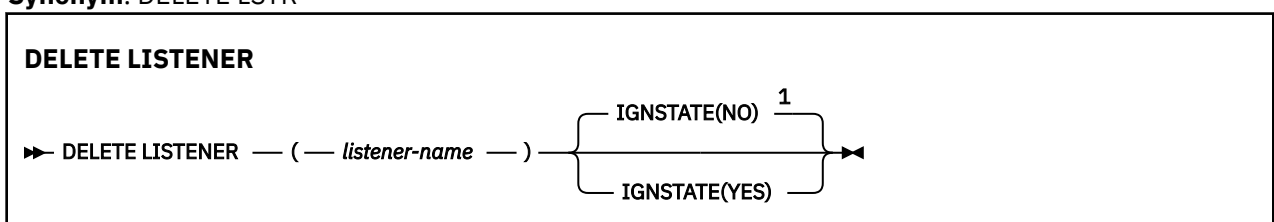
Use the MQSC command DELETE LISTENER to delete a listener definition.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Usage notes for DELETE LISTENER” on page 615](#)
- [“Keyword and parameter descriptions for DELETE LISTENER” on page 615](#)

**Synonym:** DELETE LSTR



Notes:

<sup>1</sup> Valid only on AIX, Linux, and Windows.

## Usage notes for DELETE LISTENER

1. The command fails if an application has the specified listener object open, or if the listener is currently running.

## Keyword and parameter descriptions for DELETE LISTENER

### *(listener-name)*

The name of the listener definition to be deleted. This parameter is required. The name must be that of an existing listener defined on the local queue manager.

### V9.3.0 ALW IGNSTATE

Specifies whether the command fails if the listener does not exist. The value can be one of the following values:

#### **NO**

The command fails if the listener does not exist. This is the default value.

#### **YES**

The command succeeds regardless of whether the listener exists.

## DELETE NAMELIST (delete a name list)

Use the MQSC command DELETE NAMELIST to delete a namelist definition.

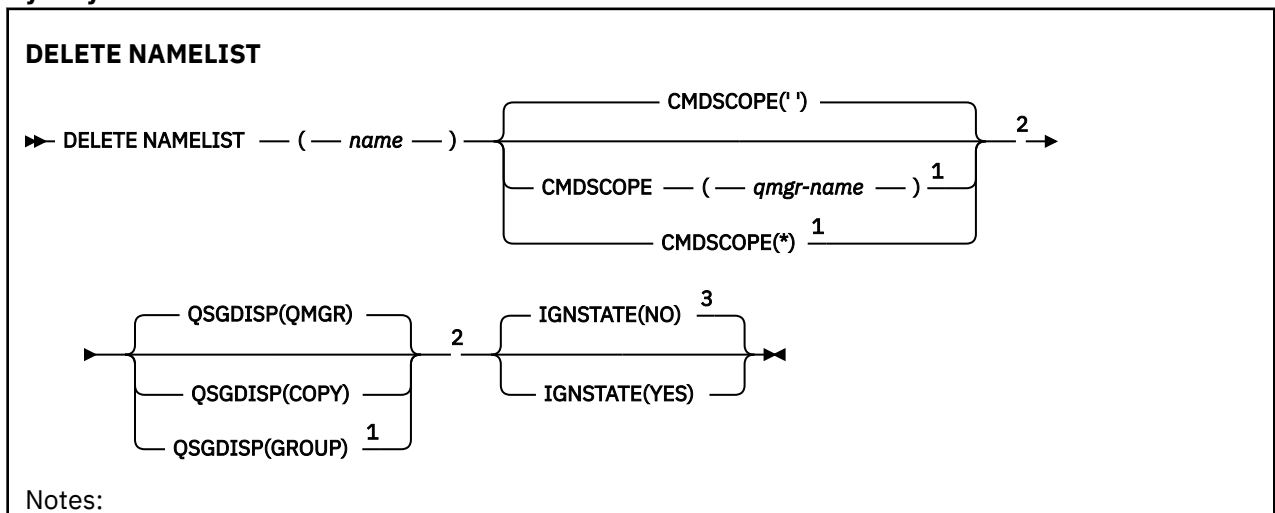
## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 616](#)
- [“Parameter descriptions for DELETE NAMELIST” on page 616](#)

**Synonym:** DELETE NL



<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

<sup>2</sup> Valid only on z/OS.

<sup>3</sup> Valid only on AIX, Linux, and Windows.

## Usage notes

Successful completion of the command does not mean that the action completed. To check for true completion, see the [DELETE NAMELIST](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for DELETE NAMELIST

You must specify which namelist definition you want to delete.

### (name)

The name of the namelist definition to be deleted. The name must be defined to the local queue manager.

If an application has this namelist open, the command fails.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

### **COPY**

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(COPY). Any object residing in the shared repository, or any object defined using a command that had the parameters QSGDISP(QMGR), is not affected by this command.

### **GROUP**

The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(GROUP). Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:



```
DELETE NAMELIST(name) QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

### QMGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

### V 9.3.0 ALW IGNSTATE

Specifies whether the command fails if the namelist does not exist. The value can be one of the following values:

#### NO

The command fails if the namelist does not exist. This is the default value.

#### YES

The command succeeds regardless of whether the namelist exists.

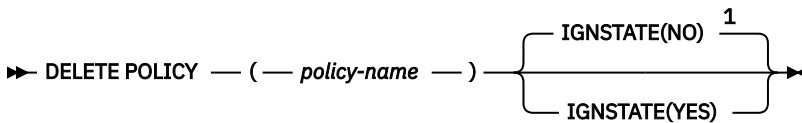
### Multi

## DELETE POLICY (delete a security policy) on Multiplatforms

Use the MQSC command DELETE POLICY to delete a security policy.

- [Syntax diagram](#)
- [“Parameter descriptions for DELETE POLICY” on page 617](#)

### DELETE POLICY



Notes:

<sup>1</sup> Valid only on AIX, Linux, and Windows.

## Parameter descriptions for DELETE POLICY

### (*policy-name*)

Specifies the policy name to be deleted.

The name of the policy, or policies, to delete are the same as the name of the queue, or queues, that the policies control.

### V 9.3.0 ALW IGNSTATE

Specifies whether the command fails if the policy does not exist. The value can be one of the following values:

#### NO

The command fails if the policy does not exist. This is the default value.

#### YES

The command succeeds regardless of whether the policy exists.

## DELETE PROCESS (delete a process definition)

Use the MQSC command DELETE PROCESS to delete a process definition.

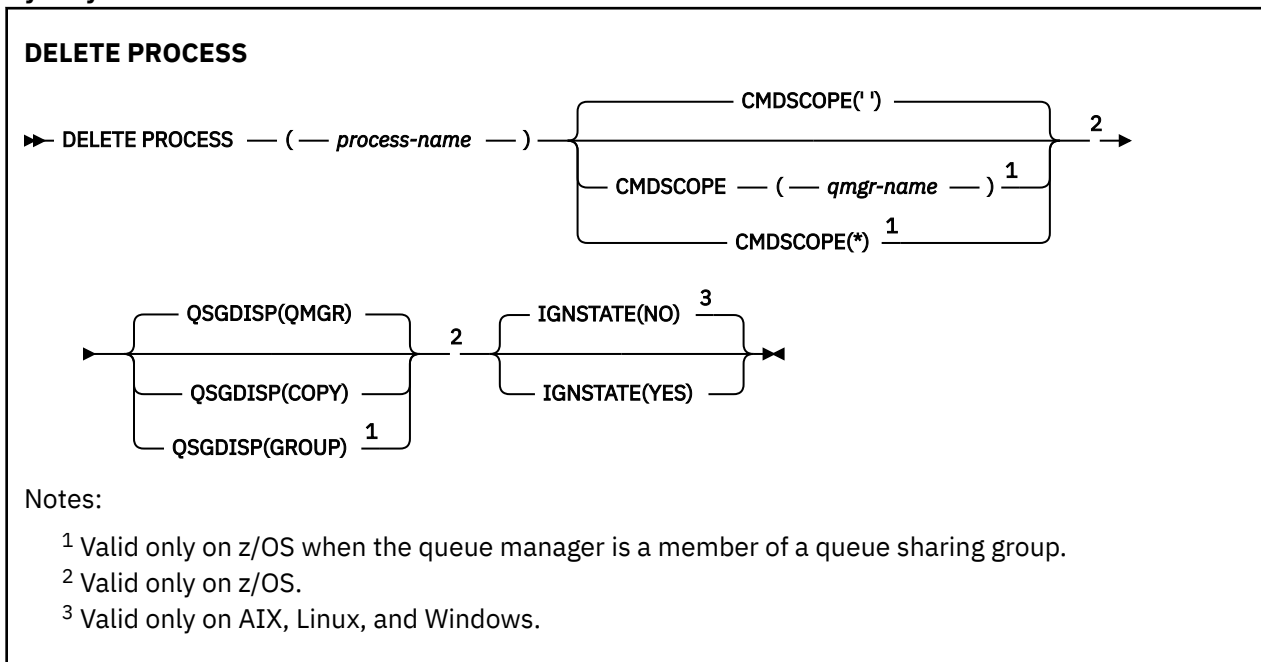
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DELETE PROCESS” on page 618](#)

**Synonym:** DELETE PRO



### Parameter descriptions for DELETE PROCESS

You must specify which process definition you want to delete.

#### **(process-name)**

The name of the process definition to be deleted. The name must be defined to the local queue manager.

If an application has this process open, the command fails.

#### **z/OS** **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

### **COPY**

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(COPY). Any object residing in the shared repository, or any object defined using a command that had the parameters QSGDISP(QMGR), is not affected by this command.

### **GROUP**

The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(GROUP). Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE PROCESS(process-name) QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

### **QMGR**

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

## **IGNSTATE**

Specifies whether the command fails if the process definition does not exist. The value can be one of the following values:

### **NO**

The command fails if the process definition does not exist. This is the default value.

### **YES**

The command succeeds regardless of whether the process definition exists.

## **DELETE PSID (delete a page set) on z/OS**

Use the MQSC command DELETE PSID to delete a page set. This command closes the page set and de-allocates it from the queue manager.

### **Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DELETE PSID” on page 620](#)
- [“Parameter descriptions for DELETE PSID” on page 620](#)

**Synonym:** DEL PSID

#### **DELETE PSID**

► DELETE PSID — ( — *psid-number* — ) ◄

### **Usage notes for DELETE PSID**

1. The identified page set must have no storage class (STGCLASS) referencing it.
2. If the page set still has buffers in the buffer pool when you issue this command, the command fails and an error message is issued. You cannot delete the page set until 3 checkpoints have been completed since the page set was emptied.
3. If the page set is not to be used again by the queue manager, update the queue manager started task procedure JCL, and remove the corresponding DEFINE PSID command from the CSQINP1 initialization data set. If the page set had a dedicated buffer pool, remove its definitions also from CSQINP1.
4. If you want to reuse the data set again as a page set, format it before doing so.

### **Parameter descriptions for DELETE PSID**

**(*psid-number*)**

Identifier of the page set. This is required. You cannot delete page set 0.

### **DELETE queues**


Use the MQSC **DELETE** command to delete a queue definition for a local, model, or remote queue, or a queue alias.

### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

This section contains the following commands:

- [“DELETE QALIAS \(delete an alias queue definition\)” on page 623](#)
- [“DELETE QLOCAL \(delete a local queue definition\)” on page 623](#)
- [“DELETE QMODEL \(delete a model queue definition\)” on page 624](#)
- [“DELETE QREMOTE \(delete local definition of remote queue\)” on page 625](#)

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

### **Usage notes for DELETE queues**

- Successful completion of the command does not mean that the action completed. To check for true completion, see the [DELETE queues](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for DELETE queues

### *(q-name)*

The name of the queue must be defined to the local queue manager for all the queue types.

For an alias queue this is the local name of the alias queue to be deleted.

For a model queue this is the local name of the model queue to be deleted.

For a remote queue this is the local name of the remote queue to be deleted.

For a local queue this is the name of the local queue to be deleted. You must specify which queue you want to delete.

**Note:** A queue cannot be deleted if it contains uncommitted messages.

If an application has this queue open, or has open a queue that eventually resolves to this queue, the command fails. The command also fails if this queue is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open.

If this queue has a SCOPE attribute of CELL, the entry for the queue is also deleted from the cell directory.

### **AUTHREC**

This parameter does not apply to z/OS.

Specifies whether the associated authority record is also deleted:

#### **YES**

The authority record associated with the object is deleted. This is the default.

#### **NO**

The authority record associated with the object is not deleted.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP or SHARED.

''

The command runs on the queue manager on which it was entered. This is the default value.

#### ***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### **PURGE and NOPURGE**

Specifies whether any existing committed messages on the queue named by the DELETE command are to be purged for the delete command to work. The default is NOPURGE.

#### **PURGE**

The deletion is to go ahead even if there are committed messages on the named queue, and these messages are also to be purged.

#### **NOPURGE**

The deletion is not to go ahead if there are any committed messages on the named queue.

**QSGDISP**

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). If the object definition is shared, you do not need to delete it on every queue manager that is part of a queue sharing group. (Queue sharing groups are available only on IBM MQ for z/OS.)

**COPY**

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(COPY). Any object residing in the shared repository, or any object defined using a command that had the parameters QSGDISP(QMGR), is not affected by this command.

**GROUP**

The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(GROUP). Any object residing on the page set of the queue manager that executes the command, or any object defined using a command that had the parameters QSGDISP(SHARED), is not affected by this command.

If the deletion is successful, the following command is generated and sent to all active queue managers in the queue sharing group to make, or delete, local copies on page set zero:

```
DELETE queue(q-name) QSGDISP(COPY)
```

or, for a local queue only:

```
DELETE QLOCAL(q-name) NOPURGE QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

**Note:** You always get the NOPURGE option even if you specify PURGE. To delete messages on local copies of the queues, you must explicitly issue the command:

```
DELETE QLOCAL(q-name) QSGDISP(COPY) PURGE
```

for each copy.

**QMGR**

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

**SHARED**

This option applies only to local queues.

The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(SHARED). Any object residing on the page set of the queue manager that executes the command, or any object defined using a command that had the parameters QSGDISP(GROUP), is not affected by this command.

**IGNSTATE**

Specifies whether the command fails if the queue does not exist. The value can be one of the following values:

**NO**

The command fails if the queue does not exist. This is the default value.

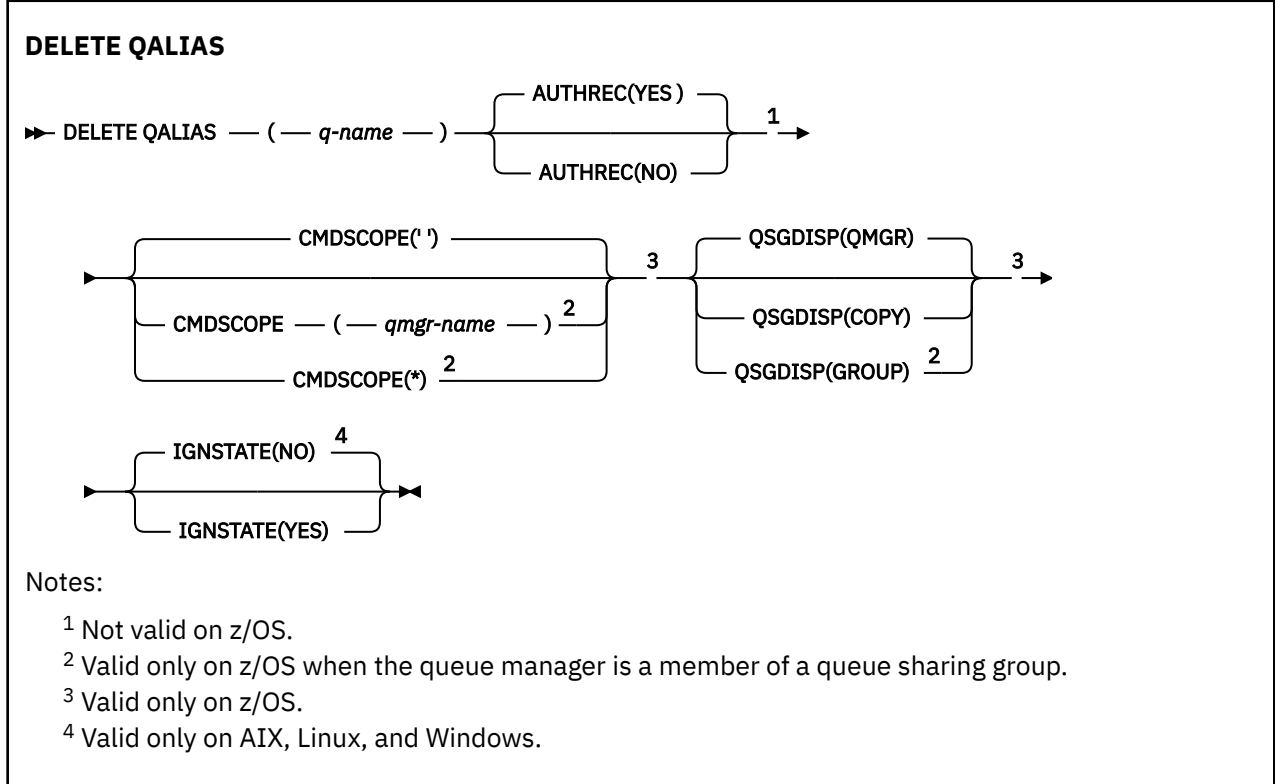
**YES**

The command succeeds regardless of whether the queue exists.

## DELETE QALIAS (delete an alias queue definition)

Use DELETE QALIAS to delete an alias queue definition.

**Synonym:** DELETE QA



The parameters are described in [“DELETE queues”](#) on page 620.

### Related concepts

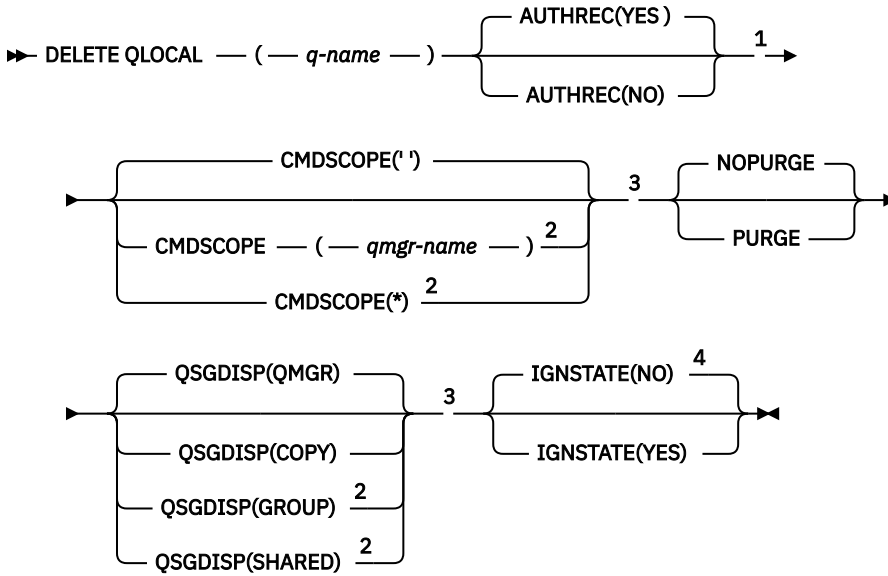
[Working with alias queues](#)

## DELETE QLOCAL (delete a local queue definition)

Use DELETE QLOCAL to delete a local queue definition. You can specify that the queue must not be deleted if it contains messages, or that it can be deleted even if it contains messages.

**Synonym:** DELETE QL

## DELETE QLOCAL



### Notes:

- 1 Not valid on z/OS.
- 2 Valid only on z/OS when the queue manager is a member of a queue sharing group.
- 3 Valid only on z/OS.
- 4 Valid only on AIX, Linux, and Windows.

The parameters are described in “DELETE queues” on page 620.

### Related tasks

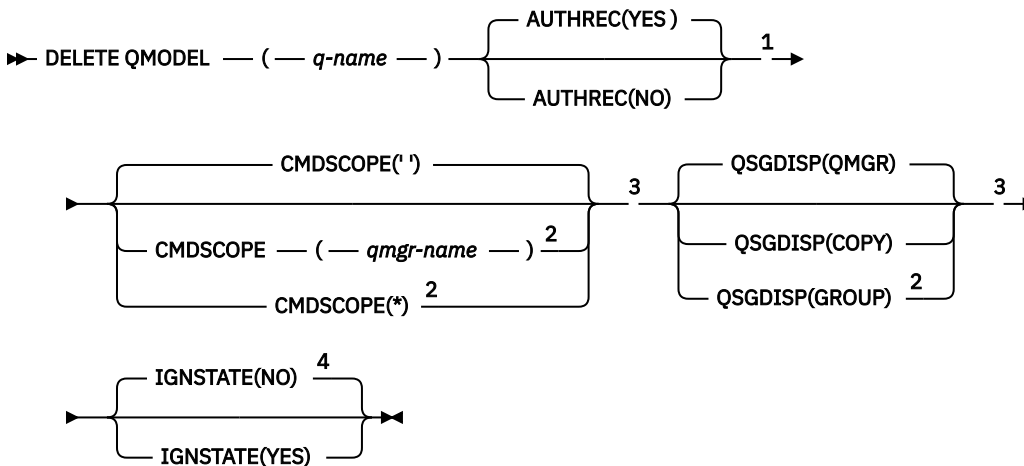
[Deleting a local queue](#)

## ***DELETE QMODEL (delete a model queue definition)***

Use **DELETE QMODEL** to delete a model queue definition.

**Synonym:** DELETE QM

## DELETE QMODEL



### Notes:



- <sup>1</sup> Not valid on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Valid only on AIX, Linux, and Windows.

The parameters are described in [“DELETE queues”](#) on page 620.

**Related concepts**

[Working with model queues](#)

**DELETE QREMOTE (delete local definition of remote queue)**

Use DELETE QREMOTE to delete a local definition of a remote queue. It does not affect the definition of that queue on the remote system.

**Synonym:** DELETE QR

**DELETE QREMOTE**

Notes:

- <sup>1</sup> Not valid on z/OS.
- <sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>3</sup> Valid only on z/OS.
- <sup>4</sup> Valid only on AIX, Linux, and Windows.

The parameters are described in [“DELETE queues”](#) on page 620.

**Multi DELETE SERVICE (delete a service definition) on Multiplatforms**

Use the MQSC command DELETE SERVICE to delete a service definition.

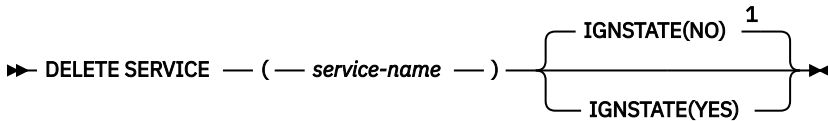
**Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Usage notes for DELETE SERVICE”](#) on page 626
- [“Keyword and parameter descriptions for DELETE SERVICE”](#) on page 626

**Synonym:**

## DELETE SERVICE



Notes:

<sup>1</sup> Valid only on AIX, Linux, and Windows.

## Usage notes for DELETE SERVICE

1. The command fails if an application has the specified service object open, or if the service is currently running.

## Keyword and parameter descriptions for DELETE SERVICE

### (*service-name*)

The name of the service definition to be deleted. This parameter is required. The name must be that of an existing service defined on the local queue manager.

V 9.3.0

ALW

### IGNSTATE

Specifies whether the command fails if the service does not exist. The value can be one of the following values:

#### NO

The command fails if the service does not exist. This is the default value.

#### YES

The command succeeds regardless of whether the service exists.

z/OS

## DELETE STGCLASS (delete a storage class) on z/OS

Use the MQSC command DELETE STGCLASS to delete a storage class definition.

## Using MQSC commands on z/OS

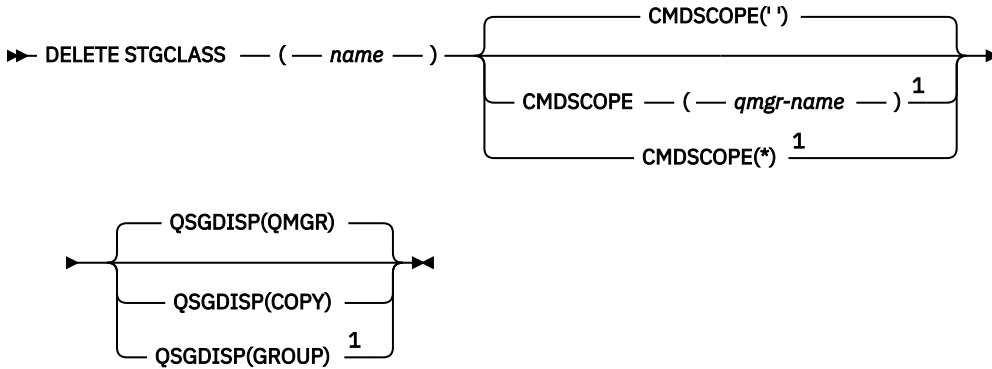
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DELETE STGCLASS” on page 627](#)

**Synonym:** DELETE STC

## DELETE STGCLASS



### Notes:

<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

## Parameter descriptions for DELETE STGCLASS

You must specify which storage class definition you want to delete.

All queues that use this storage class must be altered to use another storage class.

### (name)

The name of the storage class definition to be deleted. The name must be defined to the local queue manager.

The command fails unless all queues referencing the storage class are empty and closed.

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

''

The command runs on the queue manager on which it was entered. This is the default value.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### QSGDISP

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

### COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(COPY). Any object residing in the shared repository, or any object defined using a command that had the parameters QSGDISP(QMGR), is not affected by this command.

## GROUP

The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(GROUP). Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE STGCLASS(name) QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

## QMGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

## DELETE SUB (delete a durable subscription)

Use the MQSC command **DELETE SUB** to remove a durable subscription from the system. For a managed destination, any unprocessed messages left on the destination are removed.

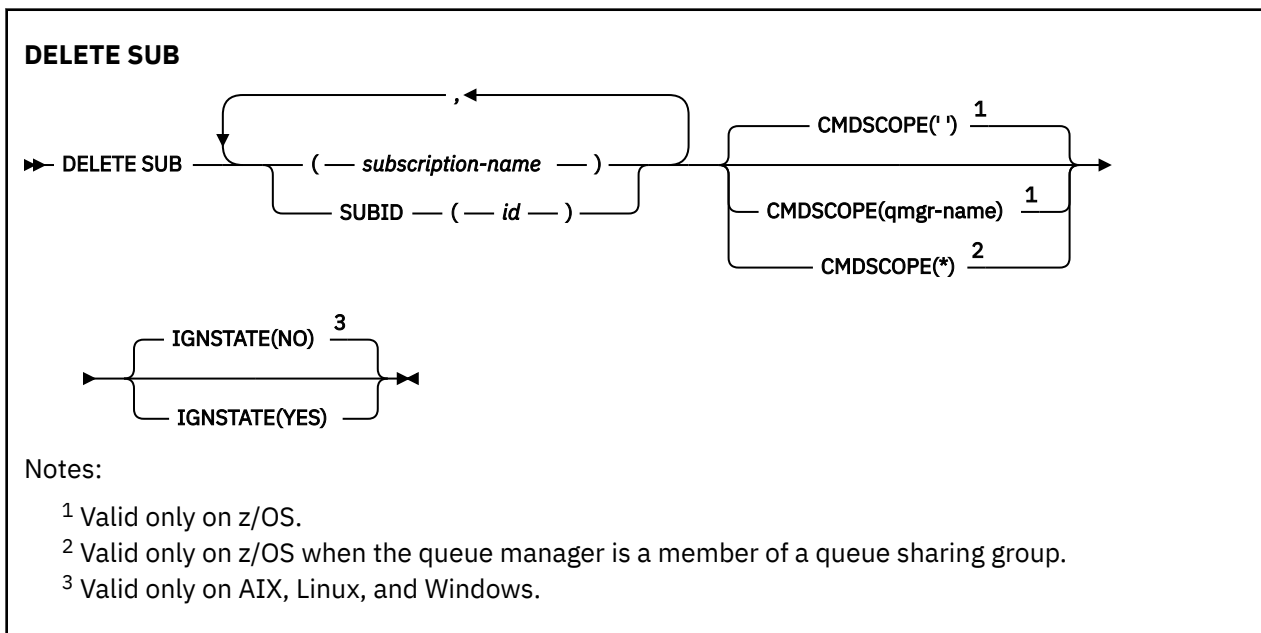
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [Usage notes for DELETE SUB](#)
- [“Parameter descriptions for DELETE SUB” on page 629](#)

**Synonym: DEL SUB**



## Usage notes for DELETE SUB

- You can specify either the name, the identifier, or both, of the subscription you want to delete.

Examples of valid forms:

```
DELETE SUB(xyz)
DELETE SUB SUBID(123)
DELETE SUB(xyz) SUBID(123)
```

- Successful completion of the command does not mean that the action completed. To check for true completion, see the [DELETE SUB step in Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for DELETE SUB

### *subscription-name*

The local name of the subscription definition to be deleted.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use **CMDSCOPE** as a filter keyword.

### **SUBID( *string* )**

The internal, unique key identifying a subscription.

### **IGNSTATE**

Specifies whether the command fails if the subscription does not exist. The value can be one of the following values:

**NO**

The command fails if the subscription does not exist. This is the default value.

**YES**

The command succeeds regardless of whether the subscription exists.

### **Related tasks**

[Deleting a subscription](#)

## DELETE TOPIC (delete an administrative topic node)

Use **DELETE TOPIC** to delete an IBM MQ administrative topic node.

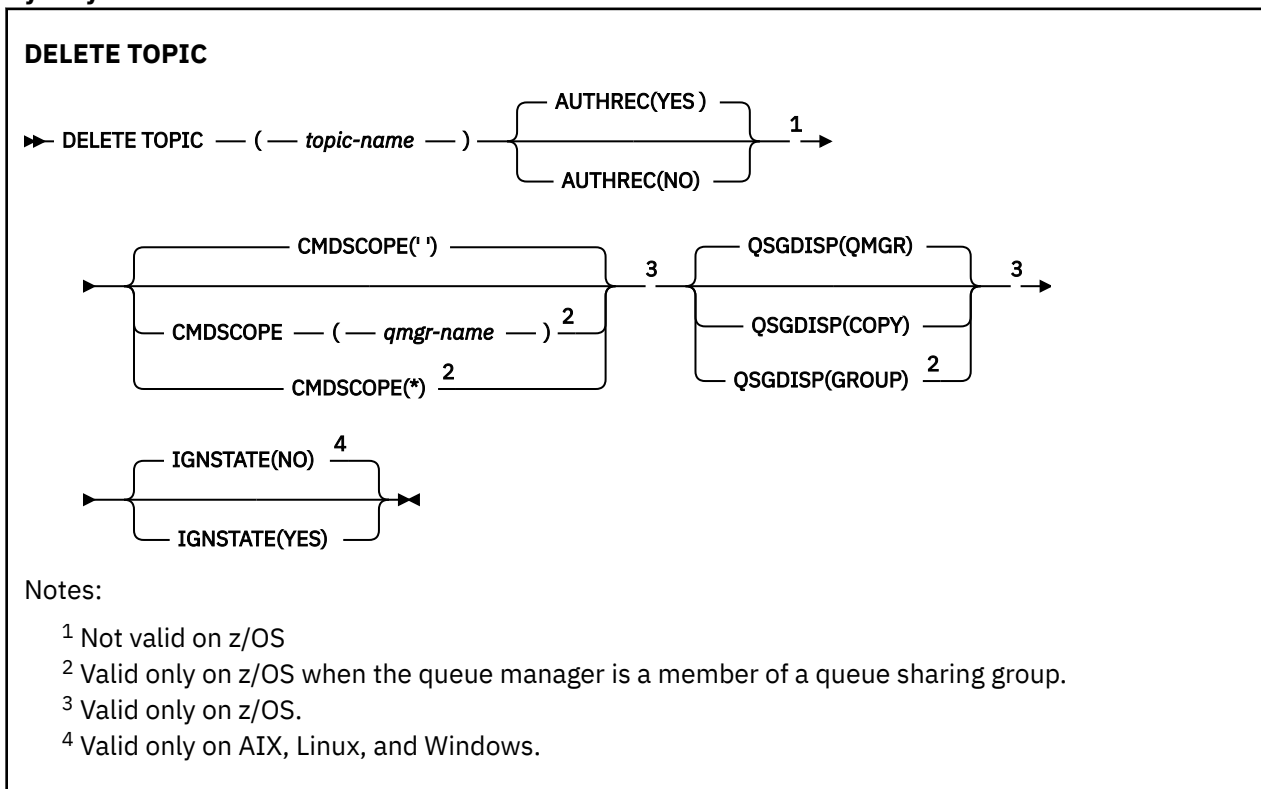
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DELETE TOPIC” on page 630](#)
- [“Parameter descriptions for DELETE TOPIC” on page 630](#)

**Synonym:** None



### Usage notes for DELETE TOPIC

- Successful completion of the command does not mean that the action completed. To check for true completion, see the [DELETE TOPIC](#) step in [Checking that async commands for distributed networks have finished](#).

### Parameter descriptions for DELETE TOPIC

#### **(topic-name)**

The name of the administrative topic object to be deleted. This parameter is required.

The name must be that of an existing administrative topic object.

#### **AUTHREC**

This parameter does not apply to z/OS

Specifies whether the associated authority record is also deleted:

**YES**

The authority record associated with the object is deleted. This is the default.

**NO**

The authority record associated with the object is not deleted.

**z/OS** **CMDSCOPE**

This parameter applies to only z/OS and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

**z/OS** **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves).

**COPY**

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(COPY). Any object residing in the shared repository, or any object defined using a command that had the parameters QSGDISP(QMGR), is not affected by this command.

**GROUP**

The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(GROUP). Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following command is generated and sent to all active queue managers in the queue sharing group to make, or delete, local copies on page set zero:

```
DELETE TOPIC(topic-name) QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

**QMGR**

The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameters QSGDISP(QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

Specifies whether the command fails if the topic does not exist. The value can be one of the following values:

**NO**

The command fails if the topic does not exist. This is the default value.

**YES**

The command succeeds regardless of whether the topic exists.

**Related tasks**

[Deleting an administrative topic definition](#)

**DISPLAY APSTATUS (display application status) on Multiplatforms**

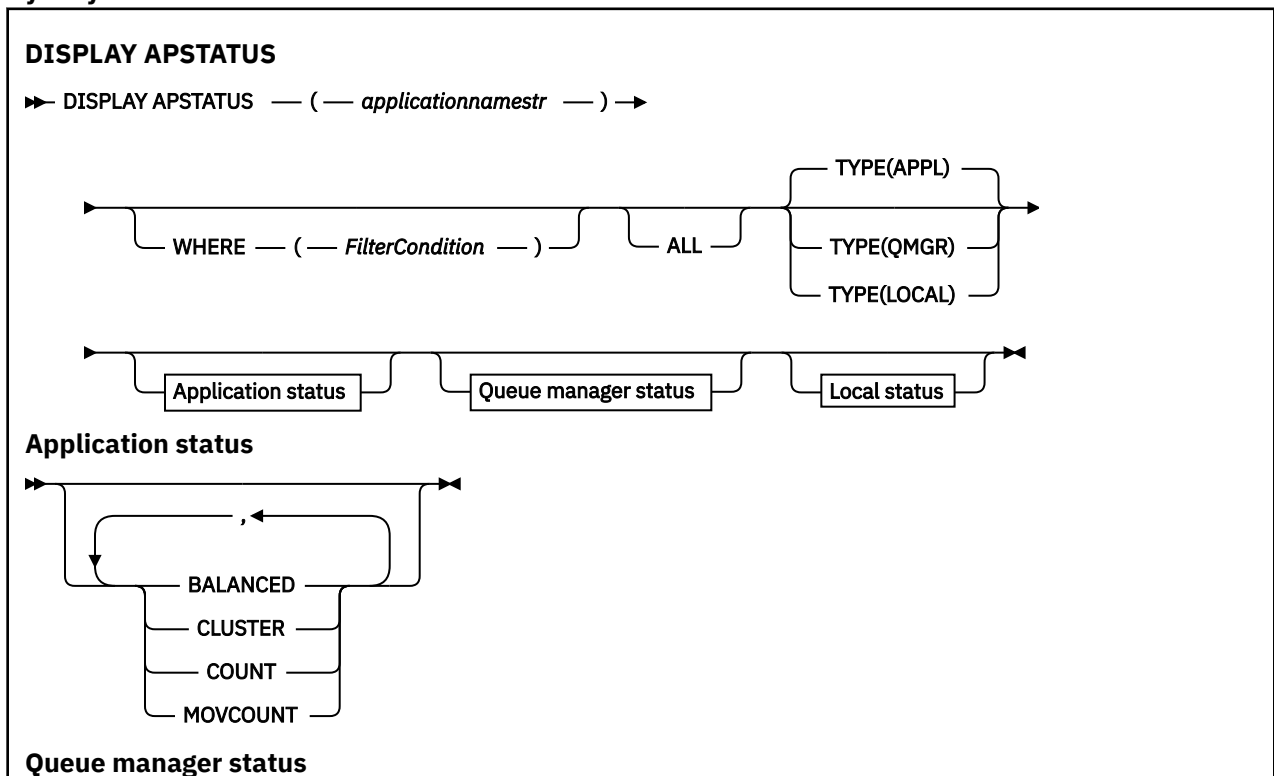
Use the MQSC command **DISPLAY APSTATUS** to display the status of one or more applications and application instances connected to a queue manager or a uniform cluster.

**Using MQSC commands**

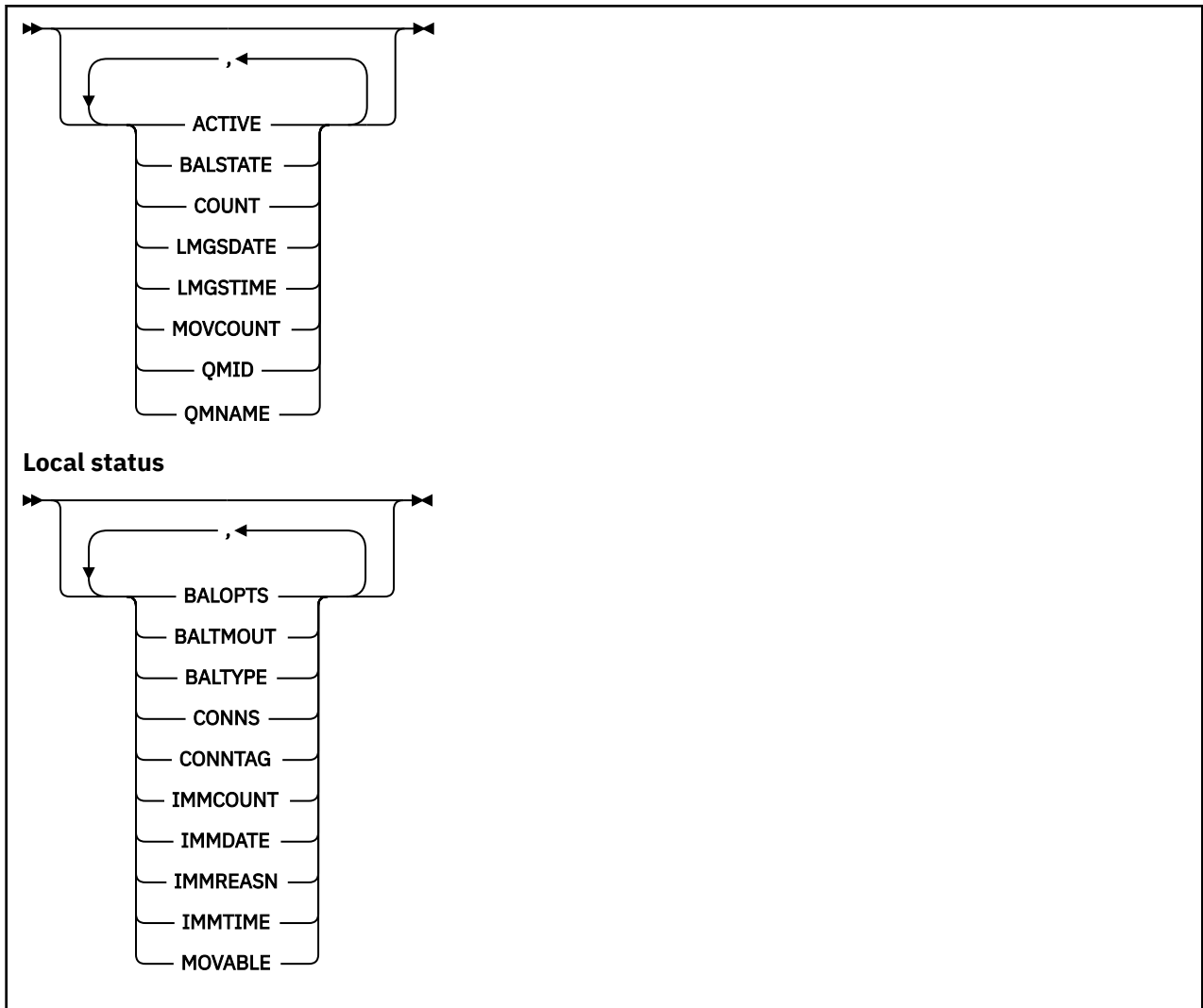
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Usage notes for DISPLAY APSTATUS” on page 633](#)
- [“Parameter descriptions for DISPLAY APSTATUS” on page 633](#)
- [“Application status” on page 635](#)
- [“Queue manager status” on page 635](#)
- [“Local status” on page 636](#)

**Synonym:** DIS APS







## Usage notes for DISPLAY APSTATUS

The application name parameter of the `DISPLAY APSTATUS` command matches against application names set by applications. See [using the application name in supported programming languages](#) for more information.

## Parameter descriptions for DISPLAY APSTATUS

The `DISPLAY APSTATUS` command requires an application name string value to determine which application details to return.

### *applicationnamestr*

The application name string can have one of the following values:

- A specific application name string value. For example, `DIS APSTATUS('myapp')` returns details of just the 'myapp' application
- A string containing one or more wildcard characters. For example, `DIS APSTATUS('*put*')` returns all applications which have 'put' in their application names.

To return a list of all user applications, use `DIS APSTATUS('*')`

To filter the list of applications returned, use the `WHERE` parameter. For example, `DIS APSTATUS('*put*') TYPE(APPL) WHERE(BALANCED eq NO)` returns application information about all unbalanced applications with 'put' in their name.

## WHERE

Specifies a filter condition to match only those applications or application instances that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

### filter-keyword

Any parameter that you can use with this **DISPLAY** command based on the **TYPE** option.

### operator

Determines whether a keywords value satisfies a condition on the given filter value. The operators are:

#### LT

Less than

#### GT

Greater than

#### EQ

Equal to

#### NE

Not equal to

#### LE

Less than or equal to

#### GE

Greater than or equal to

#### LK

Matches a generic string that you provide as a *filter-value*

#### NL

Does not match a generic string that you provide as a *filter-value*

#### CT

Contains a specified item. If the *filter-keyword* is a list, you can use this filter to display objects whose attributes contain the specified item.

#### EX

Does not contain a specified item. If the *filter-keyword* is a list, you can use this filter to display objects whose attributes do not contain the specified item.

### filter-value

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this value can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter, you can only use EQ or NE.

- A generic value. This value is a character string with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

## ALL

Use this parameter to display all attributes.

If you specify this parameter, any attributes that you request additionally have no effect; the command displays all attributes.

This value is the default, if you do not specify a generic name, and do not request any specific parameters.

## **TYPE**

Specifies the type of status information required:

### **APPL**

The command displays status information relating to each unique application name, which is the default if you do not provide a **TYPE** parameter. This represents a summary of the details from the local queue manager and any queue manager in the same uniform cluster.

### **QMGR**

The command displays status information relating to applications at a queue manager level, including the local queue manager and any queue manager in the same uniform cluster.

### **LOCAL**

The command displays status information for applications, for each application instance connected to the local queue manager.

## **Application status**

Application status parameters define the data that the command displays. You can specify these parameters in any order, but you must not specify the same parameter more than once:

### **BALANCED**

If the local queue manager is a member of a uniform cluster, this field gives an indication as to whether the number of application instances across the cluster is currently balanced, based on the last information received from the other queue managers in the cluster.

If the queue manager is not a member of a uniform cluster, this field shows NOTAPPLIC.

The value can be any of the following values:

#### **NO**

This application is not considered balanced in the uniform cluster.

#### **YES**

This application is considered balanced in the uniform cluster.

#### **NOTAPPLIC**

This application is not shared across a uniform cluster.

#### **UNKNOWN**

This is a temporary state, representing an application that has not yet undergone a scan to calculate whether it is balanced or not, on at least one queue manager, across the uniform cluster.

### **CLUSTER**

If the application details are being sent around a uniform cluster, this field displays the name of the uniform cluster, otherwise it shows a blank.

### **COUNT**

This displays the sum of the number of application instances for this application from the local queue manager, and all queue managers in the uniform cluster that have shared their application instance counts.

A queue manager not in a uniform cluster displays the count of local application instances.

### **MOVCOUNT**

This displays the sum of the number of movable application instances for this application from the local queue manager and all queue managers in the uniform cluster that have shared their application instance counts.

A queue manager not in a uniform cluster displays the count of local application instances that would be movable if put in a uniform cluster.

## **Queue manager status**

Queue manager status parameters define the data that the command displays. You can specify these parameters in any order, but you must not specify the same parameter more than once.

**ACTIVE**

Displays if the queue manager is considered active when balancing applications, which indicates whether information from that queue manager has been received recently.

**YES**

The queue manager is communicating with the uniform cluster. The local queue manager always shows **ACTIVE (YES)**.

**NO**

No status has recently been received from this queue manager. This could indicate a communication problem, or that the queue manager has been suspended from the uniform cluster.

**BALSTATE**

Indicates the state of the application instances on this queue manager, compared to the other queue managers in a uniform cluster. The value can be :

**HIGH**

There is a surplus of application instances.

**OK**

There is a balanced number of application instances.

**LOW**

There are not enough application instances.

**NOTAPPLIC**

The queue manager is not in a uniform cluster.

**UNKNOWN**

This is a temporary state representing an application that is new to the uniform cluster, and which has not yet undergone a scan to calculate whether it is balanced or not.

**COUNT**

Represents the count of application instances for this application on the queue manager.

**LMSGDATE**

The local date on which the local queue manager last received a published message from this queue manager, containing its application instance details.

**LMSGTIME**

The local time on which the local queue manager last received a published message from this queue manager, containing its application instance details.

**MOVCOUNT**

This represents the count of movable application instances for this application on the queue manager. Only application instances which are movable will be considered for rebalancing in a uniform cluster.

**QMID**

The queue manager identifier of the queue manager, that this information originated from.

**QMNAME**

The queue manager name that this information originated from. There will be one entry for the local queue manager, and one from each queue manager that has distributed information about this application in a uniform cluster.

**Local status**

Local status parameters define the data that the command displays. You can specify these parameters in any order, but you must not specify the same parameter more than once.

**V 9.3.0 BALOPTS**

The balancing options in effect for this application instance. Possible values are:

**MQBNO\_OPTIONS\_NONE**

No options are set.

## **MQBNO\_OPTIONS\_IGNORE\_TRANS**

This option allows applications to be rebalanced even if in the middle of a transaction.

### **V 9.3.0 BALTMOUT**

The time out value in effect for this application instance. Possible values are:

#### **NEVER**

No time out occurs.

#### **IMMEDIATE**

Time out occurs immediately

#### **Set value**

The timeout value in seconds, up to a maximum of 999999999 seconds.

### **V 9.3.0 BALTYPE**

The application type in effect for this application instance. Possible values are:

#### **MQBNO\_BALTYPE\_SIMPLE**

No specific rules should be applied beyond the defaults described in [Configuring the balancing behavior](#).

#### **MQBNO\_BALTYPE\_REQREP**

After each MQPUT call, a matching MQGET call is expected for a response message. Balancing is delayed until such a message is received, or the request message EXPIRY has been exceeded.

#### **MQBNO\_BALTYPE\_RAMANAGED**

Re-balancing requests are always dispatched immediately to the client, which re-balances at a point the client considers appropriate.

Note that different instances of the same application can provide different balancing options without causing any error.

#### **CONNS**

The number of connections (HCONNS) the application instance currently has.

#### **CONNTAG**

The connection tag of this application instance.

#### **IMMCOUNT**

The number of times this application instance has been asked to reconnect but has stayed connected. Any value higher than one indicates the application is not moving when requested.

#### **IMMDATE**

If the application instance is immovable for a fixed period, this indicates the date at which the instance will be eligible for moving again. If this has a value, the **IMMREASN** field should indicate why the connection is temporarily immovable. If the connection is not temporarily immovable, the value is blank.

#### **IMMREASN**

If the application instance is immovable, this indicates a reason as to why. If the application instance is movable, the value is blank. Only one **IMMREASN** is displayed even though multiple might apply; note that permanent statuses (such as NOTRECONN, NOTCLIENT) are displayed in preference to temporary values (such as MOVING, INTRANS).

The value can be any of the following values:

#### **APPNAMECHG**

This application instance cannot be moved as it is sharing a socket with a connection from an application instance which has a different application name.

### **V 9.3.0 INTRANS**

The application instance has successfully performed at least one MQI operation within syncpoint, and the timeout specified to forcibly rebalance an instance has not been reached.

#### **MOVING**

This application instance cannot be moved as it has recently been requested to move, and has not yet disconnected.

This status should be temporary. **IMMDATE** and **IMMTIME** indicate when this application instance is considered eligible to move again if this state unexpectedly persists.

#### **NONE**

This application instance is currently considered movable.

#### **V9.3.0 NOREDIRECT**

The client application has indicated that it cannot process redirect hints from the queue manager. This might be because the application isn't using a CCDT to connect to the queue manager.

#### **NOTCLIENT**

This application instance cannot be moved as it is not a client connection.

#### **NOTRECONN**

This application instance cannot be moved as it is not a reconnectable client connection.

#### **V9.3.0 REPLY**

The application instance is of type REQREP and has sent at least one request message for which a corresponding response has not been received. The timeout specified to forcibly rebalance an instance has not been reached.

#### **IMMTIME**

If the application instance is immovable for a fixed period, this indicates the time at which the instance will be eligible for moving again. If this has a value, the **IMMREASN** field should indicate why the connection is temporarily immovable. If the connection is not temporarily immovable, the value is blank.

#### **MOVABLE**

This indicates whether this application instance is considered movable or not.

#### **Related tasks**

[Monitoring application balancing](#)

#### **Related reference**

[“MQCMD\\_INQUIRE\\_APPL\\_STATUS \(inquire application status\) Response on Multiplatforms” on page 1157](#)

The response to the Inquire Application Status (**MQCMD\_INQUIRE\_APPL\_STATUS**) PCF command consists of the response header followed by the *ApplicationName* structure and the requested combination of attribute parameter structures (where applicable) for the requested *ApplicationStatusInfoType*.

### **z/OS DISPLAY ARCHIVE (display archive system information) on z/OS**

Use the MQSC command DISPLAY ARCHIVE to display archive system parameters and information.

#### **Using MQSC commands on z/OS**

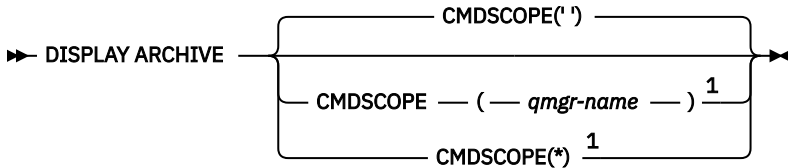
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DISPLAY ARCHIVE” on page 639](#)
- [“Parameter descriptions for DISPLAY ARCHIVE” on page 639](#)

**Synonym:** DIS ARC

## DISPLAY ARCHIVE



### Notes:

<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

## Usage notes for DISPLAY ARCHIVE

1. DISPLAY ARCHIVE returns a report that shows the initial values for the archiving parameters, and the current values as changed by the SET ARCHIVE command.

- Units in which primary and secondary space allocations are made (ALCUNIT).
- Prefix for first archive log data set name (ARCPFX1).
- Prefix for second archive log data set name (ARCPFX2).
- The retention period of the archive log data set in days (ARCRETN).
- List of route codes for messages to the operator about archive log data sets (ARCWRTC).
- Whether to send message to operator and wait for reply before trying to mount an archive log data set (ARCWTOR).
- Block size of archive log data set (BLKSIZE).
- Whether archive log data sets are cataloged in the ICF (CATALOG).
- Whether archive log data sets should be compacted (COMPACT).
- Primary space allocation for DASD data sets (PRIQTY).
- Whether archive log data sets are protected by ESM profiles when the data sets are created (PROTECT).
- Maximum time, in seconds, allowed for quiesce when ARCHIVE LOG with MODE(QUIESCE) specified (QUIESCE).
- Secondary space allocation for DASD data sets. See the ALCUNIT parameter for the units to be used (SECQTY).
- Whether the archive data set name should include a time stamp (TSTAMP).
- Device type or unit name on which the first copy of archive log data sets is stored (UNIT).
- Device type or unit name on which the second copy of archive log data sets is stored (UNIT2).

It also reports the status of tape units used for archiving.

For more details of these parameters, see [“SET ARCHIVE \(change archive system settings\) on z/OS” on page 915](#).

2. This command is issued internally by IBM MQ at the end of queue manager startup.

## Parameter descriptions for DISPLAY ARCHIVE

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered. This is the default value.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*


The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## DISPLAY AUTHINFO (display authentication information)

Use MQSC command DISPLAY AUTHINFO to display the attributes of an authentication information object.

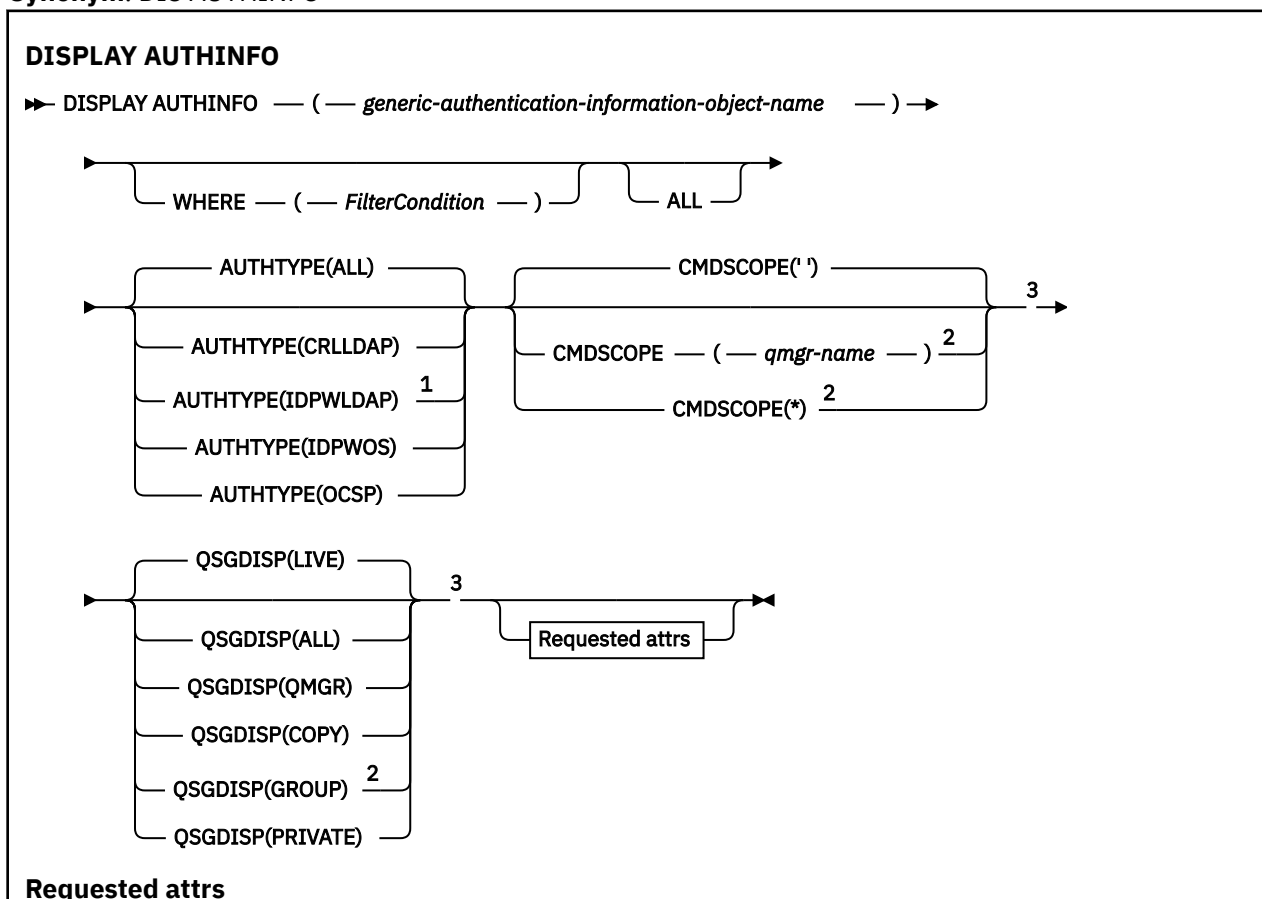
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

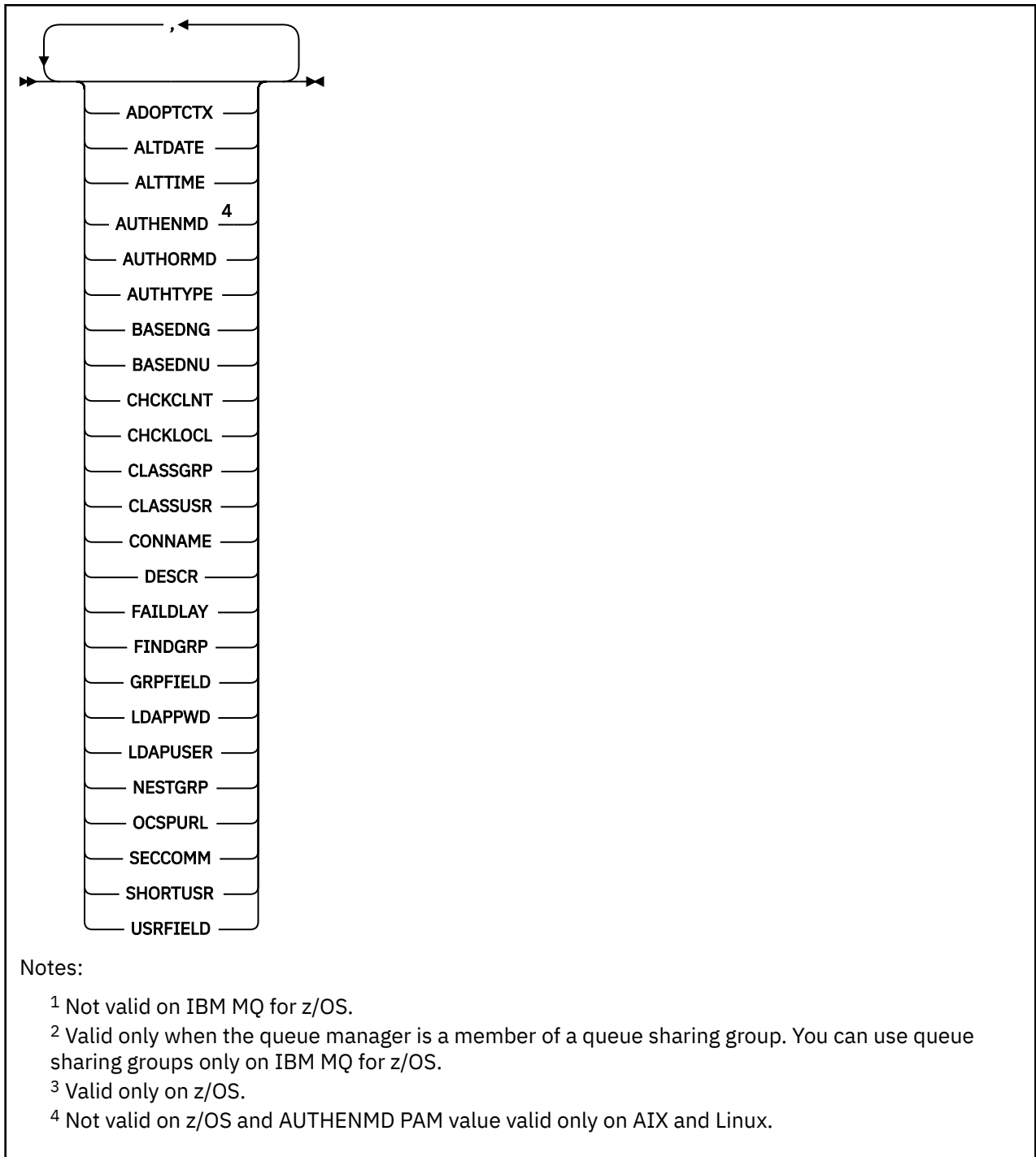
 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY AUTHINFO” on page 641](#)
- [“Requested parameters” on page 644](#)

**Synonym:** DIS AUTHINFO







## Parameter descriptions for DISPLAY AUTHINFO

### **(generic-authentication-object-name)**

The name of the authentication information object to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk (\*) matches all authentication information objects with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all authentication information objects.

### **WHERE**

Specify a filter condition to display only those authentication information objects that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

**filter-keyword**

Almost any parameter that can be used to display attributes for this DISPLAY command. However, you cannot use the CMDSCOPE or QSGDISP parameters as filter keywords.

**operator**

This is used to determine whether an authentication information object satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.  
You can use any of the operators except LK and NL.
- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. You cannot use a generic filter-value with numeric values. Only a single trailing wildcard character (asterisk) is permitted.

You can only use operators LK or NL for generic values on the DISPLAY AUTHINFO command.

**ALL**

Specify this to display all the parameters. If this parameter is specified, any parameters that are requested specifically have no effect; all parameters are still displayed.

This is the default if you do not specify a generic name and do not request any specific parameters.

**z/OS** On z/OS this is also the default if you specify a filter condition using the WHERE parameter, but on other platforms only requested attributes are displayed.

**z/OS** **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

**\***

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

**AUTHTYPE**

Specifies the authentication information type of the objects for which information is to be displayed. Values are:

**ALL**

This is the default value and displays information for objects defined with AUTHTYPE(CRLLDAP) and with AUTHTYPE(OCSP).

**CRLLDAP**

Displays information only for objects defined with AUTHTYPE(CRLLDAP).

**IDPWLDAP**

Displays information only for objects defined with AUTHTYPE(IDPWLDAP).

**IDPWOS**

Displays information only for objects defined with AUTHTYPE(IDPWOS).

**OCSP**

Displays information only for objects defined with AUTHTYPE(OCSP).

 **QSGDISP**

Specifies the disposition of the objects for which information is to be displayed. Values are:

**LIVE**

This is the default value and displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

**ALL**

Displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP).

If QSGDISP(LIVE) is specified or defaulted, or if QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

**COPY**

Displays information only for objects defined with QSGDISP(COPY).

**GROUP**

Displays information only for objects defined with QSGDISP(GROUP). This is allowed only if there is a shared queue manager environment.

**PRIVATE**

Displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY). Note that QSGDISP(PRIVATE) displays the same information as QSGDISP(LIVE).

**QMGR**

Displays information only for objects defined with QSGDISP(QMGR).

QSGDISP displays one of the following values:

**QMGR**

The object was defined with QSGDISP(QMGR).

**GROUP**

The object was defined with QSGDISP(GROUP).

**COPY**

The object was defined with QSGDISP(COPY).

You cannot use QSGDISP as a filter keyword.

**Requested parameters**

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

The default, if no parameters are specified (and the ALL parameter is not specified) is that the object names and their AUTHTYPES, and, on z/OS, their QSGDISPs, are displayed.

**ADOPTCTX**

Displays the presented credentials as the context for this application.

**ALTDATE**

The date on which the definition was last altered, in the form yyyy-mm-dd

**ALLTIME**

The time at which the definition was last altered, in the form hh.mm.ss

**AUTHENMD**

Authentication method. Possible values are:

**OS**

Displays the traditional UNIX password verification method permissions.

**PAM**

Displays the Pluggable Authentication Method permissions.

You can set the PAM value only on AIX and Linux platforms.

**AUTHORMD**

Displays the authorization method. Possible values are:

**OS**

Use operating system groups to determine permissions associated with a user.

**SEARCHGRP**

A group entry in the LDAP repository contains an attribute listing the Distinguished Name of all users belonging to that group.

**SEARCHUSR**

A user entry in the LDAP repository contains an attribute listing the Distinguished Name of all the groups to which the specified user belongs.

**SRCHGRPSN**

A group entry in the LDAP repository contains an attribute listing the short user name of all users belonging to that group.

**AUTHTYPE**

The type of the authentication information

**BASEDNG**

Displays the Base DN for groups.

**BASEDNU**

Displays the base distinguished name to search for users within the LDAP server.

**CHKLOCL or CHKCLNT**

These attributes are valid only for an **AUTHTYPE** of *IDPWOS* or *IDPWLDAP*. The possible values are:

**NONE**

Displays all locally bound applications that have no user ID and password authentication.


**OPTIONAL**

Displays the user IDs and passwords provided by an application. Note that it is not mandatory to provide these attributes. This option might be useful during migration, for example.

**REQUIRED**

Displays all applications providing a valid user ID and password.

**REQDADM**

Displays privileged users supplying a valid user ID and password, Non-privileged users are treated as with the OPTIONAL setting. See also the following note.  (This setting is not allowed on z/OS systems.)

**CLASSGRP**

Displays the LDAP object class for group records.

**CLASSUSR**

Displays the LDAP object class for user records within the LDAP repository.

**CONNAME**

The host name, IPv4 dotted decimal address, or IPv6 hexadecimal notation of the host on which the LDAP server is running. Applies only to objects with AUTHTYPE(CRLLDAP) or AUTHTYPE(IDPWLDAP).

**DESCR**

Description of the authentication information object.

**FAILDLAY**

Delay in seconds before an authentication failure is returned to an application.


**FINDGRP**

Displays the name of the attribute within an LDAP entry to determine group membership.

**GRPFIELD**

Displays the LDAP attribute that represents a simple name for the group.

**LDAPPWD**

Password associated with the Distinguished Name of the user on the LDAP server. If nonblank, this is displayed as asterisks  on all platforms except z/OS. Applies only to objects with AUTHTYPE(CRLLDAP) or AUTHTYPE(IDPWLDAP).

**LDAPUSER**

Distinguished Name of the user on the LDAP server. Applies only to objects with AUTHTYPE(CRLLDAP) or AUTHTYPE(IDPWLDAP).

**NESTGRP**

Displays whether a group is a member of another group..

**OCSPURL**

The URL of the OCSP responder used to check for certificate revocation. Applies only to objects with AUTHTYPE(OCSP).

**SECCOMM**

Displays the method used to connect the LDAP server.

**SHORTUSR**

Displays the user record being used as a short name.

**USRFIELD**

Displays the user record being used in the LDAP user record, only if the user ID does not contain a qualifier.

See [“Usage notes for DEFINE AUTHINFO” on page 448](#) for more information about individual parameters.

## **DISPLAY AUTHREC (display authority records) on Multiplatforms**

Use the MQSC command DISPLAY AUTHREC to display the authority records associated with a profile name.

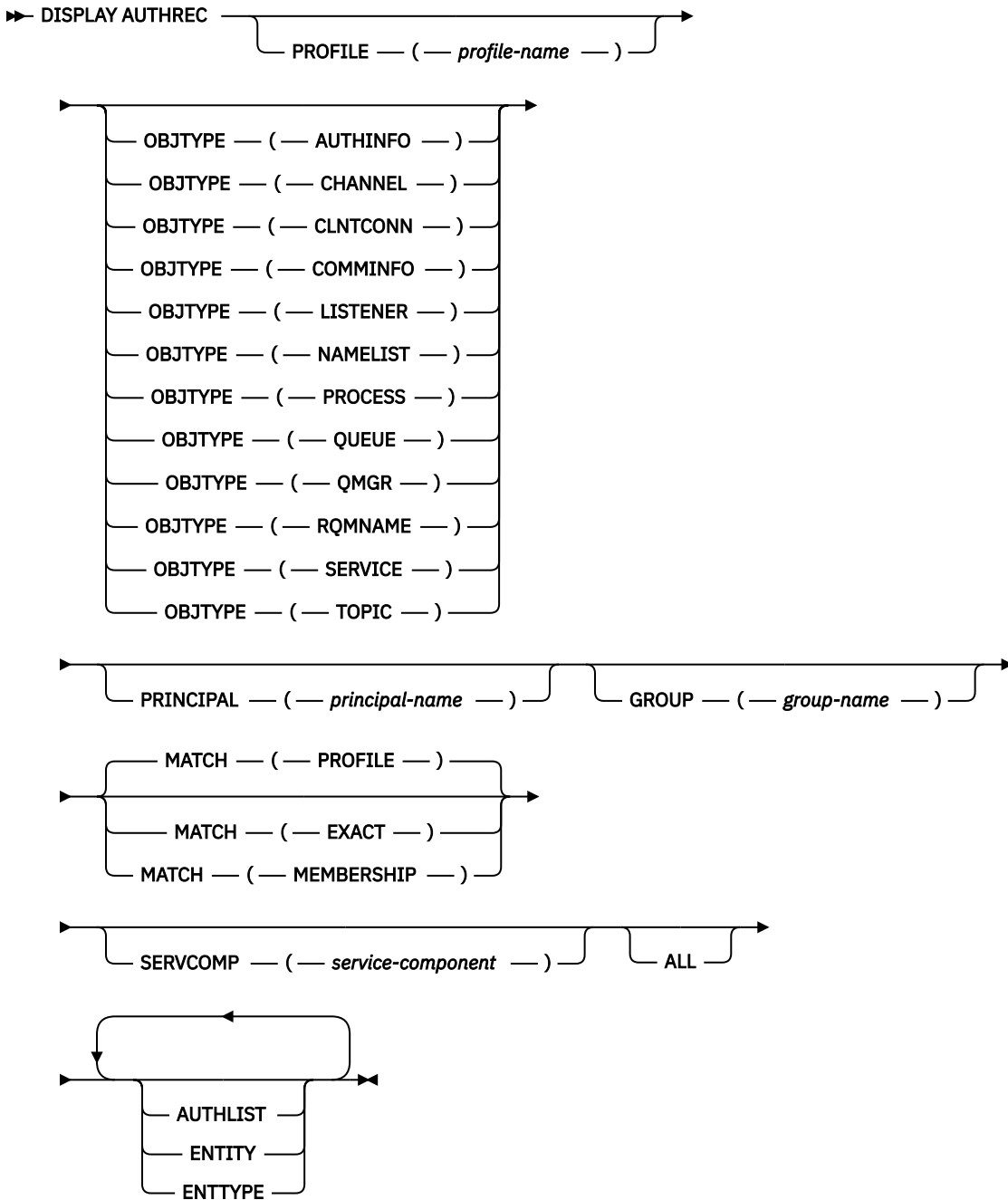
### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions” on page 647](#)
- [“Requested parameters” on page 649](#)

**Synonym:** DIS AUTHREC

## DISPLAY AUTHREC



### Parameter descriptions

#### PROFILE(*profile-name*)

The name of the object or generic profile for which to display the authority records. If you omit this parameter, all authority records that satisfy the values of the other parameters are displayed.

#### OBJTYPE

The type of object referred to by the profile. Specify one of the following values:

##### AUTHINFO

Authentication information record

**CHANNEL**

Channel

**CLNTCONN**

Client connection channel

**COMMINFO**

Communication information object

**LISTENER**

Listener

**NAMELIST**

Namelist

**PROCESS**

Process

**QUEUE**

Queue

**QMGR**

Queue manager

**RQMNAME**

Remote queue manager

**SERVICE**

Service

**TOPIC**

Topic

If you omit this parameter, authority records for all object types are displayed.

**PRINCIPAL(*principal-name*)**

A principal name. This is the name of a user for whom to retrieve authorizations to the specified object. On IBM MQ for Windows, the name of the principal can optionally include a domain name, specified in this format: `user@domain`.

This parameter cannot be specified with `GROUP`.

**GROUP(*group-name*)**

A group name. This is the name of the user group on which to make the inquiry. You can specify one name only and it must be the name of an existing user group.

**Windows** For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following formats:

```
GroupName@domain  
domain\GroupName
```

This parameter cannot be specified with `PRINCIPAL`.

**MATCH**

Specify this parameter to control the set of authority records that is displayed. Specify one of the following values:

**PROFILE**

Return only those authority records which match the specified profile, principal, and group names. This means that a profile of `ABCD` results in the profiles `ABCD`, `ABC*`, and `AB*` being returned (if `ABC*` and `AB*` have been defined as profiles). If the profile name is a generic profile, only authority records which exactly match the specified profile name are returned. If a principal is specified, no profiles are returned for any group in which the principal is a member; only the profiles defined for the specified principal or group.

This is the default value.



## **MEMBERSHIP**

Return only those authority records which match the specified profile, and the entity field of which matches the specified principal and the profiles pertaining to any groups in which the principal is a member that contribute to the cumulative authority for the specified entity.

If this option is specified, the PROFILE and OBJTYPE parameters must also be specified. In addition, either the PRINCIPAL or GROUP parameter must also be supplied. If OBJTYPE(QMGR) is specified, the profile name is optional.

## **EXACT**

Return only those authority records which exactly match the specified profile name and EntityName. No matching generic profiles are returned unless the profile name is, itself, a generic profile. If a principal is specified, no profiles are returned for any group in which the principal is a member; only the profile defined for the specified principal or group.

## **SERVCOMP(service-component)**

The name of the authorization service for which information is to be displayed.

If you specify this parameter, it specifies the name of the authorization service to which the authorizations apply. If you omit this parameter, the inquiry is made to the registered authorization services in turn in accordance with the rules for chaining authorization services.

## **ALL**

Specify this parameter to display all of the authorization information available for the entity and the specified profile.

## **Requested parameters**

You can request the following information about the authorizations:

### **AUTHLIST**

Specify this parameter to display the list of authorizations.

### **ENTITY**

Specify this parameter to display the entity name.

### **ENTTYPE**

Specify this parameter to display the entity type.

### **Related reference**

[“dmpmqaut \(dump MQ authorizations\)” on page 55](#)

Dump a list of current authorizations for a range of IBM MQ object types and profiles.

[“setmqaut \(grant or revoke authority\)” on page 193](#)

Change the authorizations to a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

[“SET AUTHREC \(set authority records\) on Multiplatforms” on page 920](#)

Use the MQSC command SET AUTHREC to set authority records associated with a profile name.

## **ALW DISPLAY AUTHSERV (display authorization services information) on AIX, Linux, and Windows**

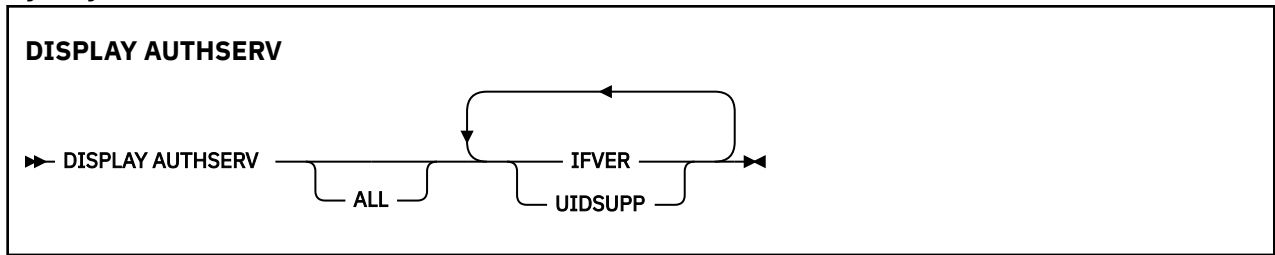
Use the MQSC command DISPLAY AUTHSERV to display information about the level of function supported by the installed authorization services.

### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions” on page 650](#)
- [“Requested parameters” on page 650](#)

**Synonym:** DIS AUTHSERV



## Parameter descriptions

### ALL

Specify this parameter to display all the information for each authorization service.

### Requested parameters

You can request the following information for the authorization service:

#### IFVER

Specify this parameter to display the current interface version of the authorization service.

#### UIDSUPP

Specify this parameter to display whether the authorization service supports user IDs.

## **DISPLAY CFSTATUS (display CF application structure status) on z/OS**

Use the MQSC command **DISPLAY CFSTATUS** to display the status of one or more CF application structures. This command is valid only on IBM MQ for z/OS when the queue manager is a member of a queue sharing group.

### Using MQSC commands on z/OS

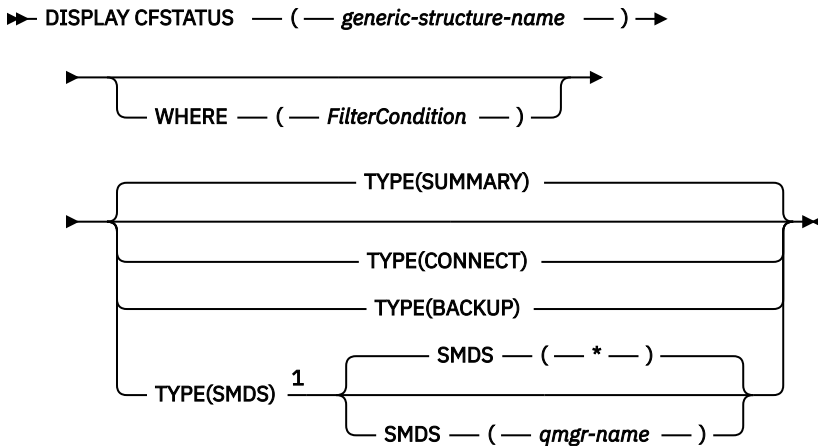
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Keyword and parameter descriptions for DISPLAY CFSTATUS” on page 651](#)
- [“Summary status” on page 653](#)
- [“Connection status” on page 654](#)
- [“Backup status” on page 655](#)
- [“SMDS status” on page 656](#)

**Synonym:** DIS CFSTATUS

## DISPLAY CFSTATUS



### Notes:

- <sup>1</sup> This option is only supported when the CFSTRUCT is defined with OFFLOAD(SMDS).

## Keyword and parameter descriptions for DISPLAY CFSTATUS

The name of the application structure for the status information to be displayed must be specified. This can be a specific application structure name or a generic name. By using a generic name, it is possible to display either:

- status information for all application structure definitions
- status information for one or more application structures that match the specified name

The type of status information to be returned can also be specified. This can be:

- summary status information for the application structure in the queue sharing group
- connection status information for each queue manager in the queue sharing group for each matching application structure name
- backup status information for each backup taken for each matching application structure defined in the queue sharing group

### (*generic-structure-name*)

The 12-character name of the CF application structure to be displayed. A trailing asterisk (\*) matches all structure names with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all structure names.

The CF structure name must be defined within the queue sharing group.

The CFSTATUS generic name can be the administration CF structure name (CSQ\_ADMIN) or any generic form of this name. Data for this structure, however, is only displayed when TYPE is set to SUMMARY.

### WHERE

Specify a filter condition to display status information for those CF application structures that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### filter-keyword

Almost any parameter that is returned by this DISPLAY command. However, you cannot use the TYPE parameter as a filter keyword.

#### operator

This is used to determine whether a CF application structure satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which contain the specified item.

**EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not contain the specified item.

**CTG**

Contains an item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which match the generic string.

**EXG**

Does not contain any item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not match the generic string.

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, GE, only. However, if the value is one from a possible set of values returnable on a parameter (for example, the value ACTIVE on the STATUS parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string in the QMNAME parameter) with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. The value can be explicit or, if it is a character value, it can be explicit or generic. If it is explicit, use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed. If it is generic, use CTG or EXG as the operator. If ABC\* is specified with the operator CTG, all items where one of the attribute values begins with ABC are listed.

**TYPE**

Specifies the type of status information required to be displayed. Values are:

**SUMMARY**

Display summary status information for each application structure. This is the default.

**CONNECT**

Display connection status information for each application structure for each active queue manager.

**BACKUP**

Display backup status information for each application structure.

**SMDS**

Display shared message data set information.

**SMDS****qmgr-name**

Specifies the queue manager for which the shared message data set status is to be displayed.

\*

Displays the status for all shared message data sets associated with the specified CFSTRUCT except those which have both STATUS(NOTFOUND) and ACCESS(ENABLED).

**Summary status**

For summary status, the following information is returned for each structure that satisfies the selection criteria:

- The name of the application structure matching the generic name.
- The type of information returned.

**CFTYPE**

The CF structure type. This is one of the following:

**ADMIN**

This is the CF administration structure.

**APPL**

This is a CF application structure.

**STATUS**

The status of the CF application structure. This is one of the following:

**ACTIVE**

The structure is active.

**FAILED**

The structure has failed.

**NOTFOUND**

The structure is not allocated in the CF, but has been defined to Db2. Check and resolve any messages in the job log about this structure.

**INBACKUP**

The structure is in the process of being backed-up.

**INRECOVER**

The structure is in the process of being recovered.

**UNKNOWN**

The status of the CF structure is not known because, for example, Db2 might be unavailable.

**SIZEMAX (size)**

The size in kilobytes of the application structure.

**SIZEUSED (integer)**

The percentage of the size of the application structure that is in use. Therefore SIZEUSED(25) would indicate that a quarter of the space allocated to this application structure is in use.

**ENTSMAX (integer)**

The number of CF list entries defined for this application structure.

**Note:** The number does not include any entries that are in storage class memory (SCM), and which might have been allocated to the structure.

**ENTSUSED (integer)**

The number of CF list entries for this application structure that are in use.

**Note:** The number does not include any entries that are in storage class memory (SCM), and which might have been allocated to the structure.

**FAILTIME (time)**

The time that this application structure failed. The format of this field is hh .mm .ss. This parameter is only applicable when the CF structure is in FAILED or INRECOVER state. If the structure is not in a failed state, this is displayed as FAILTIME().

**FAILDATE (date)**

The date that this application-structure failed. The format of this field is yyyy-mm-dd. This parameter is only applicable when the CF structure is in FAILED or INRECOVER state. If the structure is not in a failed state, then this is displayed as FAILDATE().

**OFFLDUSE**

This indicates whether offloaded large message data potentially exists in shared message data sets, Db2 or both.

When the offload method is switched, the previous offload method needs to remain available for retrieving and deleting old messages, so the OFFLDUSE status is changed to indicate BOTH. When a queue manager disconnects normally from a structure that has OFFLDUSE(BOTH) it checks whether there still are any messages which were stored using the old offload method. If not, it changes the OFFLDUSE status to match the current offload method and issues message CSQE245I to indicate that the switch is complete.

This parameter is one of the following:

**NONE**

No offloaded large messages are present.

**SMDS**

Offloaded large messages can exist in shared message data sets.

**Db2**

Offloaded large messages can exist in Db2.

**BOTH**

Offloaded large messages can exist both in shared message data sets and in Db2.

**Connection status**

For connection status, the following information is returned for each connection to each structure that satisfies the selection criteria:

- The name of the application structure matching the generic name.
- The type of information returned.

**QMNAME (qmgrname)**

The queue manager name.

**SYSNAME (systemname)**

The name of the z/OS image of the queue manager that last connected to the application structure. These can be different across queue managers depending on the customer configuration setup.

**STATUS**

A status indicating whether this queue manager is connected to this application structure. This is one of the following:

**ACTIVE**

The structure is connected to this queue manager.

**FAILED**

The queue manager connection to this structure has failed.

**NONE**

The structure has never been connected to this queue manager.

**UNKNOWN**

The status of the CF structure is not known.

**FAILTIME (time)**

The time that this queue manager lost connectivity to this application structure. The format of this field is hh.mm.ss. This parameter is only applicable when the CF structure is in FAILED state. If the structure is not in a failed state, this is displayed as FAILTIME().

**FAILDATE (date)**

The date that this queue manager lost connectivity to this application structure. The format of this field is yyyy-mm-dd. This parameter is only applicable when the CF structure is in FAILED state. If the structure is not in a failed state, this is displayed as FAILDATE().

**Backup status**

For backup status, the following information is returned for each structure that satisfies the selection criteria:

- The name of the application structure matching the generic name.
- The type of information returned.

**STATUS**

The status of the CF application structure. This is one of the following:

**ACTIVE**

The structure is active.

**FAILED**

The structure has failed.

**NONE**

The structure is defined as RECOVER(YES), but has never been backed up.

**INBACKUP**

The structure is in the process of being backed-up.

**INRECOVER**

The structure is in the process of being recovered.

**UNKNOWN**

The status of the CF structure is not known.

**QMNAME (qmgrname)**

The name of the queue manager that took the last successful backup for this application structure.

**BKUPTIME (time)**

The end time of the last successful backup taken for this application structure. The format of this field is hh.mm.ss.

**BKUPDATE (date)**

The date of the last successful backup taken for this application structure. The format of this field is yyyy-mm-dd.

**BKUPSIZE (size)**

The size in megabytes of the last successful backup taken for this application structure.

**BKUPSRBA (hexadecimal)**

This is the backup data set start RBA for the start of the last successful backup taken for this application structure.

**BKUPERBA (hexadecimal)**

This is the backup data set end RBA for the end of the last successful backup taken for this application structure.

**LOGS (qmgrname-list)**

This is the list of queue managers, the logs of which are required to perform a recovery.

**FAILTIME (time)**

The time that this CF structure failed. The format of this field is hh.mm.ss. This parameter is only applicable when the CF structure is in FAILED state. If the structure is not in a failed state, this is displayed as FAILTIME().

**FAILDATE (date)**

The date that this CF structure failed. The format of this field is yyyy-mm-dd. This parameter is only applicable when the CF structure is in FAILED state. If the structure is not in a failed state, this is displayed as FAILDATE().

**SMDS status**

The DISPLAY CFSTATUS command with TYPE(SMDS) displays status information relating to one or more shared message data sets associated with a specific application structure.

The following data is returned for each selected data set:

**SMDS**

The queue manager name which owns the shared message data set for which properties are being displayed

**STATUS**

The current status of the shared message data set. This is one of the following:

**NOTFOUND**

The data set has never been used, or the attempt to open it for the first time failed. Check and resolve any messages in the job log about this structure.

**NEW**

The data set is being opened and initialized for the first time, ready to be made active.

**ACTIVE**

The data set is available for normal use.

**FAILED**

The data set is in an unusable state and probably requires recovery.

**INRECOVER**

Data set recovery (using RECOVER CFSTRUCT) is in progress.

**RECOVERED**

The data set has been recovered or otherwise repaired, and is ready for use again, but requires some restart processing the next time it is opened. This restart processing ensures that obsolete references to any deleted messages have been removed from the coupling facility structure before the data set is made available again. The restart processing also rebuilds the data set space map.

**EMPTY**

The data set contains no messages. The data set is put into this state if it is closed normally by the owning queue manager at a time when it does not contain any messages. It can also be put into EMPTY state when the previous data set contents are to be discarded because the application structure has been emptied (using **RECOVER CFSTRUCT** with TYPE PURGE or, for a nonrecoverable structure only, by deleting the previous instance of the structure). The next time the data set is opened by its owning queue manager, the space map is reset to empty, and the status is changed to ACTIVE. As the previous data set contents are no longer required, a data set in this state can be replaced with a newly allocated data set, for example to change the space allocation or move it to another volume.

**ACCESS**

The current availability state of the shared message data set. This parameter is one of the following:



**ENABLED**

The data set can be used, and no error has been detected since the time that it was enabled. If the data set has STATUS(RECOVERED) it can only be opened by the owning queue manager for restart purposes, but if it has STATUS(ACTIVE) all queue managers can open it.

**SUSPENDED**

The data set is unavailable because of an error.

This occurs specifically when the STATUS is set to FAILED either because of an error accessing the data set, or using the ALTER SMDS command.

The queue manager can try to enable access again automatically if the error might no longer be present, for example when recovery completes, or if the status is manually set to RECOVERED. Otherwise, it can be enabled again by a command in order to retry the action which originally failed.

**DISABLED**

The shared message data set cannot be used because it has been explicitly disabled using a command. It can only be enabled again by using another command to enable it. For more information, see [“RESET SMDS \(reset shared message data sets\) on z/OS” on page 905](#).

**RCVDATE**

The recovery start date.

If recovery is currently enabled for the data set, this indicates the date when it was activated, in the form yyyy-mm-dd. If recovery is not enabled, this is displayed as RCVDATE().

**RCVTIME**

The recovery start time.

If recovery is currently enabled for the data set, this indicates the time when it was activated, in the form hh.mm.ss. If recovery is not enabled, this is displayed as RCVTIME().

**FAILDATE**

The failure date.

If the data set was put into a failed state, and has not yet been restored to the active state, this indicates the date when the failure was indicated, in the form yyyy-mm-dd. If the data set is in the active state, this is displayed as FAILDATE().

**FAILTIME**

The failure time.

If the data set was put into a failed state and has not yet been restored to the active state, this indicates the time when the failure was indicated, in the form hh.mm.ss. If the data set is in the active state, this is displayed as FAILTIME().

## **DISPLAY CFSTRUCT (display CF application structure settings) on z/OS**

Use the MQSC command DISPLAY CFSTRUCT to display the attributes of one or more CF application structures. This command is valid only on z/OS when the queue manager is a member of a queue sharing group.

### **Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

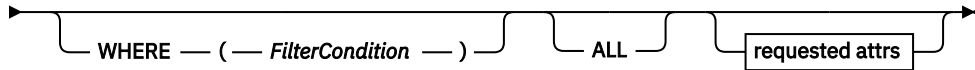
- [Syntax diagram](#)
- [“Usage notes for DISPLAY CFSTRUCT” on page 658](#)

- “Keyword and parameter descriptions for DISPLAY CFSTRUCT” on page 658
- “Requested parameters” on page 660

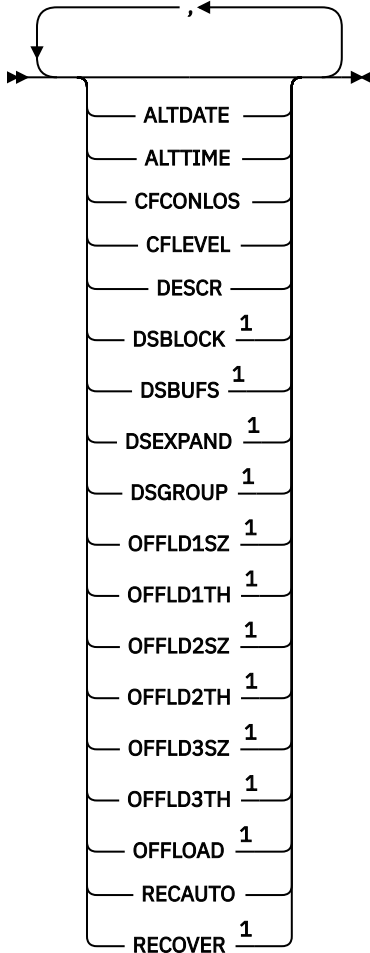
**Synonym:** DIS CFSTRUCT

## DISPLAY CFSTRUCT

► DISPLAY CFSTRUCT — ( — *generic-structure-name* — ) →



### Requested attrs



Notes:

<sup>1</sup> For more information about this parameter, see [Planning your coupling facility and offload storage environment](#).

## Usage notes for DISPLAY CFSTRUCT

1. The command cannot specify the CF administration structure (CSQ\_ADMIN).

## Keyword and parameter descriptions for DISPLAY CFSTRUCT

The name of the application structure to be displayed must be specified. This can be a specific application structure name or a generic name. By using a generic name, it is possible to display either:

- all application structure definitions
- one or more application structures that match the specified name

**( *generic-structure-name* )**

The 12-character name of the CF application structure to be displayed. A trailing asterisk (\*) matches all structure names with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all structure names.

The CF structure name must be defined within the queue sharing group.

**WHERE**

Specify a filter condition to display only those CF application structures that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

**filter-keyword**

Any parameter that can be used to display attributes for this DISPLAY command.

**operator**

This is used to determine whether a CF application structure satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use any of the operators except LK and NL. However, if the value is one from a possible set of values returnable on a parameter (for example, the value YES on the RECOVER parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

You can only use operators LK or NL for generic values on the DISPLAY CFSTRUCT command.

**ALL**

Specify this to display all attributes. If this keyword is specified, any attributes that are requested specifically have no effect; all attributes are still displayed.

This is the default behavior if you do not specify a generic name and do not request any specific attributes.

## Requested parameters

Specify one or more attributes that define the data to be displayed. The attributes can be specified in any order. Do not specify the same attribute more than once.

The default, if no parameters are specified (and the ALL parameter is not specified) is that the structure names are displayed.

### **ALTDATE**

The date on which the definition was last altered, in the form yyyy-mm-dd.

### **ALTTIME**

The time at which the definition was last altered, in the form hh.mm.ss.

### **CFCONLOS**

The action to be taken when the queue manager loses connectivity to the CF application structure.

### **CFLEVEL**

Indicates the functional capability level for this CF application structure.

### **DESCR**

Descriptive comment.

### **DSBLOCK**

The logical block size, which is the unit in which shared message data set space is allocated to individual queues.

### **DSBUFS**

The number of buffers allocated in each queue manager for accessing shared message data sets.

### **DSEXPAND**

Whether the queue manager expands a shared message data set.

### **DSGROUP**

The generic data set name to be used for the group of shared message data sets.

### **OFFLD1SZ**

Offload rule 1: The message size value specifying an integer followed by K, giving the number of kilobytes.

### **OFFLD1TH**

Offload rule 1: The coupling facility structure percentage usage threshold value as an integer.

### **OFFLD2SZ**

Offload rule 2: The message size value specifying an integer followed by K, giving the number of kilobytes.

### **OFFLD2TH**

Offload rule 2: The coupling facility structure percentage usage threshold value as an integer.

### **OFFLD3SZ**

Offload rule 3: The message size value specifying an integer followed by K, giving the number of kilobytes.

### **OFFLD3TH**

Offload rule 3: The coupling facility structure percentage usage threshold value as an integer.

### **OFFLOAD**

If the CFLEVEL is less than 4, the only value you can display is NONE.

If the CFLEVEL is 4, the only value can display is Db2.

If the CFLEVEL is 5, the values displayed are Db2, SMDS, or BOTH. These values depict whether offloaded message data is stored in a group of shared message data sets, or in Db2, or both.

In addition, the offload rules parameter values for OFFLD1SZ, OFFLD1TH, OFFLD2SZ, OFFLD2TH, OFFLD3SZ, and OFFLD3TH are displayed.

## RECAUTO

Indicates whether automatic recovery action is taken when a queue manager detects that the structure is failed, or when a queue manager loses connectivity to the structure and no systems in the SysPlex have connectivity to the Coupling Facility that the structure is allocated in. Values are:

### YES

The structure and associated shared message data sets which also need recovery are automatically recovered.

### NO

The structure is not automatically recovered.

## RECOVER

Indicates whether CF recovery for the application structure is supported. Values are:

### NO

CF application structure recovery is not supported.

### YES


CF application structure recovery is supported.

## DISPLAY CHANNEL (display channel definition)

Use the MQSC command DISPLAY CHANNEL to display a channel definition.

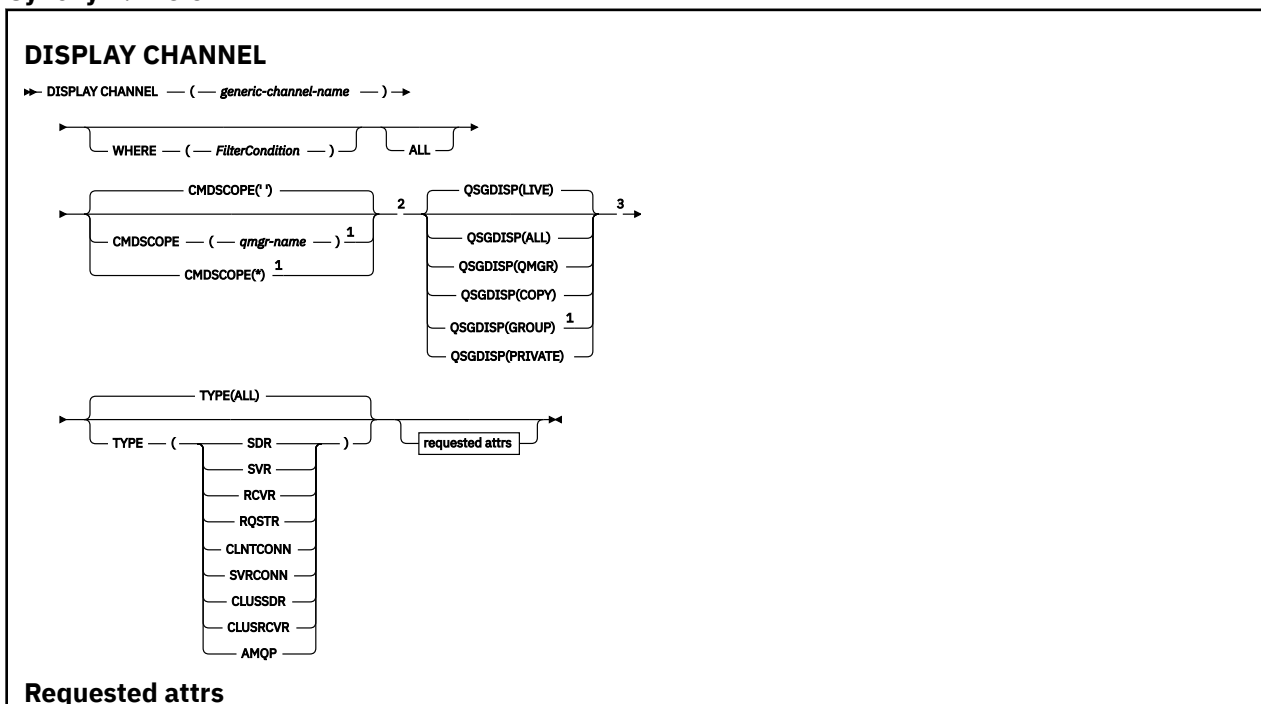
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 663](#)
- [“Parameter descriptions for DISPLAY CHANNEL” on page 663](#)
- [“Requested parameters” on page 666](#)

**Synonym:** DIS CHL



AFFINITY
ALTDATA
ALTTIME
AMQPKA
BATCHHB
BATCHINT
BATCHLIM
BATCHSZ
CERTLABL
CHLTYPE
CLNTWGHT
CLUSNL
CLUSTER
CLWLPRTY
CLWLRANK
CLWLWGHT
COMPHDR
COMPMSG
CONNNAME
CONVERT
DEFCDISP <sup>3</sup>
DEFRECON
DESCR
DISCINT
HBINT
JAASCFG
KAIN
LOCLADDR
LONGRTY
LONGTMR
MAXINST
MAXINSTC
MAXMSGL
MCANAME
MCAATYPE
MCAUSER
MODENAME
MONCHL
MRDATA
MREXIT
MRRTY
MRTMR
MSGDATA
MSGEXIT
NETPRTY
NPMSPEED
PASSWORD
PORT
PROPCTL
PUTAUT <sup>4</sup>
QMNAME
RCVDATA
RCVEXIT
RESETSEQ <sup>5</sup>
SCYDATA
SCYEXIT
SENDATA
SENDEXIT
SEQWRAP
SHARECNV
SHORTRTY
SHORTTMR
SPLPROT <sup>3</sup>
SSLCAUTH
SSLCIPH
SSLKEYP
SSLKEYR
SSLPEER
STATCHL
TPNAME
TPROOT
TRPTYPE
USECLTID
USEDLQ
USERID
XMITQ

Notes:

- <sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>2</sup> Not valid for z/OS client-connection channels.

<sup>3</sup> Valid only on z/OS.

<sup>4</sup> Valid only for RCVR, RQSTR, CLUSRCVR and (for z/OS only) SVRCONN channel types.

<sup>5</sup> Not valid on z/OS.

## Usage notes

You can only display cluster-sender channels if they were created manually. See [Cluster channels](#).

The values shown describe the current definition of the channel. If the channel has been altered since it was started, any currently running instance of the channel object might not have the same values as the current definition.

## Parameter descriptions for DISPLAY CHANNEL

You must specify the name of the channel definition you want to display. It can be a specific channel name or a generic channel name. By using a generic channel name, you can display either:

- All channel definitions
- One or more channel definitions that match the specified name

### **(generic-channel-name)**

The name of the channel definition to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk (\*) matches all channel definitions with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all channel definitions.

## WHERE

Specify a filter condition to display only those channels that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

### **filter-keyword**

Almost any parameter that can be used to display attributes for this DISPLAY command. However, you cannot use the CMDSCOPE, QSGDISP, or MCANAME parameters as filter keywords. You cannot use TYPE (or CHLTYPE) if it is also used to select channels. Channels of a type for which the filter keyword is not a valid attribute are not displayed.

### **operator**

This is used to determine whether a channel satisfies the filter value on the given filter keyword. The operators are:

#### **LT**

Less than

#### **GT**

Greater than

#### **EQ**

Equal to

#### **NE**

Not equal to

#### **LE**

Less than or equal to

#### **GE**

Greater than or equal to

#### **LK**

Matches a generic string that you provide as a *filter-value*

#### **NL**

Does not match a generic string that you provide as a *filter-value*

**CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which contain the specified item.

**EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not contain the specified item.

**CTG**

Contains an item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which match the generic string.

**EXG**

Does not contain any item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not match the generic string.

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value SDR on the TYPE parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.


You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. The value can be explicit or, if it is a character value, it can be explicit or generic. If it is explicit, use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed. If it is generic, use CTG or EXG as the operator. If ABC\* is specified with the operator CTG, all items where one of the attribute values begins with ABC are listed.


**ALL**

Specify ALL to display the results of querying all the parameters. If ALL is specified, any request for a specific parameter is ignored. The result of querying with ALL is to return the results for all of the possible parameters.

This is the default, if you do not specify a generic name and do not request any specific parameters.

 On z/OS this is also the default if you specify a filter condition using the WHERE parameter, but on other platforms, only requested attributes are displayed.

If no parameters are specified (and the ALL parameter is not specified or defaulted), the default is that the channel names only are displayed.

 On z/OS, the CHLTYPE and QSGDISP values are also displayed.

 **CMDSCOPE**

This parameter specifies how the command is executed when the queue manager is a member of a queue sharing group.



..

The command is executed on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command is executed on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

**z/OS QSGDISP**

Specifies the disposition of the objects for which information is to be displayed. Values are:

**LIVE**

This is the default value and displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

**ALL**

Displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP).

If QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

**Note:** In the QSGDISP(LIVE) case, this occurs only where a shared and a non-shared queue have the same name; such a situation should not occur in a well-managed system.

In a shared queue manager environment, use

```
DISPLAY CHANNEL(name) CMDSCOPE(*) QSGDISP(ALL)
```

to list ALL objects matching

```
name
```

in the queue sharing group without duplicating those in the shared repository.

**COPY**

Display information only for objects defined with QSGDISP(COPY).

**GROUP**

Display information only for objects defined with QSGDISP(GROUP). This is allowed only if there is a shared queue manager environment.

**PRIVATE**

Display information only for objects defined with QSGDISP(QMGR) or QSGDISP(COPY). Note that QSGDISP(PRIVATE) displays the same information as QSGDISP(LIVE).

**QMGR**

Display information only for objects defined with QSGDISP(QMGR).

QSGDISP displays one of the following values:

**QMGR**

The object was defined with QSGDISP(QMGR).

**GROUP**

The object was defined with QSGDISP(GROUP).

**COPY**

The object was defined with QSGDISP(COPY).

You cannot use QSGDISP as a filter keyword.

**TYPE**

This is optional. It can be used to restrict the display to channels of one type.

The value is one of the following:

**ALL**

Channels of all types are displayed (this is the default).

**SDR**

Sender channels only are displayed.

**SVR**

Server channels only are displayed.

**RCVR**

Receiver channels only are displayed.

**RQSTR**

Requester channels only are displayed.

**CLNTCONN**

Client-connection channels only are displayed.

**SVRCONN**

Server-connection channels only are displayed.

**CLUSSDR**

Cluster-sender channels only are displayed. ).

**CLUSRCVR**

Cluster-receiver channels only are displayed. ).

**AMQP**

AMQP channels only are displayed.

CHLTYPE( *type* ) can be used as a synonym for this parameter. ,

**Requested parameters**

Specify one or more DISPLAY CHANNEL parameters that define the data to be displayed. You can specify the parameters in any order, but do not specify the same parameter more than once.

Some parameters are relevant only for channels of a particular type or types. Attributes that are not relevant for a particular type of channel cause no output, nor is an error raised. The following table shows the parameters that are relevant for each type of channel. There is a description of each parameter after the table. Parameters are optional unless the description states that they are required.

Parameter	SDR	SVR	RCVR	RQSTR	CLNT- CONN	SVR- CONN	CLUS- SDR	CLUS- RCVR	AM QP
AFFINIT Y					✓				
ALTDATE	✓	✓	✓	✓	✓	✓	✓	✓	✓
ALTTIME	✓	✓	✓	✓	✓	✓	✓	✓	✓
AMQPKA									✓

Table 167. Parameters that result in data being returned from the DISPLAY CHANNEL command (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNT- CONN	SVR- CONN	CLUS- SDR	CLUS- RCVR	AM QP
AUTOSTART		✓	✓	✓		✓			
BATCHHB	✓	✓					✓	✓	
BATCHINT	✓	✓					✓	✓	
BATCHLIM	✓	✓					✓	✓	
BATCHSZ	✓	✓	✓	✓			✓	✓	
CERTLABL	✓	✓	✓	✓	✓	✓	✓	✓	✓
channel-name	✓	✓	✓	✓	✓	✓	✓	✓	✓
CHLTYP E	✓	✓	✓	✓	✓	✓	✓	✓	✓
CLNTWGHT					✓				
CLUSNL							✓	✓	
CLUSTER							✓	✓	
CLWLPRTY							✓	✓	
CLWLRA NK							✓	✓	
CLWLWGHT							✓	✓	
COMPHDR	✓	✓	✓	✓	✓	✓	✓	✓	
COMPM SG	✓	✓	✓	✓	✓	✓	✓	✓	
CONNAME	✓	✓		✓	✓		✓	✓	
CONVERT	✓	✓					✓	✓	
DEFCDISP	✓	✓	✓	✓		✓			
DEFREC ON					✓				

Table 167. Parameters that result in data being returned from the DISPLAY CHANNEL command (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNT- CONN	SVR- CONN	CLUS- SDR	CLUS- RCVR	AM QP
<u>DESCR</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>DISCINT</u>	✓	✓				✓	✓	✓	
<u>HBINT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>KAINT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>LOCLAD DR</u>	✓	✓		✓	✓		✓	✓	✓
<u>LONGRT Y</u>	✓	✓					✓	✓	
<u>LONGTM R</u>	✓	✓					✓	✓	
<u>MAXINS T</u>						✓			✓
<u>MAXINS TC</u>						✓			
<u>MAXMS GL</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>MCANA ME</u>	✓	✓		✓			✓	✓	
<u>MCATYP E</u>	✓	✓		✓			✓	✓	
<u>MCAUSE R</u>	✓	✓	✓	✓		✓	✓	✓	✓
<u>MODEN AME</u>	✓	✓		✓	✓		✓	✓	
<u>MONCHL</u>	✓	✓	✓	✓		✓	✓	✓	
<u>MRDATA</u>			✓	✓				✓	
<u>MREXIT</u>			✓	✓				✓	
<u>MRRTY</u>			✓	✓				✓	
<u>MRTMR</u>			✓	✓				✓	
<u>MSGDAT A</u>	✓	✓	✓	✓			✓	✓	
<u>MSGEXI T</u>	✓	✓	✓	✓			✓	✓	
<u>NETPRT Y</u>								✓	

Table 167. Parameters that result in data being returned from the DISPLAY CHANNEL command (continued)


Parameter	SDR	SVR	RCVR	RQSTR	CLNT- CONN	SVR- CONN	CLUS- SDR	CLUS- RCVR	AM QP
<u>NPMSPE</u> <u>ED</u>	✓	✓	✓	✓			✓	✓	
<u>PASSWO</u> <u>RD</u>	✓	✓		✓	✓		✓		
<u>PORT</u>									✓
<u>PROPCT</u> <u>L</u>	✓	✓					✓	✓	
<u>PUTAUT</u>			✓	✓		✓ "1" on page 670		✓	
<u>QMNAM</u> <u>E</u>					✓				
<u>RESETS</u> <u>EQ</u>	✓	✓	✓	✓			✓	✓	
<u>RCVDAT</u> <u>A</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>RCVEXIT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SCYDAT</u> <u>A</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SCYEXIT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SENDDA</u> <u>TA</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SENDEX</u> <u>IT</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SEQWRA</u> <u>P</u>	✓	✓	✓	✓			✓	✓	
<u>SHAREC</u> <u>NV</u>						✓			
<u>SHORTR</u> <u>TY</u>	✓	✓					✓	✓	
<u>SHORTT</u> <u>MR</u>	✓	✓					✓	✓	
 <b>SPLPRO</b> <b>T</b>	✓	✓	✓	✓					
<u>SSLCAU</u> <u>TH</u>		✓	✓	✓		✓		✓	✓
<u>SSLCIPH</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>SSLPEER</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 167. Parameters that result in data being returned from the DISPLAY CHANNEL command (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNT- CONN	SVR- CONN	CLUS- SDR	CLUS- RCVR	AM QP
STATCHL	✓	✓	✓	✓			✓	✓	
TPNAME	✓	✓		✓	✓	✓	✓	✓	
TPROOT									✓
TRPTYPE	✓	✓	✓	✓	✓	✓	✓	✓	
USECLTD									✓
USEDLQ	✓	✓	✓	✓			✓	✓	
USERID	✓	✓		✓	✓		✓		
XMITQ	✓	✓							

**Note:**

1. PUTAUT is valid for a channel type of SVRCONN on z/OS only.

**AFFINITY**

The channel affinity attribute.

**PREFERRED**

Subsequent connections in a process attempt to use the same channel definition as the first connection.

**NONE**

All connections in a process select an applicable definition based on the weighting with any applicable CLNTWGHT(0) definitions selected first in alphabetical order.

**ALTDATE**

The date on which the definition was last altered, in the form yyyy-mm-dd.

**ALLTIME**

The time at which the definition was last altered, in the form hh.mm.ss.

**AMQPKA**

The keep alive time for an AMQP channel in milliseconds.

**AUTOSTART**

Whether an LU 6.2 responder process should be started for the channel.

**BATCHHB**

The batch heartbeating value being used.

**BATCHINT**

Minimum batch duration.

**BATCHLIM**

Batch data limit.

The limit of the amount of data that can be sent through a channel.

**BATCHSZ**

Batch size.


**CERTLABL**

Certificate label.

**CHLTYPE**

Channel type.

The channel type is always displayed if you specify a generic channel name and do not request any other parameters. On z/OS, the channel type is always displayed.

 On Multiplatforms, TYPE can be used as a synonym for this parameter.

**CLNTWGHT**

The client channel weighting.

The special value 0 indicates that no random load balancing is performed and applicable definitions are selected in alphabetical order. If random load balancing is performed the value is in the range 1 - 99 where 1 is the lowest weighting and 99 is the highest.

**CLUSTER**

The name of the cluster to which the channel belongs.

**CLUSNL**

The name of the namelist that specifies the list of clusters to which the channel belongs.

**CLWLPRTY**

The priority of the channel for the purposes of cluster workload distribution.

**CLWLRANK**

The rank of the channel for the purposes of cluster workload distribution.

**CLWLWGHT**

The weighting of the channel for the purposes of cluster workload distribution.

**COMPHDR**

The list of header data compression techniques supported by the channel. For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference.

**COMPMSG**

The list of message data compression techniques supported by the channel. For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference.

**CONNAME**

Connection name.

**CONVERT**

Whether sender should convert application message data.

**DEFCDISP**

Specifies the default channel disposition of the channels for which information is to be returned. If this keyword is not present, channels of all default channel dispositions are eligible.

**ALL**

Channels of all default channel dispositions are displayed.

This is the default setting.

**PRIVATE**

Only channels where the default channel disposition is PRIVATE are displayed.

**SHARED**

Only channels where the default channel disposition is FIXSHARED or SHARED are displayed.

**Note:** This does not apply to client-connection channel types on z/OS.

**DESCR**

Default client reconnection option.

**DESCR**

Description.

**DISCINT**

Disconnection interval.

**HBINT**

Heartbeat interval.

**KAINT**

KeepAlive timing for the channel.

**LOCLADDR**

Local communications address for the channel.

**LONGRTY**

Long retry count.

**LONGTMR**

Long retry timer.

**MAXINST( *integer* )**

The maximum number of instances of a server-connection channel that are permitted to run simultaneously.

**MAXINSTC( *integer* )**

The maximum number of instances of a server-connection channel, started from a single client, that are permitted to run simultaneously.

**Note:** In this context, connections originating from the same remote network address are regarded as coming from the same client.

**MAXMSGL**

Maximum message length for channel.

**MCANAME**

Message channel agent name.

You cannot use MCANAME as a filter keyword.

**MCATYPE**

Whether message channel agent runs as a separate process or a separate thread.

**MCAUSER**

Message channel agent user identifier.

**MODENAME**

LU 6.2 mode name.

**MONCHL**

Online monitoring data collection.

**MRDATA**

Channel message-retry exit user data.

**MREXIT**

Channel message-retry exit name.

**MRRTY**

Channel message-retry count.

**MRTMR**

Channel message-retry time.

**MSGDATA**

Channel message exit user data.

**MSGEXIT**

Channel message exit names.

**NETPRTY**


The priority for the network connection.

**NPMSPEED**

Nonpersistent message speed.



## PASSWORD

Password for initiating LU 6.2 session. If nonblank, this is displayed as asterisks  on all platforms except z/OS.

## PORT

The port number used to connect an AMQP channel.

## PROPCTL

Message property control.

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor).

This parameter is applicable to Sender, Server, Cluster Sender, and Cluster Receiver channels.

This parameter is optional.

Permitted values are:

### COMPAT

This is the default value.

Message properties	Result
The message contains a property with a prefix of <b>mcd.</b> , <b>jms.</b> , <b>usr.</b> or <b>mqext.</b>	All optional message properties (where the <b>Support</b> value is MQPD_SUPPORT_OPTIONAL), except those in the message descriptor or extension, are placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.
The message does not contain a property with a prefix of <b>mcd.</b> , <b>jms.</b> , <b>usr.</b> or <b>mqext.</b>	All message properties, except those in the message descriptor or extension, are removed from the message before the message is sent to the remote queue manager.
The message contains a property where the <b>Support</b> field of the property descriptor is not set to MQPD_SUPPORT_OPTIONAL	The message is rejected with reason MQRC_UNSUPPORTED_PROPERTY and treated in accordance with its report options.
The message contains one or more properties where the <b>Support</b> field of the property descriptor is set to MQPD_SUPPORT_OPTIONAL but other fields of the property descriptor are set to non-default values	The properties with non-default values are removed from the message before the message is sent to the remote queue manager.
The MQRFH2 folder that would contain the message property needs to be assigned with the <i>content='properties'</i> attribute	The properties are removed to prevent MQRFH2 headers with unsupported syntax flowing to a V6 or prior queue manager.

### NONE

All properties of the message, except those in the message descriptor or extension, are removed from the message before the message is sent to the remote queue manager.

If the message contains a property where the **Support** field of the property descriptor is not set to MQPD\_SUPPORT\_OPTIONAL then the message is rejected with reason MQRC\_UNSUPPORTED\_PROPERTY and treated in accordance with its report options.

### ALL

All properties of the message are included with the message when it is sent to the remote queue manager. The properties, except those in the message descriptor (or extension), are placed in one or more MQRFH2 headers in the message data.

**PUTAUT**

Put authority.

**QMNAME**

Queue manager name.

**RESETSEQ**

Pending reset sequence number.

This is the sequence number from an outstanding request and it indicates a user RESET CHANNEL command request is outstanding.

A value of zero indicates that there is no outstanding RESET CHANNEL. The value can be in the range 1 - 999999999.

This parameter is not applicable on z/OS.

**RCVDATA**

Channel receive exit user data.

**RCVEXIT**

Channel receive exit names.

**SCYDATA**

Channel security exit user data.

**SCYEXIT**

Channel security exit names.

**SENDDATA**

Channel send exit user data.

**SENDEXIT**

Channel send exit names.

**SEQWRAP**

Sequence number wrap value.

**SHARECNV**

Sharing conversations value.

**SHORTRTY**

Specifies the maximum number of times that the channel is to try allocating a session to its partner.

**SHORTTMR**

Short retry timer.

 **SPLPROT**

SPLPROT (Security Policy Protection) specifies how a server-to-server Message Channel Agent should deal with message protection when AMS is active and an applicable policy exists.

**SSLCAUTH**

Whether TLS client authentication is required.

**SSLCIPH**

Cipher specification for the TLS connection.

**SSLPEER**

Filter for the Distinguished Name from the certificate of the peer queue manager or client at the other end of the channel.

**STATCHL**

Statistics data collection.

**TPNAME**

LU 6.2 transaction program name.

**TPROOT**

The topic root for an AMQP channel.

**TRPTYPE**

Transport type.

## USECLTID

Specifies that the client ID should be used for authorization checks for an AMQP channel, instead of the MCAUSER attribute value.

## USEDLQ

Determines whether the dead-letter queue is used when messages cannot be delivered by channels.

## USERID

User identifier for initiating LU 6.2 session.

## XMITQ

Transmission queue name.

For more details of these parameters, see [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466.

## **ALW** DISPLAY CHANNEL (display channel definition) MQTT

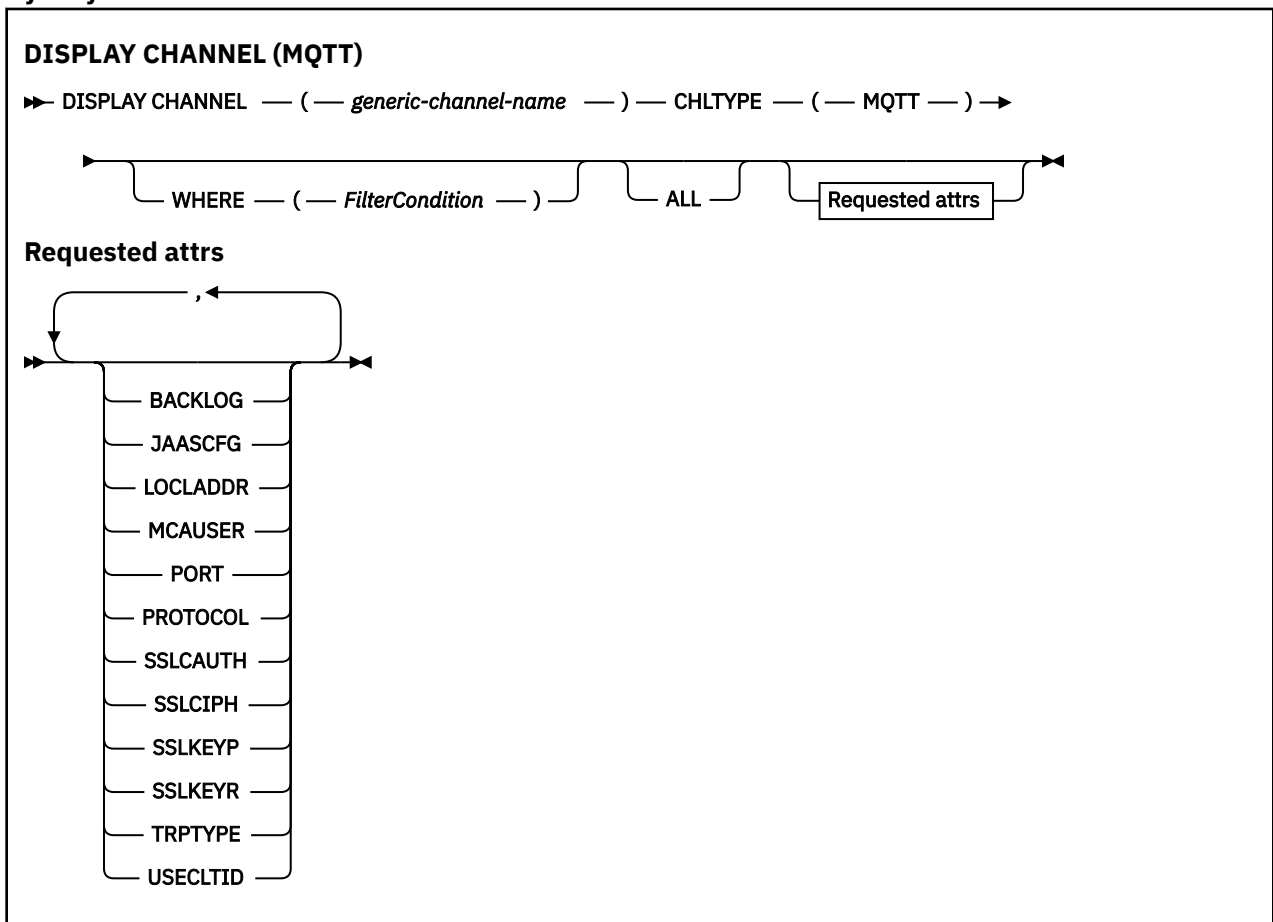
Use the MQSC command DISPLAY CHANNEL (MQTT) to display an MQ Telemetry channel definition.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY CHANNEL \(MQTT\)”](#) on page 676
- [“Requested parameters”](#) on page 677

**Synonym:** DIS CHL



DISPLAY CHANNEL (MQTT) command is only valid for MQ Telemetry channels.

## Parameter descriptions for DISPLAY CHANNEL (MQTT)

You must specify the name of the channel definition you want to display. This can be a specific channel name or a generic channel name. By using a generic channel name, you can display either:

- All channel definitions
- One or more channel definitions that match the specified name

### **(*generic-channel-name*)**

The name of the channel definition to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk (\*) matches all channel definitions with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all channel definitions.

### **CHLTYPE( *type* )**

The value is always MQTT.

TYPE can be used as a synonym for this parameter.

### **WHERE**

Specify a filter condition to display only those channels that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Almost any parameter that can be used to display attributes for this DISPLAY command. However, you cannot use the CMDSCOPE, QSGDISP, or MCANAME parameters as filter keywords. You cannot use TYPE (or CHLTYPE) if it is also used to select channels. Channels of a type for which the filter keyword is not a valid attribute are not displayed.

#### **operator**

This is used to determine whether a channel satisfies the filter value on the given filter keyword. The operators are:

##### **LT**

Less than

##### **GT**

Greater than

##### **EQ**

Equal to

##### **NE**

Not equal to

##### **LE**

Less than or equal to

##### **GE**

Greater than or equal to

##### **LK**

Matches a generic string that you provide as a *filter-value*

##### **NL**

Does not match a generic string that you provide as a *filter-value*

##### **CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which contain the specified item.

##### **EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not contain the specified item.

**CTG**

Contains an item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which match the generic string.

**EXG**

Does not contain any item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not match the generic string.

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value SDR on the TYPE parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. The value can be explicit or, if it is a character value, it can be explicit or generic. If it is explicit, use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed. If it is generic, use CTG or EXG as the operator. If ABC\* is specified with the operator CTG, all items where one of the attribute values begins with ABC are listed.

**ALL**

Specify ALL to display the results of querying all the parameters. If ALL is specified, any request for a specific parameter is ignored. The result of querying with ALL is to return the results for all of the possible parameters.

This is the default, if you do not specify a generic name and do not request any specific parameters.

If no parameters are specified (and the ALL parameter is not specified or defaulted), the default is that the channel names only are displayed.

**Requested parameters**

Specify one or more DISPLAY CHANNEL parameters that define the data to be displayed. You can specify the parameters in any order, but do not specify the same parameter more than once.

Some parameters are relevant only for channels of a particular type or types. Attributes that are not relevant for a particular type of channel cause no output, nor is an error raised. The following table shows the parameters that are relevant for each type of channel. There is a description of each parameter after the table. Parameters are optional unless the description states that they are required.

**BACKLOG**

The number of outstanding connection requests that the telemetry channel can support at any one time. When the backlog limit is reached, any further clients trying to connect will be refused connection until the current backlog is processed. The value is in the range 0 - 999999999. The default value is 4096.

**CHLTYPE**

Channel type.

There is only one valid value for this parameter: MQTT.

**JAASCFG**

The name of a stanza in the JAAS configuration file.

**LOCLADDR**

The local communications address for the channel.

**MCAUSER**

The message channel agent user identifier.

**PORT**

The port number on which the telemetry (MQXR) service accepts client connections.

**PROTOCOL**

The communication protocol supported by the channel.

**SSLCAUTH**

Defines whether IBM MQ requires a certificate from the TLS client.

**SSLCIPH**

When **SSLCIPH** is used with a telemetry channel, it means TLS Cipher Suite.

**SSLKEYP**

The password for the key repository. If no passphrase is entered, you must use unencrypted connections.

**SSLKEYR**

The name of the TLS key repository. For full details, see the SSLKEYR parameter of the [ALTER QMGR](#) command.

**TRPTYPE**

The transmission protocol to be used. For a Telemetry channel, this is always TCP (that is, the TCP/IP protocol).

**USECLTID**

Indicates whether you want to use the MQTT client ID for the connection as the IBM MQ user ID for that connection.

For more details of these parameters, see [“DEFINE CHANNEL \(define a new channel\) for MQTT”](#) on page 523.

**z/OS DISPLAY CHINIT (display channel initiator information) on z/OS**

Use the MQSC command DISPLAY CHINIT to display information about the channel initiator. The command server must be running.

**Using MQSC commands on z/OS**

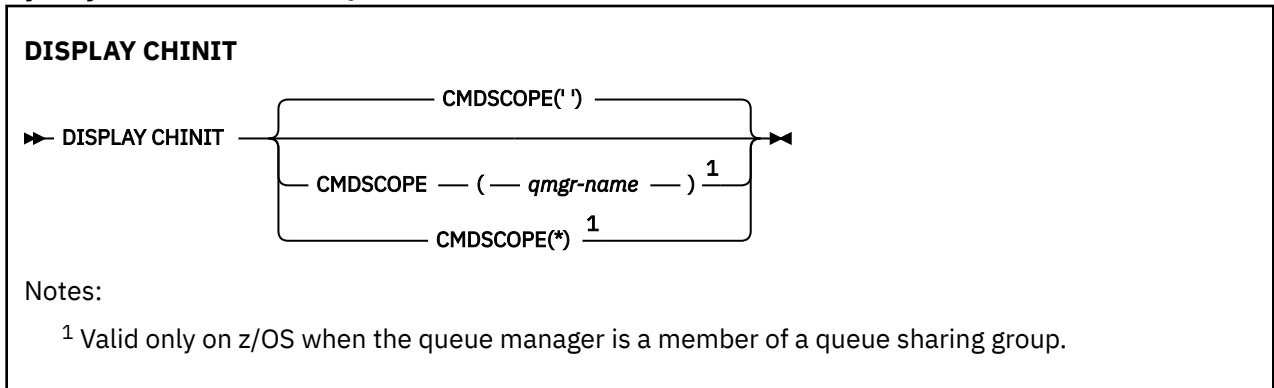
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)

- “Usage notes for DISPLAY CHINIT” on page 679
- “Parameter descriptions for DISPLAY CHINIT” on page 679

**Synonym:** DIS CHI or DIS DQM



## Usage notes for DISPLAY CHINIT

1. The response to this command is a series of messages showing the current status of the channel initiator. This includes the following:
  - Whether the channel initiator is running or not
  - Which listeners are started, and information about them.
  - How many dispatchers are started, and how many were requested
  - How many adapter subtasks are started, and how many were requested
  - How many TLS subtasks are started, and how many were requested
  - The TCP system name
  - How many channel connections are current, and whether they are active, stopped, or retrying
  - The maximum number of current connections

## Parameter descriptions for DISPLAY CHINIT

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## DISPLAY CHLAUTH (display channel authentication record)

Use the MQSC command DISPLAY CHLAUTH to display the attributes of a channel authentication record.

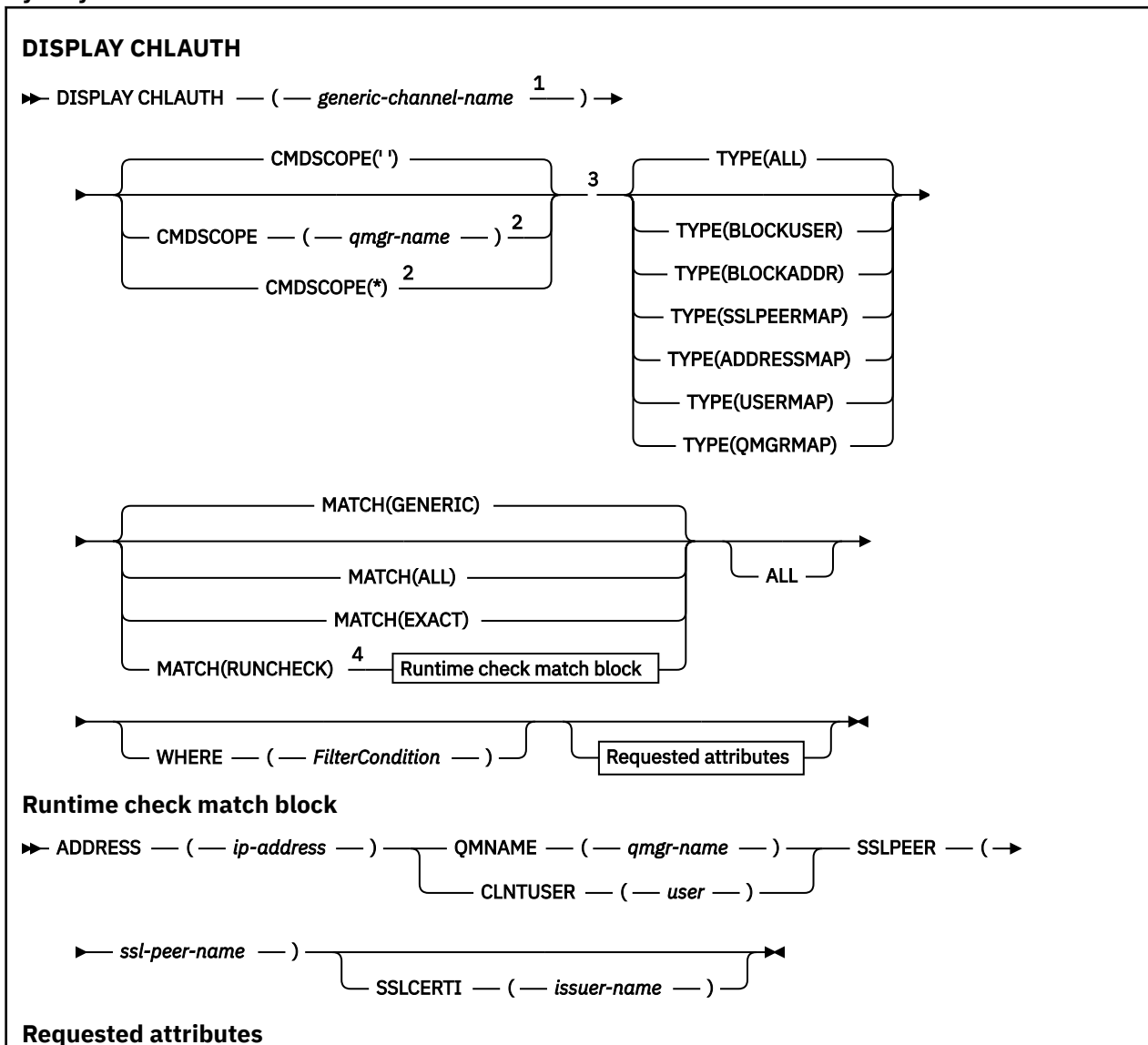
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

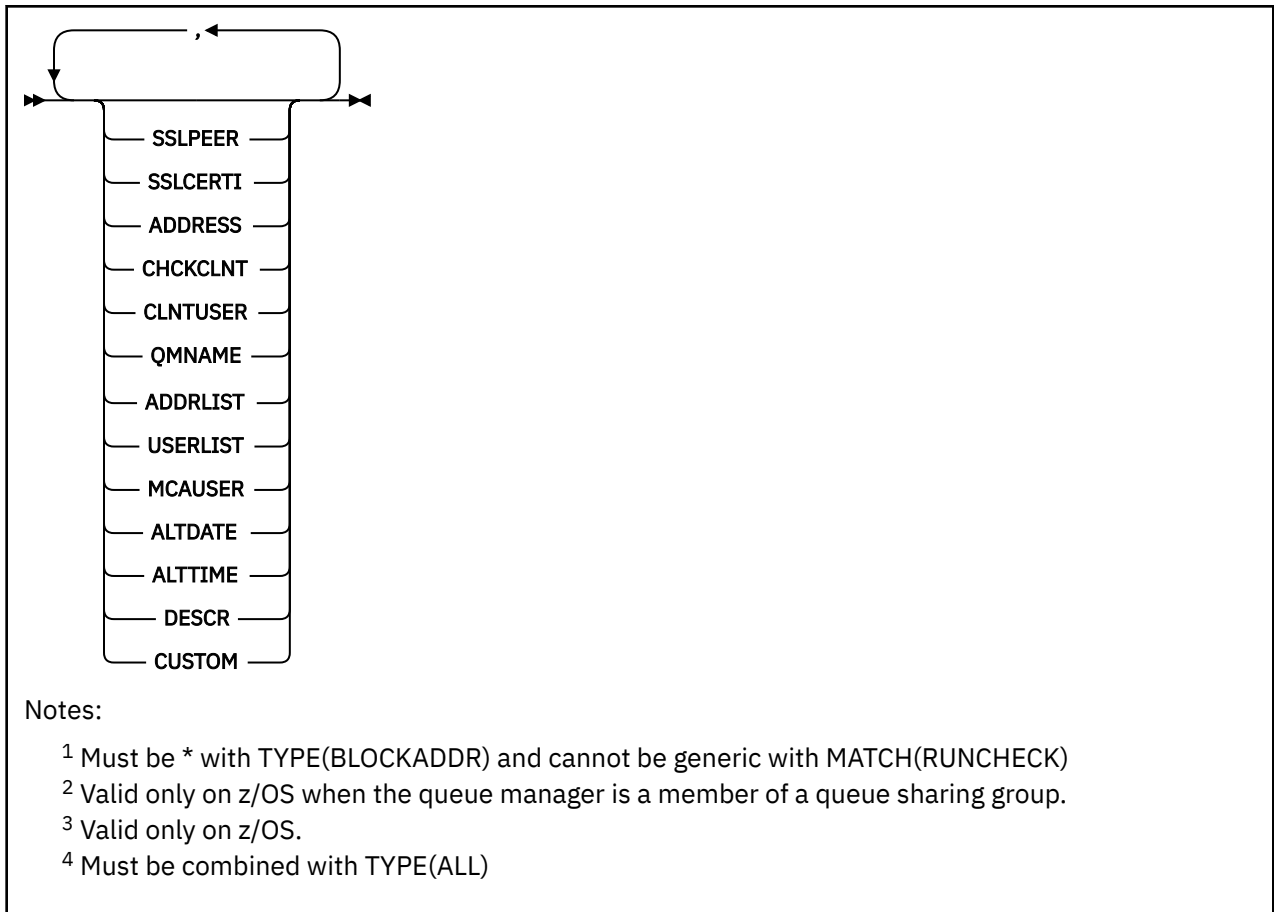
**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [Parameters](#)

**Synonym:** DIS CHLAUTH







## Parameters

### generic-channel-name

The name of the channel or set of channels to display. You can use the asterisk (\*) as a wildcard to specify a set of channels. When an asterisk is used on z/OS, single quotes must be used around the whole value. When **MATCH** is RUNCHECK this parameter must not be generic.

### ADDRESS

The IP address to be matched.

This parameter is valid only when **MATCH** is RUNCHECK, must not be generic and must not be a host name.

### ALL

Specify this parameter to display all attributes. If this keyword is specified, any attributes that are requested specifically have no effect; all attributes are still displayed.

This is the default behavior if you do not specify a generic name and do not request any specific attributes.

### CLNTUSER

The client asserted user ID to be mapped to a new user ID, allowed through unchanged, or blocked.

This can be the user ID flowed from the client indicating the user ID the client side process is running under, or the user ID presented by the client on an MQCONN call using MQCSP.

This parameter is valid only with TYPE(USERMAP) and when **Match** is RUNCHECK.

The maximum length of the string is MQ\_CLIENT\_USER\_ID\_LENGTH.

This parameter applies to z/OS only and specifies how the command is run when the queue manager is a member of a queue sharing group.

''

The command is run on the queue manager on which it was entered. This is the default value.

#### **qmgr-name**

The command is run on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command is run on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect is the same as entering the command on every queue manager in the queue sharing group.

### **MATCH**

Indicates the type of matching to be applied.

#### **RUNCHECK**

Returns the record that is matched by a specific inbound channel at run time if it connects to this queue manager. The specific inbound channel is described by providing values that are not generic:

- Channel name.
- **ADDRESS** attribute containing an IP address, that is then reverse looked up as part of running the command to discover the host name, if the queue manager is configured with **REVDNS (ENABLED)**.
- **SSLCERTI** attribute, only if the inbound channel uses TLS.
- **SSLPEER** attribute, only if the inbound channel uses TLS.
- **QMNAME** or **CLNTUSER** attribute, depending on whether the inbound channel is a client or queue manager channel.

If the record discovered has **WARN** set to YES, a second record might also be displayed to show the actual record the channel will use at run time. This parameter must be combined with **TYPE (ALL)**.

#### **EXACT**

Return only those records which exactly match the channel profile name supplied. If there are no asterisks in the channel profile name, this option returns the same output as **MATCH (GENERIC)**.

#### **GENERIC**

Any asterisks in the channel profile name are treated as wildcards. If there are no asterisks in the channel profile name, this returns the same output as **MATCH (EXACT)**. For example, a profile of **ABC\*** could result in records for **ABC**, **ABC\***, and **ABCD** being returned.

#### **ALL**

Return all possible records that match the channel profile name supplied. If the channel name is generic in this case, all records that match the channel name are returned even if more specific matches exist. For example, a profile of **SYSTEM.\*.SVRCONN** could result in records for **SYSTEM.\***, **SYSTEM.DEF.\***, **SYSTEM.DEF.SVRCONN**, and **SYSTEM.ADMIN.SVRCONN** being returned.

### **QMNAME**

The name of the remote partner queue manager to be matched

This parameter is valid only when **MATCH** is **RUNCHECK** and must not be generic.

### **SSLCERTI**

The Certificate issuer Distinguished Name of the certificate to be matched.

The **SSLCERTI** field, if not blank, is matched in addition to the **SSLPEER** value.

This parameter is valid only when **MATCH** is RUNCHECK and must not be generic.

### **SSLPEER**

The Subject Distinguished Name of the certificate to be matched.

The **SSLPEER** value is specified in the standard form used to specify a Distinguished Name.

This parameter is valid only when **MATCH** is RUNCHECK and must not be generic.

### **TYPE**

The type of Channel Authentication Record for which to display details. Possible values are:

- ALL
- BLOCKUSER
- BLOCKADDR
- SSLPEERMAP
- ADDRESSMAP
- USERMAP
- QMGRMAP

### **WHERE**

Specify a filter condition to display only those channel authentication records that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Any parameter that can be used to display attributes for this DISPLAY command.

#### **operator**

This is used to determine whether a channel authentication record satisfies the filter value on the given filter keyword. The operators are as follows:

#### **LT**

Less than

#### **GT**

Greater than

#### **EQ**

Equal to

#### **NE**

Not equal to

#### **LE**

Less than or equal to

#### **GE**

Greater than or equal to

#### **LK**

Matches a generic string that you provide as a *filter-value*

#### **NL**

Does not match a generic string that you provide as a *filter-value*

#### **CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which contain the specified item.

#### **EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not contain the specified item.

**CTG**

Contains an item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which match the generic string.

**EXG**

Does not contain any item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not match the generic string.

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, the value can be either explicit or generic:


- An explicit value, that is a valid value for the attribute being tested.

You can use any of the operators except LK and NL. However, if the value is one from a possible set of values returnable on a parameter (for example, the value ALL on the MATCH parameter), you can only use EQ or NE.

- A generic value. This is a character string with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

You can only use operators LK or NL for generic values.

- An item in a list of values. The value can be explicit or, if it is a character value, it can be explicit or generic. If it is explicit, use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed. If it is generic, use CTG or EXG as the operator. If ABC\* is specified with the operator CTG, all items where one of the attribute values begins with ABC are listed.

**Note:**  On z/OS there is a 256 character limit for the filter-value of the MQSC **WHERE** clause. This limit is not in place for other platforms.

**Requested parameters**

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

**TYPE**

The type of channel authentication record

**SSLPEER**

The Distinguished Name of the certificate.

**ADDRESS**

The IP address

**CHCKCLNT**

Whether a user ID and password are to be supplied by connections which match this rule.

**CLNTUSER**

The client asserted user ID

**QMNAME**

The name of the remote partner queue manager

**MCAUSER**

The user identifier to be used when the inbound connection matches the TLS DN, IP address, client asserted user ID or remote queue manager name supplied.

**ADDRLIST**

A list of IP address patterns which are banned from connecting into this queue manager on any channel.

## USERLIST

A list of user IDs which are banned from use of this channel or set of channels.

## ALTDATE

The date on which the channel authentication record was last altered, in the format *yyyy-mm-dd*.

## ALTIME

The time on which the channel authentication record was last altered, in the form *hh.mm.ss*.

## DESCR

Descriptive information about the channel authentication record.

## SSLCERTI

The Certificate issuer Distinguished Name of the certificate to be matched.

## CUSTOM

Reserved for future use.

## Related concepts

[Channel authentication records](#)

## Related reference

“Generic IP addresses for channel authentication records” on page 933


In the various commands that create and display channel authentication records, you can specify certain parameters as either a single IP address or a pattern to match a set of IP addresses.

## DISPLAY CHSTATUS (display channel status)

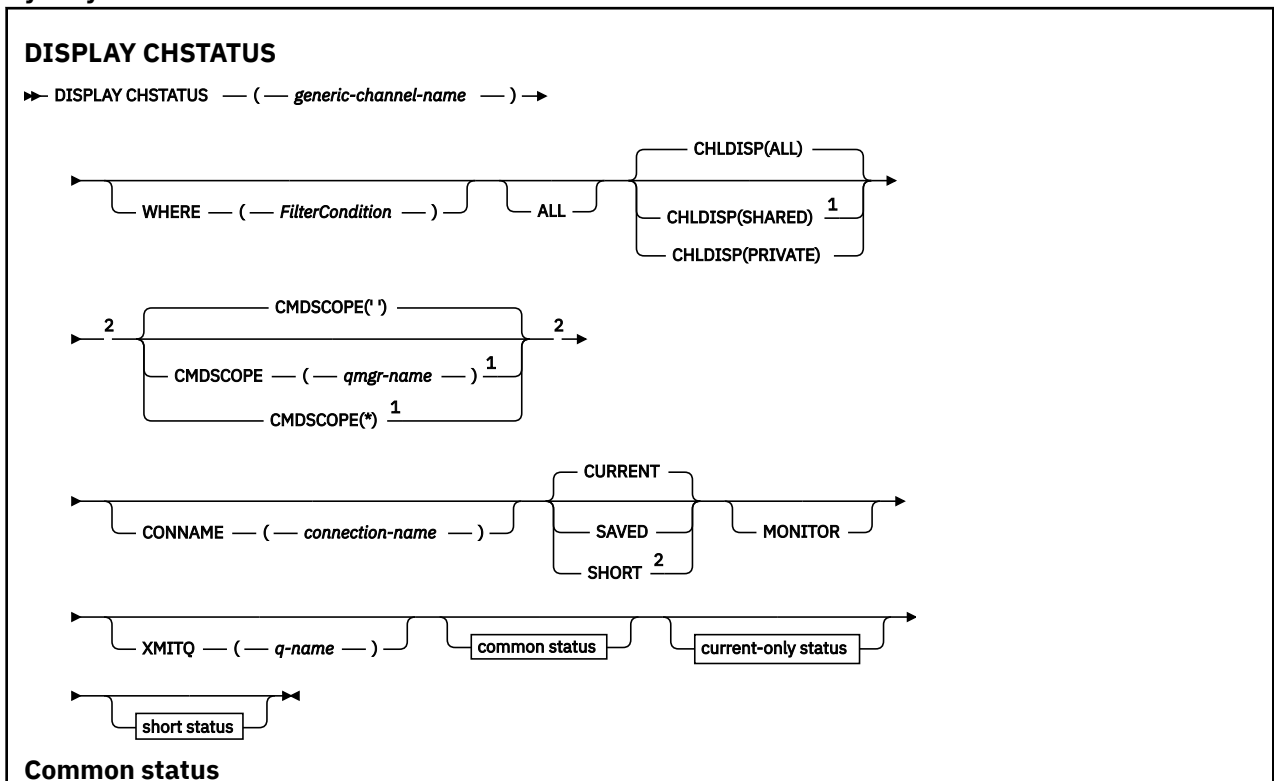
Use the MQSC command **DISPLAY CHSTATUS** to display the status of one or more channels.

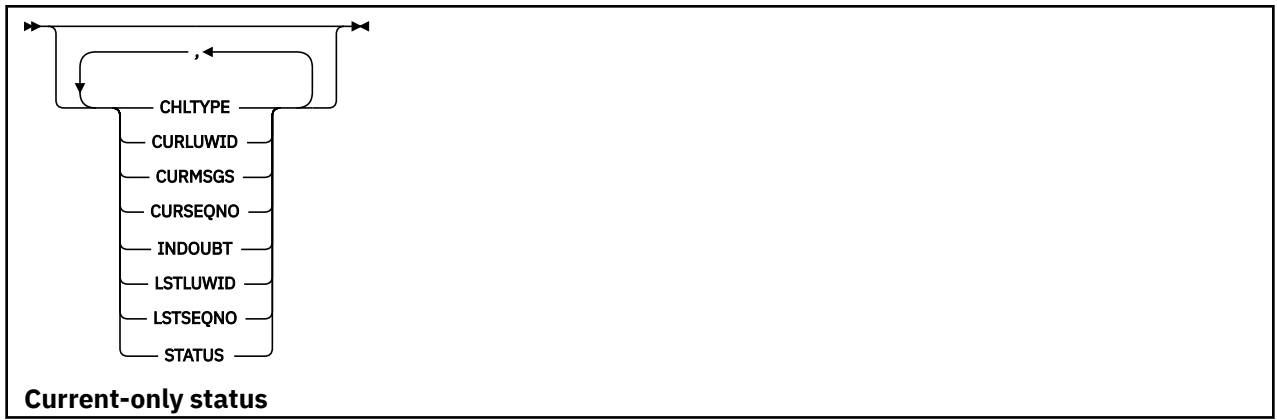
## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

**Synonym:** DIS CHS







- AMQPKA
- BATCHES
- BATCHSZ
- BUFSRCVD
- BUFSSENT
- BYTSRCVD
- BYTSSENT
- CHSTADA
- CHSTATI
- COMPHDR
- COMPMSG
- COMPRATE <sup>3</sup>
- COMPTIME <sup>3</sup>
- CURSHCNV
- EXITTIME <sup>3</sup>
- HBINT
- JOBNAME <sup>4</sup>
- KAJNT <sup>2</sup>
- LOCLADDR
- LONGRTS
- LSTMSGDA
- LSTMSGTI
- MAXSHCNV
- MAXMSGL <sup>2</sup>
- MCASTAT <sup>4</sup>
- MCAUSER
- MONCHL <sup>3</sup>
- MSGS
- NETTIME <sup>3</sup>
- NPMSPEED
- QMNAME <sup>2</sup>
- RAPPLTAG
- RPRODUCT
- RQMNAME
- RVERSION
- SECPROT
- SHORTRTS
- SSLCERTI
- SSLCIPH
- SSLCERTU <sup>2</sup>
- SSLKEYDA
- SSLKEYTI
- SSLPEER
- SSLRKEYS
- STATCHL <sup>2</sup>
- STOPREQ
- SUBSTATE
- XBATCHSZ <sup>3</sup>
- XQMSGSA <sup>3</sup>
- XQTIME <sup>3</sup>

**Short status**



Notes:

- <sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>2</sup> Valid only on z/OS.
- <sup>3</sup> Also displayed by selection of the MONITOR parameter.
- <sup>4</sup> Ignored if specified on z/OS.

## Usage notes for DISPLAY CHSTATUS on z/OS



1. The command fails if the channel initiator has not been started.
2. The command server must be running.
3. If you want to see the overall status of the channel (that is, the status of the queue sharing group) use the command **DISPLAY CHSTATUS SHORT**, which obtains the status information of the channel from Db2.
4. If any numeric parameter exceeds 999,999,999, it is displayed as 999999999.
5. The status information that is returned for various combinations of **CHLDISP**, **CMDSCOPE**, and status type are summarized in [Table 169 on page 688](#), [Table 170 on page 688](#), and [Table 171 on page 689](#).

*Table 169. CHLDISP and CMDSCOPE for DISPLAY CHSTATUS CURRENT*

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)	CMDSCOPE(*)
PRIVATE	Common and current-only status for current private channels on the local queue manager	Common and current-only status for current private channels on the named queue manager	Common and current-only status for current private channels on all queue managers
SHARED	Common and current-only status for current shared channels on the local queue manager	Common and current-only status for current shared channels on the named queue manager	Common and current-only status for current shared channels on all queue managers
ALL	Common and current-only status for current private and shared channels on the local queue manager	Common and current-only status for current private and shared channels on the named queue manager	Common and current-only status for current private and shared channels on all active queue managers

*Table 170. CHLDISP and CMDSCOPE for DISPLAY CHSTATUS SHORT*

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)	CMDSCOPE(*)
PRIVATE	STATUS and short status for current private channels on the local queue manager	STATUS and short status for current private channels on the named queue manager	STATUS and short status for current private channels on all active queue managers
SHARED	STATUS and short status for current shared channels on all active queue managers in the queue sharing group	Not permitted	Not permitted



Table 170. CHLDISP and CMDSCOPE for DISPLAY CHSTATUS SHORT (continued)			
CHLDISP	CMDScope() or CMDScope (local-qmgr)	CMDScope (qmgr-name)	CMDScope(*)
ALL	STATUS and short status for current private channels on the local queue manager and current shared channels in the queue sharing group ( "5.a" on page 689 )	STATUS and short status for current private channels on the named queue manager	STATUS and short status for current private, and shared, channels on all active queue managers in the queue sharing group ( "5.a" on page 689 )

**Note:**

- a. In this case you get two separate sets of responses to the command on the queue manager where it was entered; one for PRIVATE and one for SHARED.

Table 171. CHLDISP and CMDSCOPE for DISPLAY CHSTATUS SAVED			
CHLDISP	CMDScope() or CMDScope (local-qmgr)	CMDScope (qmgr-name)	CMDScope(*)
PRIVATE	Common status for saved private channels on the local queue manager	Common status for saved private channels on the named queue manager	Common status for saved private channels on all active queue managers
SHARED	Common status for saved shared channels on all active queue managers in the queue sharing group	Not permitted	Not permitted
ALL	Common status for saved private channels on the local queue manager and saved shared channels in the queue sharing group	Common status for saved private channels on the named queue manager	Common status for saved private, and shared, channels on all active queue managers in the queue sharing group

### Parameter descriptions for DISPLAY CHSTATUS on all platforms

You must specify the name of the channel for which you want to display status information. This can be a specific channel name or a generic channel name. By using a generic channel name, you can display either the status information for all channels, or status information for one or more channels that match the specified name.

You can also specify whether you want the current status data (of current channels only), or the saved status data of all channels.

Status for all channels that meet the selection criteria is displayed, whether the channels were defined manually or automatically.


The classes of data available for channel status are **saved** and **current**, and (on z/OS only) **short**.

The status fields available for saved data are a subset of the fields available for current data and are called **common** status fields. Note that although the common data *fields* are the same, the data *values* might be different for saved and current status. The rest of the fields available for current data are called **current-only** status fields.

- **Saved** data consists of the common status fields noted in the syntax diagram.
  - For a sending channel data is updated before requesting confirmation that a batch of messages has been received and when confirmation has been received

- For a receiving channel data is reset just before confirming that a batch of messages has been received
- For a server connection channel no data is saved.
- Therefore, a channel that has never been current cannot have any saved status.

**Note:** Status is not saved until a persistent message is transmitted across a channel, or a nonpersistent message is transmitted with a NPMSPEED of NORMAL. Because status is saved at the end of each batch, a channel does not have any saved status until at least one batch has been transmitted.

- **Current** data consists of the common status fields and current-only status fields as noted in the syntax diagram. The data fields are continually updated as messages are sent/received.
-  **Short** data consists of the STATUS current data item and the short status field as noted in the syntax diagram.

This method of operation has the following consequences:

- An inactive channel might not have any saved status - if it has never been current or has not yet reached a point where saved status is reset.
- The "common" data fields might have different values for saved and current status.
- A current channel always has current status and might have saved status.

Channels can either be current or inactive:

#### **Current channels**

These are channels that have been started, or on which a client has connected, and that have not finished or disconnected normally. They might not yet have reached the point of transferring messages, or data, or even of establishing contact with the partner. Current channels have **current** status and might also have **saved** status.

The term **Active** is used to describe the set of current channels that are not stopped.

#### **Inactive channels**

These are channels that either:

- Have not been started
- On which a client has not connected
- Have finished
- Have disconnected normally

(Note that if a channel is stopped, it is not yet considered to have finished normally - and is, therefore, still current.) Inactive channels have either **saved** status or no status at all.

There can be more than one instance of the same named receiver, requester, cluster-receiver, or server-connection channel current at the same time (the requester is acting as a receiver). This occurs if several senders, at different queue managers, each initiate a session with this receiver, using the same channel name. For channels of other types, there can only be one instance current at any time.

For all channel types, however, there can be more than one set of saved status information available for a channel name. At most one of these sets relates to a current instance of the channel, the rest relate to previously current instances. Multiple instances arise if different transmission queue names or connection names have been used with the same channel. This can happen in the following cases:

- At a sender or server:
  - If the same channel has been connected to by different requesters (servers only)
  - If the transmission queue name has been changed in the definition
  - If the connection name has been changed in the definition
- At a receiver or requester:
  - If the same channel has been connected to by different senders or servers

- If the connection name has been changed in the definition (for requester channels initiating connection)

The number of sets that are displayed for a channel can be limited by using the XMITQ, CONNAME, and CURRENT parameters on the command.

**( *generic-channel-name* )**

The name of the channel definition for which status information is to be displayed. A trailing asterisk (\*) matches all channel definitions with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all channel definitions. A value is required for all channel types.


**WHERE**


Specify a filter condition to display status information for those channels that satisfy the selection criterion of the filter condition.

The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

**filter-keyword**

The parameter to be used to display attributes for this DISPLAY command.

 You cannot use the following parameters as filter keywords on Multiplatforms: COMPRATE, COMPTIME, CURRENT, EXITTIME, JOBNAME, NETTIME, SAVED, SHORT, XBATCSZ, or XQTIME.

 You cannot use the following parameters as filter keywords on z/OS: CHLDISP, CMDSCOPE, MCASTAT, or MONITOR.

You cannot use CONNAME or XMITQ as filter keywords if you also use them to select channel status.

Status information for channels of a type for which the filter keyword is not valid is not displayed.

**operator**

This is used to determine whether a channel satisfies the filter value on the filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which contain the specified item.

**EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not contain the specified item.

### **filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value SDR on the CHLTYPE parameter), you can only use EQ or NE.

- A generic value. This is a character string with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. Use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed.

### **ALL**

Specify this to display all the status information for each relevant instance.

If **SAVED** is specified, this causes only common status information to be displayed, not current-only status information.

If this parameter is specified, any parameters requesting specific status information that are also specified have no effect; all the information is displayed.

### **z/OS CHLDISP**

This parameter applies to z/OS only and specifies the disposition of the channels for which information is to be displayed, as used in the **START** and **STOP CHANNEL** commands, and **not** that set by **QSGDISP** for the channel definition. Values are:

#### **ALL**

This is the default value and displays requested status information for private channels.

If there is a shared queue manager environment and the command is being executed on the queue manager where it was issued, or if **CURRENT** is specified, this option also displays the requested status information for shared channels.

#### **PRIVATE**

Display requested status information for private channels.

#### **SHARED**

Display requested status information for shared channels. This is allowed only if there is a shared queue manager environment, and either:

- **CMDSCOPE** is blank or the local queue manager
- **CURRENT** is specified

**CHLDISP** displays the following values:

#### **PRIVATE**

The status is for a private channel.

#### **SHARED**

The status is for a shared channel.

#### **FIXSHARED**

The status is for a shared channel, tied to a specific queue manager.

### **z/OS CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which it was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

**Note:** See [Table 1](#), [Table 2](#), and [Table 3](#) for the permitted combinations of CHLDISP and CMDSCOPE.

**CONNNAME( *connection-name* )**


The connection name for which status information is to be displayed, for the specified channel or channels.

This parameter can be used to limit the number of sets of status information that is displayed. If it is not specified, the display is not limited in this way.

The value returned for CONNNAME might not be the same as in the channel definition, and might differ between the current channel status and the saved channel status. (Using CONNNAME for limiting the number of sets of status is therefore not recommended.)



For example, when using TCP, if CONNNAME in the channel definition:

- Is blank or is in "host name" format, the channel status value has the resolved IP address.
- Includes the port number, the current channel status value includes the port number (except on z/OS), but the saved channel status value does not.

 For SAVED or SHORT status, this value could also be the queue manager name, or queue sharing group name, on the remote system.

 For SAVED status, this value could also be:

1. The queue manager name on the remote system.
2. A combination of the queue manager name and QMID of the queue manager name on the remote system.

For the second option, the format is  QMNAME@QMID or  QMNAME&QMID, where QMNAME is the queue manager name, blank padded to the right, up to 48 characters.

To show this, in the following example there are 45 blank spaces between the character 2 (in QM2) and the @ character.

```
CONNNAME(QM2                                     @QM2_2023-07-18_12.24.06)
```

**CURRENT**

This is the default, and indicates that current status information as held by the channel initiator for current channels only is to be displayed.

Both common and current-only status information can be requested for current channels.

Short status information is not displayed if this parameter is specified.

**SAVED**

Specify this to display saved status information for both current and inactive channels.

Only common status information can be displayed. Short and current-only status information is not displayed for current channels if this parameter is specified.

### **z/OS SHORT**

This indicates that short status information and the STATUS item for current channels only is to be displayed.

Other common status and current-only status information is not displayed for current channels if this parameter is specified.

### **MONITOR**

Specify this to return the set of online monitoring parameters. These are COMPRATE, COMPTIME, EXITTIME, MONCHL, NETTIME, XBATCSZ, XQMSGSA, and XQTIME. If you specify this parameter, any of the monitoring parameters that you request specifically have no effect; all monitoring parameters are still displayed.

### **XMITQ(*q-name*)**

The name of the transmission queue for which status information is to be displayed, for the specified channel or channels.

This parameter can be used to limit the number of sets of status information that is displayed. If it is not specified, the display is not limited in this way.

The following information is always returned, for each set of status information:

- The channel name
- The transmission queue name (for sender and server channels)
- The connection name
- The remote queue manager, or queue sharing group, name (only for current status, and for all channel types except server-connection channels)
- The remote partner application name (for server-connection channels)
- The type of status information returned (CURRENT, or SAVED, or on z/OS only, SHORT)
- STATUS (except SAVED on z/OS)
- On z/OS, CHLDISP
- STOPREQ (only for current status)
- SUBSTATE

If no parameters requesting specific status information are specified (and the ALL parameter is not specified), no further information is returned.

If status information is requested that is not relevant for the particular channel type, this is not an error.

### **Common status**

The following information applies to sets of current status data and also to sets of saved status data. Some of this information does not apply to server-connection channels.

#### **CHLTYPE**

The channel type. This is one of the following:

##### **SDR**

A sender channel

##### **SVR**

A server channel

##### **RCVR**

A receiver channel

##### **RQSTR**

A requester channel

**CLUSSDR**

A cluster-sender channel

**CLUSRCVR**

A cluster-receiver channel

**SVRCONN**

A server-connection channel

**AMQP**

An AMQP channel

**CURLUWID**

The logical unit of work identifier associated with the current batch, for a sending or a receiving channel.

For a sending channel, when the channel is in doubt it is the LUWID of the in-doubt batch.

For a saved channel instance, this parameter has meaningful information only if the channel instance is in doubt. However, the parameter value is still returned when requested, even if the channel instance is not in doubt.

It is updated with the LUWID of the next batch when this is known.

This parameter does not apply to server-connection channels.

**CURMSG**

For a sending channel, this is the number of messages that have been sent in the current batch. It is incremented as each message is sent, and when the channel becomes in doubt it is the number of messages that are in doubt.

For a saved channel instance, this parameter has meaningful information only if the channel instance is in doubt. However, the parameter value is still returned when requested, even if the channel instance is not in doubt.

For a receiving channel, it is the number of messages that have been received in the current batch. It is incremented as each message is received.

The value is reset to zero, for both sending and receiving channels, when the batch is committed.

This parameter does not apply to server-connection channels.

**CURSEQNO**

For a sending channel, this is the message sequence number of the last message sent. It is updated as each message is sent, and when the channel becomes in doubt it is the message sequence number of the last message in the in-doubt batch.

For a saved channel instance, this parameter has meaningful information only if the channel instance is in doubt. However, the parameter value is still returned when requested, even if the channel instance is not in doubt.

For a receiving channel, it is the message sequence number of the last message that was received. It is updated as each message is received.

This parameter does not apply to server-connection channels.

**INDOUBT**

Whether the channel is currently in doubt.

This is only YES while the sending Message Channel Agent is waiting for an acknowledgment that a batch of messages that it has sent has been successfully received. It is NO at all other times, including the period during which messages are being sent, but before an acknowledgment has been requested.

For a receiving channel, the value is always NO.

This parameter does not apply to server-connection channels.

**LSTLUWID**

The logical unit of work identifier associated with the last committed batch of messages transferred.

This parameter does not apply to server-connection channels.

**LSTSEQNO**

Message sequence number of the last message in the last committed batch. This number is not incremented by nonpersistent messages using channels with a NPMSPEED of FAST.

This parameter does not apply to server-connection channels.

**STATUS**

Current status of the channel. This is one of the following:

**BINDING**

Channel is performing channel negotiation and is not yet ready to transfer messages.

**INITIALIZING**

The channel initiator is attempting to start a channel.

On z/OS, this is displayed as INITIALIZI.

**PAUSED**

The channel is waiting for the message-retry interval to complete before retrying an MQPUT operation.

**REQUESTING**

A local requester channel is requesting services from a remote MCA.

**RETRYING**

A previous attempt to establish a connection has failed. The MCA will reattempt connection after the specified time interval.

**RUNNING**

The channel is either transferring messages at this moment, or is waiting for messages to arrive on the transmission queue so that they can be transferred.

**STARTING**

A request has been made to start the channel but the channel has not yet begun processing. A channel is in this state if it is waiting to become active.

**STOPPED**

This state can be caused by one of the following:

- Channel manually stopped

A user has entered a stop channel command against this channel.

- Retry limit reached

The MCA has reached the limit of retry attempts at establishing a connection. No further attempt will be made to establish a connection automatically.

A channel in this state can be restarted only by issuing the START CHANNEL command, or starting the MCA program in an operating-system dependent manner.

**STOPPING**

Channel is stopping or a close request has been received.

**SWITCHING**

The channel is switching transmission queues.

On z/OS, STATUS is not displayed if saved data is requested.

**Multi** On Multiplatforms, the value of the STATUS field returned in the saved data is the status of the channel at the time the saved status was written. Normally, the saved status value is RUNNING. To see the current status of the channel, the user can use the DISPLAY CHSTATUS CURRENT command.

**Note:** For an inactive channel, CURMSGs, CURSEQNO, and CURLUWID have meaningful information only if the channel is INDOUBT. However they are still displayed and returned if requested.



## Current-only status

The following information applies only to current channel instances. The information applies to all channel types, except where stated.

### AMQPKA

The keep alive time for an AMQP channel in milliseconds. If the AMQP client has not sent any frames within the keep alive interval, then the connection is closed with a `amqp:resource-limit-exceeded` AMQP error condition.

This parameter is valid only for channels with a channel type ( `CHLTYPE` ) of AMQP

### BATCHES

Number of completed batches during this session (since the channel was started).

### BATCHSZ

The batch size being used for this session.

This parameter does not apply to server-connection channels, and no values are returned; if specified on the command, this is ignored.

### BUFSRCVD

Number of transmission buffers received. This includes transmissions to receive control information only.

### BUFSSENT

Number of transmission buffers sent. This includes transmissions to send control information only.

### BYTSRCVD

Number of bytes received during this session (since the channel was started). This includes control information received by the message channel agent.

If the value for `BYTSSENT` or `BYTSRCVD` exceeds 999999999, it wraps to zero.

### BYTSSENT

Number of bytes sent during this session (since the channel was started). This includes control information sent by the message channel agent.

If the value for `BYTSSENT` or `BYTSRCVD` exceeds 999999999, it wraps to zero.

### CHSTADA

Date when this channel was started (in the form `yyyy-mm-dd`).

### CHSTATI

Time when this channel was started (in the form `hh.mm.ss`).

### COMPHDR

The technique used to compress the header data sent by the channel. Two values are displayed:

- The default header data compression value negotiated for this channel.
- The header data compression value used for the last transmission segment that was eligible for compression, which might or might not carry a message. The header data compression value can be altered in a sending channels message exit. If no eligible transmission segment has been sent, the second value is blank.

### COMPMSG

The technique used to compress the message data sent by the channel. Two values are displayed:

- The default message data compression value negotiated for this channel.
- The message data compression value used for the last message sent. The message data compression value can be altered in a sending channels message exit. If no message has been sent, the second value is blank.

### COMPRATE

The compression rate achieved displayed to the nearest percentage; that is, a rate of 25 indicates messages are being compressed to 75% of their original length.

Two values are displayed:

- The first value based on recent activity over a short period.
- The second value based on activity over a longer period.


These values are reset every time the channel is started and are displayed only when the STATUS of the channel is RUNNING. If monitoring data is not being collected, or if no messages have been sent by the channel, the values are shown as blank.

A value is only displayed for this parameter if MONCHL is set for this channel. See [“Setting monitor values” on page 705](#).

### COMPTIME

The amount of time for each message, displayed in microseconds, spent on compression or decompression. Two values are displayed:

- The first value based on recent activity over a short period.
- The second value based on activity over a longer period.

**Note:**  On z/OS, COMPTIME is the amount of time for each message, provided that the message does not have to be processed in segments. This segmenting of the message on z/OS occurs when the message is:

- 32 KB or larger, or
- 16 KB or larger, and the channel has TLS encryption.

If the message is split into segments, COMPTIME is the time spent compressing each segment. This means that a message that is split into 8 segments actually spends (COMPTIME \* 8) microseconds during compression or decompression.

A value is only displayed for this parameter if MONCHL is set for this channel. See [“Setting monitor values” on page 705](#).

### CURSHCNV

The CURSHCNV value is blank for all channel types other than server-connection channels. For each instance of a server-connection channel, the CURSHCNV output gives a count of the number of conversations currently running over that channel instance.

A value of zero indicates that the channel is running as it did in versions of the product earlier than IBM WebSphere MQ 7.0, regarding:

- Administrator stop-quiesce
- Heartbeating
- Read ahead
- Sharing conversations
- Client Asynchronous consumption

### EXITTIME

Amount of time, displayed in microseconds, spent processing user exits per message. Two values are displayed:

- The first value based on recent activity over a short period.
- The second value based on activity over a longer period.

These values depend on the configuration and behavior of your system, as well as the levels of activity within it, and serve as an indicator that your system is performing normally. A significant variation in these values may indicate a problem with your system. They are reset every time the channel is started and are displayed only when the STATUS of the channel is RUNNING.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONCHL is set for this channel. See [“Setting monitor values” on page 705](#).

**HBINT**

The heartbeat interval being used for this session.

**JOBNAME**

A name that identifies the MQ process that is currently providing and hosting the channel.

**Multi** On Multiplatforms, this name is the concatenation of the process identifier and the thread identifier of the MCA program, displayed in hexadecimal.

**z/OS** This information is not available on z/OS. The parameter is ignored if specified.

**z/OS** You cannot use JOBNAME as a filter keyword on z/OS.

**z/OS KAINT**

The keepalive interval being used for this session. This is valid only on z/OS.

**LOCLADDR**

Local communications address for the channel. The value returned depends on the TRPTYPE of the channel (currently only TCP/IP is supported).

**LONGRTS**

Number of long retry wait start attempts left. This applies only to sender or server channels.

**LSTMSGDA**

Date when the last message was sent or MQI call was handled, see LSTMSGTI.

**LSTMSGTI**

Time when the last message was sent or MQI call was handled.

For a sender or server, this is the time the last message (the last part of it if it was split) was sent. For a requester or receiver, it is the time the last message was put to its target queue. For a server-connection channel, it is the time when the last MQI call completed.

In the case of a server-connection channel instance on which conversations are being shared, this is the time when the last MQI call completed on any of the conversations running on the channel instance.

**z/OS MAXMSGL**

The maximum message length being used for this session (valid only on z/OS ).

**MAXSHCNV**

The MAXSHCNV value is blank for all channel types other than server-connection channels. For each instance of a server-connection channel, the MAXSHCNV output gives the negotiated maximum of the number of conversations that can run over that channel instance.

A value of zero indicates that the channel is running as it did in versions earlier than IBM WebSphere MQ 7.0, regarding:

- Administrator stop-quiesce
- Heartbeating
- Read ahead
- Sharing conversations
- Client asynchronous consumption

**Multi MCASTAT**

Whether the Message Channel Agent is currently running. This is either "running" or "not running". Note that it is possible for a channel to be in stopped state, but for the program still to be running.

**z/OS** This information is not available on z/OS. The parameter is ignored if specified.


**z/OS** You cannot use MCASTAT as a filter keyword on z/OS.


## MCAUSER

The user ID used by the MCA. This can be the user ID set in the channel definition, the default user ID for message channels, a user ID transferred from a client if this is a server-connection channel, or a user ID specified by a security exit.

This parameter applies only to server-connection, receiver, requester, and cluster-receiver channels.

On server connection channels that share conversations, the MCAUSER field contains a user ID if all the conversations have the same MCA user ID value. If the MCA user ID in use varies across these conversations, the MCAUSER field contains a value of \*.

 The maximum length on [Multiplatforms](#) is 64 characters.

 The maximum length on z/OS is 12 characters.

## MONCHL

Current level of monitoring data collection for the channel.

This parameter is also displayed when you specify the MONITOR parameter.

## MSGS

Number of messages sent or received (or, for server-connection channels, the number of MQI calls handled) during this session (since the channel was started).

In the case of a server-connection channel instance on which conversations are being shared, this is the total number of MQI calls handled on all of the conversations running on the channel instance.

## NETTIME

Amount of time, displayed in microseconds, to send a request to the remote end of the channel and receive a response. This time only measures the network time for such an operation. Two values are displayed:

- The first value based on recent activity over a short period.
- The second value based on activity over a longer period.

These values depend on the configuration and behavior of your system, as well as the levels of activity within it, and serve as an indicator that your system is performing normally. A significant variation in these values may indicate a problem with your system. They are reset every time the channel is started and are displayed only when the STATUS of the channel is RUNNING.

This parameter applies only to sender, server, and cluster-sender channels.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONCHL is set for this channel. See [“Setting monitor values”](#) on page 705.

## NPMSPEED

The nonpersistent message handling technique being used for this session.

## PORT

The port number used to connect an AMQP channel. The default port for AMQP 1.0 connections is 5672.

## RAPPLTAG

The remote partner application name. This is the name of the client application at the remote end of the channel.



For Managed File Transfer, **RAPPLTAG** displays *Managed File Transfer Agent agent name*.

This parameter applies only to server-connection channels.

**Note:** If multiple IBM MQ connections use the same channel instance, that is, the channel is defined with SHARECNV > 1 and the same process makes multiple connections to the queue manager, if the connections specify different application names the RAPPLTAG field displays an asterisk: RAPPLTAG(\*).

## RPRODUCT

The remote partner product identifier. This is the product identifier of the IBM MQ code running at the remote end of the channel. The possible values are shown in Table 172 on page 701.

Product Identifier	Description
MQMM	Queue manager on a distributed platform
  MQMV	Queue manager on z/OS
MQCC	IBM MQ C client
MQNM	IBM MQ .NET fully managed client
MQJB	IBM MQ Classes for JAVA
MQJF	Managed File Transfer Agent
MQJM	IBM MQ Classes for JMS (normal mode)
MQJN	IBM MQ Classes for JMS (migration mode)
MQJU	Common Java interface to the MQI
MQXC	XMS client C/C++ (normal mode)
MQXD	XMS client C/C++ (migration mode)
MQXN	XMS client .NET (normal mode)
MQXM	XMS client .NET (migration mode)
MQXU	IBM MQ .NET XMS client (unmanaged/XA)
MQNU	IBM MQ .NET unmanaged client

## RQMNAME

The queue manager name, or queue sharing group name, of the remote system. This parameter does not apply to server-connection channels.

## RVERSION

The remote partner version. This is the version of the IBM MQ code running at the remote end of the channel.

The remote version is displayed as **VVRRMMFF**, where

### VV

Version

### RR

Release

### MM

Maintenance level

### FF


Fix level

## SECPROT

Defines the security protocol currently in use.

Does not apply to client-connection channels.

Set automatically, based on the value you set for SSLCIPH in **DEFINE CHANNEL**.

 From IBM MQ 9.2.0, **SECPROT** is supported on z/OS.

Possible values are:

**NONE**

No security protocol

**Deprecated SSLV3**

SSL 3.0

This protocol is deprecated. See [Deprecated CipherSpecs](#)

**Deprecated TLSV1**

TLS 1.0

This protocol is deprecated. See [Deprecated CipherSpecs](#)

**TLSV12**

TLS 1.2

**TLSV13**

TLS 1.3

From IBM MQ 9.2.0, the product supports the TLS 1.3 security protocol on all platforms.

**z/OS** On IBM MQ for z/OS, TLS 1.3 is supported only on z/OS 2.4 or later.

**SHORTRTS**

Number of short retry wait start attempts left. This applies only to sender or server channels.

**SSLCERTI**

The full Distinguished Name of the issuer of the remote certificate. The issuer is the Certificate Authority that issued the certificate.

The maximum length is 256 characters, so longer Distinguished Names are truncated.

**z/OS SSLCERTU**

The local user ID associated with the remote certificate. This is valid on z/OS only.

**SSLCIPH**

The CipherSpec being used by the connection.

This parameter, which already existed in **DEFINE CHANNEL**, is displayed by **DISPLAY CHSTATUS** from IBM MQ 9.2.0.

For more information, see [the SSLCIPH property in DEFINE CHANNEL](#).

The value for this parameter is also used to set the value of [SECPROT](#).

**SSLKEYDA**

Date on which the previous successful TLS secret key reset was issued.

**Note:** TLS 1.3 key resets are integral to TLS 1.3, and not communicated to applications. As a result, on z/OS queue managers, for receiver channels, this value will not be set when the channel is communicating using a TLS 1.3 CipherSpec. On distributed queue managers this value will not be accurate, and might even be set to zero at either end of a channel, when the channel is communicating using a TLS 1.3 CipherSpec.

**SSLKEYTI**

Time at which the previous successful TLS secret key reset was issued.

**Note:** TLS 1.3 key resets are integral to TLS 1.3, and not communicated to applications. As a result, on z/OS queue managers, for receiver channels, this value will not be set when the channel is communicating using a TLS 1.3 CipherSpec. On distributed queue managers this value will not be accurate, and might even be set to zero at either end of a channel, when the channel is communicating using a TLS 1.3 CipherSpec.

**SSLPEER**

Distinguished Name of the peer queue manager or client at the other end of the channel.

The maximum length is 256 characters, so longer Distinguished Names are truncated.

## SSLRKEYS

Number of successful TLS key resets. The count of TLS secret key resets is reset when the channel instance ends.

**Note:** TLS 1.3 key resets are integral to TLS 1.3, and not communicated to applications. As a result, on z/OS queue managers, for receiver channels, this value will not be set when the channel is communicating using a TLS 1.3 CipherSpec. On distributed queue managers this value will not be accurate, and might even be set to zero at either end of a channel, when the channel is communicating using a TLS 1.3 CipherSpec.

## STOPREQ

Whether a user stop request is outstanding. This is either YES or NO.

## STATCHL

Current level of statistics data collection for the channel.

## SUBSTATE

Action being performed by the channel when this command is issued. The following substates are listed in precedence order, starting with the substate of the highest precedence:

### ENDBATCH

Channel is performing end-of-batch processing.

### SEND

A request has been made to the underlying communication subsystem to send some data.

### RECEIVE

A request has been made to the underlying communication subsystem to receive some data.

## SERIALIZE

Channel is serializing its access to the queue manager. Valid on z/OS only.

### RESYNCH

Channel is resynchronizing with the partner.

### HEARTBEAT

Channel is heartbeating with the partner.

### SCYEXIT

Channel is running the security exit.

### RCVEXIT

Channel is running one of the receive exits.

### SENDEXIT

Channel is running one of the send exits.

### MSGEXIT

Channel is running one of the message exits.

### MREXIT

Channel is running the message retry exit.

### CHADEXIT

Channel is running through the channel auto-definition exit.

### NETCONNECT

A request has been made to the underlying communication subsystem to connect a partner machine.

### SSLHANDSHK

Channel is processing a TLS handshake.

### NAMESERVER

A request has been made to the name server.

### MQPUT

A request has been made to the queue manager to put a message on the destination queue.

## **MQGET**

A request has been made to the queue manager to get a message from the transmission queue (if this is a message channel ) or from an application queue (if this is an MQI channel).

## **MQICALL**

A MQ API call, other than MQPUT and MQGET, is being executed.

## **COMPRESS**

Channel is compressing or extracting data.

Not all substates are valid for all channel types or channel states. There are occasions when no substate is valid, at which times a blank value is returned.

For channels running on multiple threads, this parameter displays the substate of the highest precedence.

## **TPROOT**

The topic root for an AMQP channel. The default value for TPROOT is SYSTEM.BASE.TOPIC. With this value, the topic string an AMQP client uses to publish or subscribe has no prefix, and the client can exchange messages with other MQ pub/sub applications. To have AMQP clients publish and subscribe under a topic prefix, first create an MQ topic object with a topic string set to the prefix you want, then set TPROOT to the name of the MQ topic object you created.

This parameter is valid only for channels with a channel type ( CHLTYPE ) of AMQP

## **XBATCSZ**

Size of the batches transmitted over the channel. Two values are displayed:

- The first value based on recent activity over a short period.
- The second value based on activity over a longer period.

These values depend on the configuration and behavior of your system, as well as the levels of activity within it, and serve as an indicator that your system is performing normally. A significant variation in these values might indicate a problem with your system. They are reset every time the channel is started and are displayed only when the STATUS of the channel is RUNNING.

This parameter does not apply to server-connection channels.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONCHL is set for this channel. See [“Setting monitor values”](#) on page 705.


## **USECLTID**

Specifies that the client ID should be used for authorization checks for an AMQP channel, instead of the MCAUSER attribute value.

## **XQMSGSA**

Number of messages queued on the transmission queue available to the channel for MQGETs.

This parameter has a maximum displayable value of 999. If the number of messages available exceeds 999, a value of 999 is displayed.

 On z/OS, if the transmission queue is not indexed by *CorrelId*, this value is shown as blank.

This parameter applies to cluster-sender channels only.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONCHL is set for this channel. See [“Setting monitor values”](#) on page 705.

## **XQTIME**

The time, in microseconds, that messages remained on the transmission queue before being retrieved. The time is measured from when the message is put onto the transmission queue until



it is retrieved to be sent on the channel and, therefore, includes any interval caused by a delay in the putting application.

Two values are displayed:

- The first value based on recent activity over a short period.
- The second value based on activity over a longer period.

These values depend on the configuration and behavior of your system, as well as the levels of activity within it, and serve as an indicator that your system is performing normally. A significant variation in these values might indicate a problem with your system. They are reset every time the channel is started and are displayed only when the STATUS of the channel is RUNNING.

This parameter applies only to sender, server, and cluster-sender channels.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONCHL is set for this channel. See [“Setting monitor values” on page 705](#).

## Short status



The following information applies only to current channel instances.

### QMNAME

The name of the queue manager that owns the channel instance.

## Setting monitor values

For auto-defined cluster sender channels, these are controlled with the queue manager MONACLS parameter. See [“ALTER QMGR \(alter queue manager settings\)” on page 348](#) for more information. You cannot display or alter auto-defined cluster sender channels. However you can get their status, or issue DISPLAY CLUSQMGR, as described here: [Working with auto-defined cluster-sender channels](#).

For other channels, including manually-defined cluster sender channels, these are controlled with the channel MONCHL parameter. See [“ALTER CHANNEL \(alter channel settings\)” on page 274](#) for more information.

### Related reference

[“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\)” on page 1216](#)

The Inquire Channel Status (MQCMD\_INQUIRE\_CHANNEL\_STATUS) PCF command inquires about the status of one or more channel instances.

[“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\) Response” on page 1231](#)

The response to the Inquire Channel Status (MQCMD\_INQUIRE\_CHANNEL\_STATUS) PCF command consists of the response header followed by several structures.



## DISPLAY CHSTATUS (display channel status) AMQP

Use the MQSC command DISPLAY CHSTATUS (AMQP) to display the status of one or more AMQP channels.

## Using MQSC commands

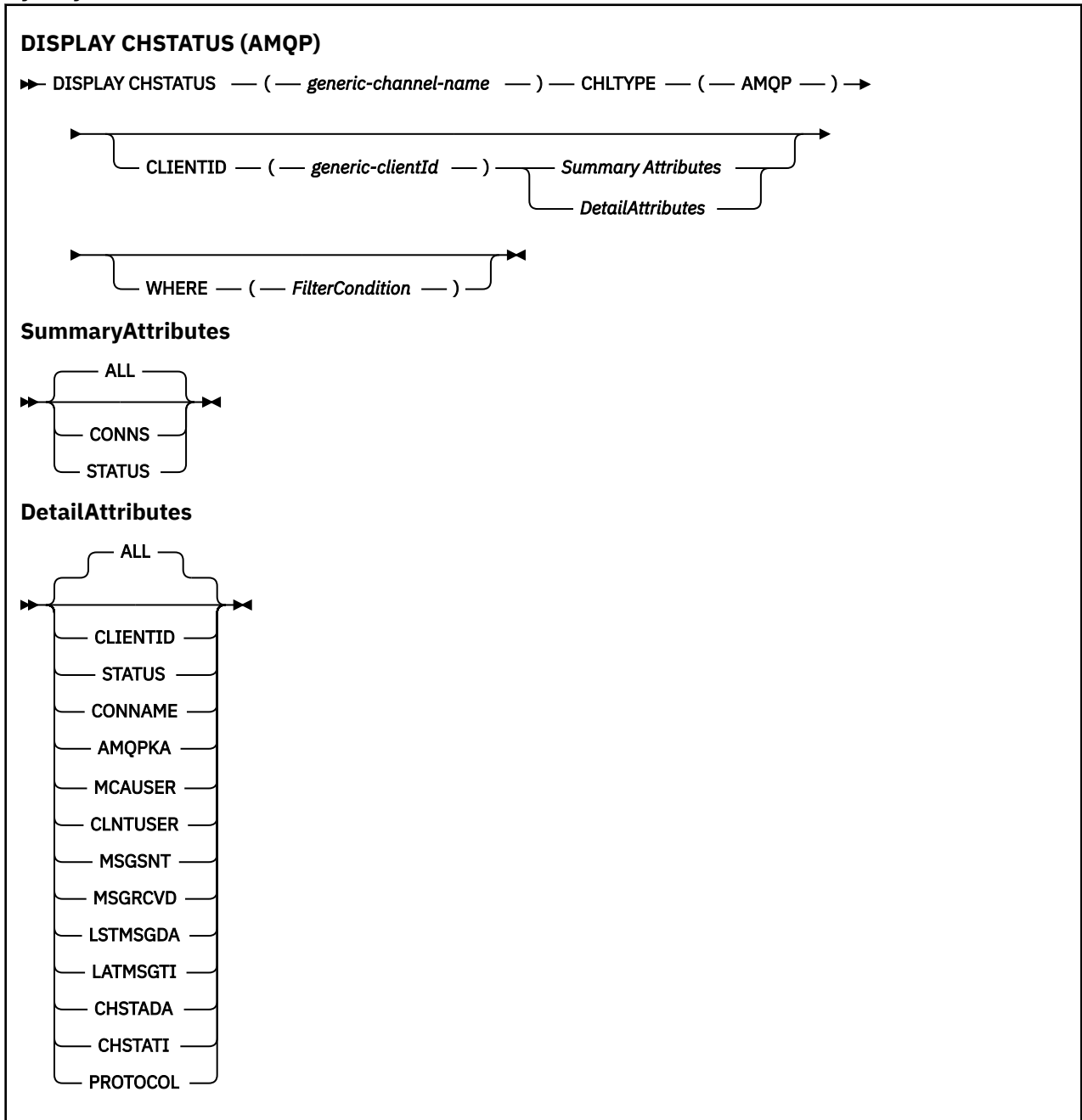
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [“Syntax diagram” on page 706](#)
- [“Parameter descriptions for DISPLAY CHSTATUS” on page 706](#)
- [“Summary attributes” on page 708](#)
- [“Client details mode” on page 708](#)

- “Examples” on page 709

## Syntax diagram

Synonym: DIS CHS



### Note:

- The default behavior is for **RUNMQSC** to return a summary of the connections to the channel. If **CLIENTID** is specified then **RUNMQSC** returns details of each client connected to the channel.

### Parameter descriptions for DISPLAY CHSTATUS

You must specify the name of the channel for which you want to display status information. This parameter can be a specific channel name or a generic channel name. By using a generic channel name, you can display either the status information for all channels, or status information for one or more channels that match the specified name.

## ( *generic-channel-name* )

The name of the channel definition for which status information is to be displayed. A trailing asterisk (\*) matches all channel definitions with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all channel definitions. A value is required for all channel types.

## WHERE

Specify a filter condition to display status information for those channels that satisfy the selection criterion of the filter condition.

The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

### **filter-keyword**

The parameter to be used to display attributes for this DISPLAY command.

Status information for channels of a type for which the filter keyword is not valid is not displayed.

### **operator**

This is used to determine whether a channel satisfies the filter value on the filter keyword. The operators are:

#### **LT**

Less than

#### **GT**

Greater than

#### **EQ**

Equal to

#### **NE**

Not equal to

#### **LE**

Less than or equal to

#### **GE**

Greater than or equal to

#### **LK**

Matches a generic string that you provide as a *filter-value*

#### **NL**

Does not match a generic string that you provide as a *filter-value*

#### **CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this operator to display objects the attributes of which contain the specified item.

#### **EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this operator to display objects the attributes of which do not contain the specified item.

### **filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this value can be:

- An explicit value, that is a valid value for the attribute that is being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value SDR on the CHLTYPE parameter), you can use EQ or NE only.

- A generic value. This value is a character string with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. Use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed.

#### **ALL**

Specify this parameter to display all the status information for each relevant instance.

If this parameter is specified, any parameters that request specific status information which are also specified have no effect; all the information is displayed.

### **Summary attributes**

When no CLIENTID parameter is added to the MQSC command DISPLAY CHSTATUS (AMQP), a summary of AMQP channel information is displayed. The number of connections is displayed as the CONNS attribute. The following attributes display a summary for each channel.

#### **ALL**

Specify this parameter to display all the status information for each relevant instance. This attribute is the default value if no attributes are requested.

This parameter is valid for AMQP channels.

If this parameter is specified, any specified parameters that are requesting specific status information have no effect; and all the information is displayed.

#### **CONNS**

The number of current connections to this channel.

#### **STATUS**

The status of this channel.

### **Client details mode**

#### **CLIENTID**

The identifier of the client.

#### **STATUS**

The status of the client.

#### **CONNAME**

The name of the remote connection (IP address)

#### **AMQPKA**

The client's keep alive interval.

#### **MCAUSER**

The user ID that the client is using to access IBM MQ resources.

#### **CLNTUSER**

The user ID that the client provided when it connected.

#### **MSGSENT**

Number of messages sent by the client since it connected last.

#### **MSGRCVD**

Number of messages received by the client since it connected last.

## LSTMSGDA

Date last message was received or sent.

## LSTMSGTI

Time last message was received or sent.

## CHSTADA

Date channel started.

## CHSTATI

Time channel was started.

## PROTOCOL

The communication protocol used by the client. The value is AMQP.

## Examples

The following command retrieves a status summary for the AMQP channel named MYAMQP:

```
dis chstatus(MYAMQP) chltype(AMQP) all
```

The command outputs the following status:

```
AMQ8417: Display Channel Status details.
CHANNEL(MYAMQP)                CHLTYPE(AMQP)
CONNECTIONS(1)                  STATUS(RUNNING)
```

The following command retrieves a full status for the AMQP channel named MYAMQP:

```
dis chstatus(*) chltype(AMQP) clientid(*) all
```

The command outputs the following status:

```
AMQ8417: Display Channel Status details.
CHANNEL(MYAMQP)                CHLTYPE(AMQP)
CLIENTID(recv_cc2022b)        STATUS(RUNNING)
CONNAME(192.168.60.1)         AMQPKA(0)
MCAUSER(matt)                 CLNTUSER( )
MSGCNT(0)                     MSGRCVD(0)
LSTMSGDA( )                   LSTMSGTI( )
CHSTADA(2015-09-18)          CHSTATI(06.23.30)
PROTOCOL(AMQP)
```

## Windows Linux AIX **DISPLAY CHSTATUS (display channel status)**

### MQTT

Use the MQSC command DISPLAY CHSTATUS (MQTT) to display the status of one or more MQ Telemetry channels.

### Using MQSC commands

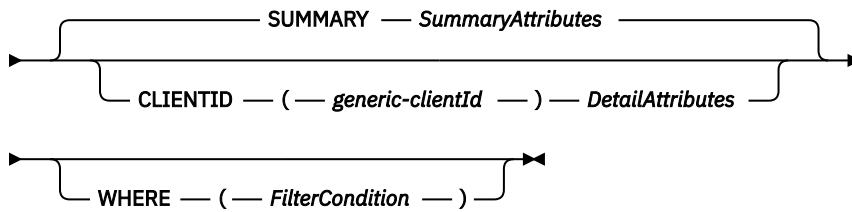
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY CHSTATUS” on page 711](#)
- [“Summary attributes” on page 712](#)

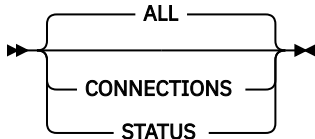
**Synonym:** DIS CHS

## DISPLAY CHSTATUS (MQTT)

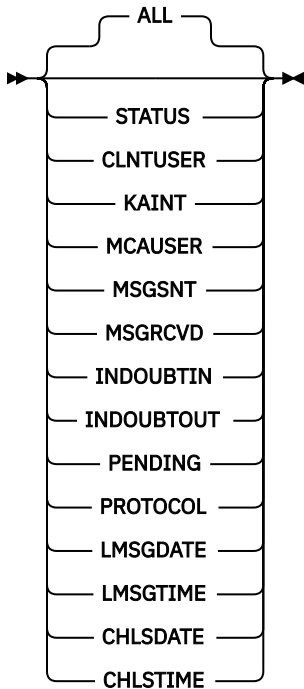
►► DISPLAY CHSTATUS — ( — *generic-channel-name* — ) — CHLTYPE — ( — MQTT — ) —►



### SummaryAttributes



### DetailAttributes



### Notes:

- The default behavior is for **RUNMQSC** to return a summary of the connections to the channel. If **CLIENTID** is specified then **RUNMQSC** returns details of each client connected to the channel.
- Either **CLIENTID**, **SUMMARY**, or neither may be specified, but not both at the same time.
- The **DISPLAY CHSTATUS** command for MQ Telemetry has the potential to return a far larger number of responses than if the command was run for an IBM MQ channel. For this reason, the MQ Telemetry server does not return more responses than fit on the reply-to queue. The number of responses is limited to the value of **MAXDEPTH** parameter of the **SYSTEM.MQSC.REPLY.QUEUE** queue. When **RUNMQSC** processes an MQ Telemetry command that is truncated by the MQ Telemetry server, the **AMQ8492** message is displayed specifying how many responses are returned based on the size of **MAXDEPTH**.
- You can use this command to list disconnected clients. As these clients are not associated with a particular channel, you list them using the wildcard character. For example,

```
DIS CHS(*) CHLTYPE(MQTT) CLIENTID(*) WHERE(STATUS EQ DISCONNECTED).
```

You should take care using this command when there could be a large number of disconnected clients.

## Parameter descriptions for DISPLAY CHSTATUS

You must specify the name of the channel for which you want to display status information. This parameter can be a specific channel name or a generic channel name. By using a generic channel name, you can display either the status information for all channels, or status information for one or more channels that match the specified name.

### ( *generic-channel-name* )

The name of the channel definition for which status information is to be displayed. A trailing asterisk (\*) matches all channel definitions with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all channel definitions. A value is required for all channel types.

## WHERE

Specify a filter condition to display status information for those channels that satisfy the selection criterion of the filter condition.

The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

### **filter-keyword**

The parameter to be used to display attributes for this DISPLAY command.

Status information for channels of a type for which the filter keyword is not valid is not displayed.

### **operator**

This is used to determine whether a channel satisfies the filter value on the filter keyword. The operators are:

#### **LT**

Less than

#### **GT**

Greater than

#### **EQ**

Equal to

#### **NE**

Not equal to

#### **LE**

Less than or equal to

#### **GE**

Greater than or equal to

#### **LK**

Matches a generic string that you provide as a *filter-value*

#### **NL**

Does not match a generic string that you provide as a *filter-value*

#### **CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this operator to display objects the attributes of which contain the specified item.

#### **EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this operator to display objects the attributes of which do not contain the specified item.

### **filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this value can be:

- An explicit value, that is a valid value for the attribute that is being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value SDR on the CHLTYPE parameter), you can use EQ or NE only.

- A generic value. This value is a character string with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. Use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed.

## **ALL**

Specify this parameter to display all the status information for each relevant instance.

If this parameter is specified, any parameters that request specific status information which are also specified have no effect; all the information is displayed.

## **Summary attributes**

When SUMMARY is added to the MQSC command DISPLAY CHSTATUS (MQTT), the number of connections is displayed as the CONNECTIONS attribute. The following attributes display a summary for each channel.

## **ALL**

Specify this parameter to display all the status information for each relevant instance. This attribute is the default value if no attributes are requested.

This parameter is valid for MQTT channels.

If this parameter is specified, any specified parameters that are requesting specific status information have no effect; and all the information is displayed.

## **CONNECTIONS**

The number of current connections to this channel.

## **STATUS**

The status of this channel.

## **Client details mode**

### **STATUS**

The status of the client.

### **CLNTUSER**

The user ID that the client provided when it connected.

### **CONNAME**

The name of the remote connection (IP address)

### **KAINT**

The client's keep alive interval.

### **MCAUSER**

The user ID that the client is using to access IBM MQ resources. This is the client user ID selected by the process described in [MQTT client identity and authorization](#).



**MSGSENT**

Number of messages sent by the client since it connected last.

**MSGRCVD**

Number of messages received by the client since it connected last.

**INDOUBTIN**

Number of in doubt, inbound messages to the client.

**INDOUBTOUT**

Number of in doubt, outbound messages to the client.

**PENDING**

Number of outbound pending messages.

**PROTOCOL**

The communication protocol used by the client. This is MQTTV311, MQTTV3, or HTTP.

**LMSGDATE**

Date last message was received or sent.

**LMSGTIME**

Time last message was received or sent.

**CHLSDATE**

Date channel started.

**CHLSTIME**


Time channel was started.

## **DISPLAY CLUSQMGR (display channel information for cluster queue managers)**

Use the MQSC command **DISPLAY CLUSQMGR** to display information about cluster channels for queue managers in a cluster.

### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 716](#)
- [“Parameter descriptions for DISPLAY CLUSQMGR” on page 716](#)
- [“Requested parameters” on page 718](#)
- [“Channel parameters” on page 719](#)

Synonym : DIS CLUSQMGR

## DISPLAY CLUSQMGR

►► DISPLAY CLUSQMGR — ( — *generic-qmname* — ) —————  
 WHERE — ( — *FilterCondition* — )

ALL CHANNEL — ( — *generic-name* — )

CLUSTER — ( — *generic-name* — )

CMDSCOPE(' ') 2  
 CMDSCOPE — ( — *qmgr-name* — ) 1  
 CMDSCOPE(\*) 1  
 Requested attributes

Channel attributes

### Requested attributes

CLUSDATE  
 CLUSTIME  
 DEFTYPE  
 QMID  
 QMTYPE  
 STATUS  
 SUSPEND  
 VERSION

### Channel attributes

ALTDATE
ALTIME
BATCHHB
BATCHINT
BATCHLIM
BATCHSZ
CLWLPRTY
CLWLRANK
CLWLWGHT
COMPHDR
COMPMSG
CONNNAME
CONVERT
DESCR
DISCINT
HBINT
KAINT
LOCLADDR
LONGRTY
LONGTMR
MAXMSGL
MCANAME
MCATYPE
MCAUSER
MODENAME
MRDATA
MREXIT
MRRTY
MRTMR
MSGDATA
MSGEXIT
NETPRTY
NPMSPEED
PASSWORD <sup>3</sup>
PROPCTL
PUTAUT
RCVDATA
RCVEXIT
SCYDATA
SCYEXIT
SENDDATA
SENDEXIT
SEQWRAP
SHORTRTY
SHORTTMR
SSLCAUTH
SSLCIPH
SSLPEER
TPNAME
TRPTYPE
USEDLQ
USERID
XMITQ

Notes:

- <sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>2</sup> Valid only on z/OS.

<sup>3</sup> Not valid on z/OS.

## Usage notes

Unlike the **DISPLAY CHANNEL** command, this command includes information about cluster channels that are auto-defined, and the status of cluster channels.

**Note:** On z/OS, the command fails if the channel initiator is not started.

## Parameter descriptions for DISPLAY CLUSQMGR

### ( *generic-qmgr-name* )

The name of the cluster queue manager for which information is to be displayed.

A trailing asterisk "\*" matches all cluster queue managers with the specified stem followed by zero or more characters. An asterisk "\*" on its own specifies all cluster queue managers.

### WHERE

Specify a filter condition to display only those cluster channels that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Almost any parameter that can be used to display attributes for this **DISPLAY** command. However, you cannot use the CMDSCOPE or MCANAME parameters as filter keywords. You cannot use CHANNEL or CLUSTER as filter keywords if you use them to select cluster queue managers.

#### **operator**

The operators are:

##### **LT**

Less than

##### **GT**

Greater than

##### **EQ**

Equal to

##### **NE**

Not equal to

##### **LE**

Less than or equal to

##### **GE**

Greater than or equal to

##### **LK**

Matches a generic string that you provide as a *filter-value*

##### **NL**

Does not match a generic string that you provide as a *filter-value*

##### **CT**

Contains a specified item. If the *filter-keyword* is a list, you can use CT to display objects the attributes of which contain the specified item.

##### **EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use EX to display objects the attributes of which do not contain the specified item.

##### **CTG**

Contains an item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use CTG to display objects the attributes of which match the generic string.

## EXG

Does not contain any item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use EXG to display objects the attributes of which do not match the generic string.

### filter-value

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, *filter-value* can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, , or GE only. If the attribute value is a value from a possible set of values, you can use only EQ or NE. For example, the value STARTING on the **STATUS** parameter.

- A generic value. *filter-value* is a character string. An example is ABC\*. If the operator is LK, all items where the attribute value begins with the string, ABC in the example, are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.


You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. The value can be explicit or, if it is a character value, it can be explicit or generic. If it is explicit, use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed. If it is generic, use CTG or EXG as the operator. If ABC\* is specified with the operator CTG, all items where one of the attribute values begins with ABC are listed.

## ALL

Specify ALL to display all the parameters. If this parameter is specified, any parameters that are also requested specifically have no effect; all parameters are still displayed.

ALL is the default if you do not specify a generic name and do not request any specific parameters.

 On z/OS ALL is also the default if you specify a filter condition using the WHERE parameter, but on other platforms, only requested attributes are displayed.

## CHANNEL ( *generic-name* )

This is optional, and limits the information displayed to cluster channels with the specified channel name. The value can be a generic name.

## CLUSTER ( *generic-name* )

This is optional, and limits the information displayed to cluster queue managers with the specified cluster name. The value can be a generic name.

## CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. '' is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered. You can enter a different queue manager name, if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of \* is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

## Requested parameters

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

Some parameters are relevant only for cluster channels of a particular type or types. Attributes that are not relevant for a particular type of channel cause no output, and do not cause an error.

### **CLUSDATE**

The date on which the definition became available to the local queue manager, in the form yyyy-mm-dd.

### **CLUSTIME**

The time at which the definition became available to the local queue manager, in the form hh.mm.ss.

### **DEFTYPE**

How the cluster channel was defined:

#### **CLUSSDR**

As a cluster-sender channel from an explicit definition.

#### **CLUSSDRA**

As a cluster-sender channel by auto-definition alone.

#### **CLUSSDRB**

As a cluster-sender channel by auto-definition and an explicit definition.

#### **CLUSRCVR**

As a cluster-receiver channel from an explicit definition.

### **QMID**

The internally generated unique name of the cluster queue manager.

### **QMTYPE**

The function of the cluster queue manager in the cluster:

#### **REPOS**

Provides a full repository service.

#### **NORMAL**

Does not provide a full repository service.

### **STATUS**

The status of the channel for this cluster queue manager is one of the following values:

#### **STARTING**

The channel was started and is waiting to become active.

#### **BINDING**


The channel is performing channel negotiation and is not yet ready to transfer messages.

#### **INACTIVE**

The channel is not active.

#### **INITIALIZING**

The channel initiator is attempting to start a channel.

 On z/OS, INITIALIZING is displayed as INITIALIZI.

#### **RUNNING**

The channel is either transferring messages at this moment, or is waiting for messages to arrive on the transmission queue so that they can be transferred.

#### **STOPPING**

The channel is stopping, or received a close request.

**RETRYING**

A previous attempt to establish a connection failed. The MCA attempts to connect again after the specified time interval.

**PAUSED**

The channel is waiting for the message-retry interval to complete before trying an MQPUT operation again.

**STOPPED**

This state can be caused by one of the following events:

- Channel manually stopped.

A user entered a stop channel command for this channel.

- The number of attempts to establish a connection reached the maximum number of attempts allowed for the channel.

No further attempt is made to establish a connection automatically.

A channel in this state can be restarted only by issuing the **START CHANNEL** command, or starting the MCA program in an operating-system dependent manner.

**REQUESTING**

A local requester channel is requesting services from a remote MCA.

**SWITCHING**

The channel is switching transmission queues.

**SUSPEND**

Specifies whether this cluster queue manager is suspended from the cluster or not (as a result of the **SUSPEND QMGR** command). The value of SUSPEND is either YES or NO.

**VERSION**

The version of the IBM MQ installation that the cluster queue manager is associated with.

The version has the format VVRRMMFF:

- VV: Version
- RR: Release
- MM: Maintenance level
- FF: Fix level

**XMITQ**

The cluster transmission queue.

**Channel parameters****ALTDATE**

The date on which the definition or information was last altered, in the form yyyy-mm-dd

**ALTTIME**

The time at which the definition or information was last altered, in the form hh.mm.ss

**BATCHHB**

The batch heartbeat value being used.

**BATCHINT**

Minimum batch duration.

**BATCHLIM**

Batch data limit.

The limit of the amount of data that can be sent through a channel.

**BATCHSZ**

Batch size.

**CLWLPRTY**

The priority of the channel for the purposes of cluster workload distribution.

**CLWLRANK**

The rank of the channel for the purposes of cluster workload distribution.

**CLWLWGHT**

The weighting of the channel for the purposes of cluster workload distribution.

**COMPHDR**

The list of header data compression techniques supported by the channel.

**COMPMSG**

The list of message data compression techniques supported by the channel.

**CONNAME**

Connection name.

**CONVERT**

Specifies whether the sender converts application message data.

**DESCR**

Description.

**DISCONT**

Disconnection interval.

**HBINT**

Heartbeat interval.

**KAINIT**

KeepAlive timing for the channel.

**LOCLADDR**

Local communications address for the channel.

**LONGRTY**

Limit of number of attempts to connect using the long duration timer.

**LONGTMR**

Long duration timer.

**MAXMSGL**

Maximum message length for channel.

**MCANAME**

Message channel agent name.

You cannot use MCANAME as a filter keyword.

**MCATYPE**

Specifies whether the message channel agent runs as a separate process or a separate thread.

**MCAUSER**

Message channel agent user identifier.

**MODENAME**

LU 6.2 mode name.

**MRDATA**

Channel message-retry exit user data.

**MREXIT**

Channel message-retry exit name.

**MRRTY**

Channel message-retry count.

**MRTMR**

Channel message-retry time.

**MSGDATA**

Channel message exit user data.



**MSGEXIT**

Channel message exit names.

**NETPRTY**

The priority for the network connection.

**NPMSPEED**

Nonpersistent message speed.

**PASSWORD**

Password for initiating LU 6.2 session (if nonblank, PASSWORD is displayed as asterisks).

**PROPCTL**

Message property control.

**PUTAUT**

Put authority.

**RCVDATA**

Channel receive exit user data.

**RCVEXIT**

Channel receive exit names.

**SCYDATA**

Channel security exit user data.

**SCYEXIT**

Channel security exit name.

**SENDDATA**

Channel send exit user data.

**SENDEXIT**

Channel send exit names.

**SEQWRAP**

Sequence number wrap value.

**SHORTRTY**

Limit of number of attempts to connect using the short duration timer.

**SHORTTMR**

Short duration timer.

**SSLCAUTH**

Specifies whether TLS client authentication is required.

**SSLCIPH**

Cipher specification for the TLS connection.

**SSLPEER**

Filter for the Distinguished Name from the certificate of the peer queue manager or client at the other end of the channel.

**TRPTYPE**

Transport type.

**TPNAME**

LU 6.2 transaction program name.

**USEDLQ**

Determines whether the dead-letter queue is used when messages cannot be delivered by channels.

**USERID**

User identifier for initiating LU 6.2 session.

For more information about channel parameters, see [“DEFINE CHANNEL \(define a new channel\)”](#) on page 466

## **DISPLAY CMDSERV (display command server status) on z/OS**

Use the MQSC command DISPLAY CMDSERV to display the status of the command server.

### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DISPLAY CMDSERV” on page 722](#)

**Synonym:** DIS CS

#### **DISPLAY CMDSERV**

▶ DISPLAY CMDSERV ◀

### Usage notes for DISPLAY CMDSERV

1. The command server takes messages from the system command input queue, and commands using CMDSCOPE, and processes them. DISPLAY CMDSERV displays the status of the command server.
2. The response to this command is a message showing the current status of the command server, which is one of the following:

#### **ENABLED**

Available to process commands

#### **DISABLED**

Not available to process commands

#### **STARTING**

START CMDSERV in progress

#### **STOPPING**

STOP CMDSERV in progress

#### **STOPPED**

STOP CMDSERV completed

#### **RUNNING**

Available to process commands, currently processing a message

#### **WAITING**

Available to process commands, currently waiting for a message

## **DISPLAY COMMINFO (display communication information) on Multiplatforms**

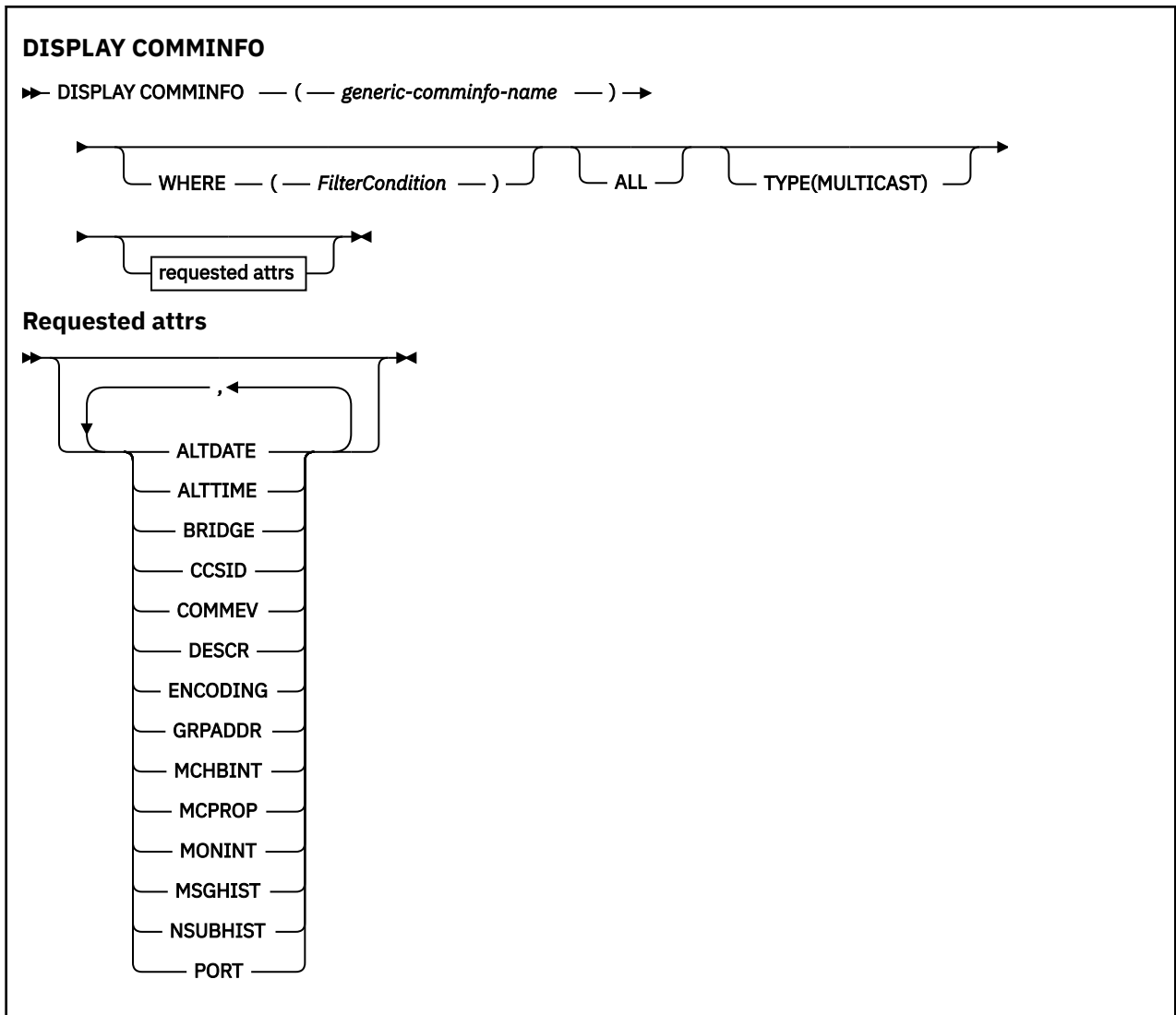
Use the MQSC command DISPLAY COMMINFO to display the attributes of a communication information object.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY COMMINFO” on page 723](#)
- [“Requested parameters” on page 724](#)

**Synonym:** DIS COMMINFO



## Parameter descriptions for DISPLAY COMMINFO

You must specify the name of the communication information object you want to display. This can be a specific communication information object name or a generic communication information object name. By using a generic communication information object name, you can display either:

- All communication information object definitions
- One or more communication information objects that match the specified name

### **(*generic-comminfo-name*)**

The name of the communication information object definition to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk (\*) matches all communication information objects with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all communication information objects. The names must all be defined to the local queue manager.

### **WHERE**

Specify a filter condition to display only those communication information object definitions that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Almost any parameter that can be used to display attributes for this DISPLAY command.

**operator**

This is used to determine whether a communication information object definition satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value DISABLED on the COMMEV parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

**ALL**

Specify this to display all the parameters. If this parameter is specified, any parameters that are requested specifically have no effect; all parameters are still displayed.

**TYPE**

Indicates the type of namelist to be displayed.

**MULTICAST**

Displays multicast communication information objects. This is the default.

**Requested parameters**

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

The default, if no parameters are specified (and the ALL parameter is not specified) is that the object names and TYPE parameters are displayed.

**ALTDATE**

The date on which the definition was last altered, in the form yyyy-mm-dd

**ALTTIME**

The time at which the definition was last altered, in the form hh.mm.ss

**BRIDGE**

Multicast bridging

**CCSID**

The coded character set identifier that messages are transmitted on.

**COMMEV**

Whether event messages are generated for Multicast.

**DESCR( *string* )**

Description

**ENCODING**

The encoding that the messages are transmitted in.

**GRPADDR**

The group IP address or DNS name.

**MCHBINT**

Multicast heartbeat interval.

**MCPROP**

Multicast property control

**MONINT**

Monitoring frequency.

**MSGHIST**

The amount of message history in kilobytes that is kept by the system to handle retransmissions in the case of NACKs (negative acknowledgments).

**NSUBHIST**

How much history a new subscriber joining a publication stream receives.

**PORT**


The port number to transmit on.

**DISPLAY CONN (display application connection information)**

Use the MQSC command **DISPLAY CONN** to display connection information about the applications connected to the queue manager. This is a useful command because it enables you to identify applications with long-running units of work.

**Using MQSC commands**

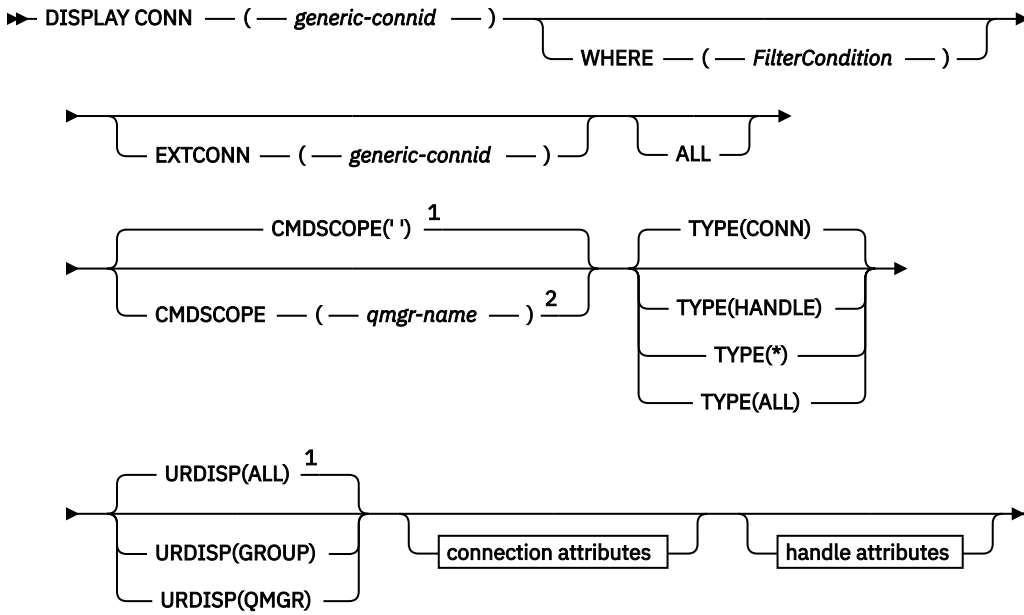
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

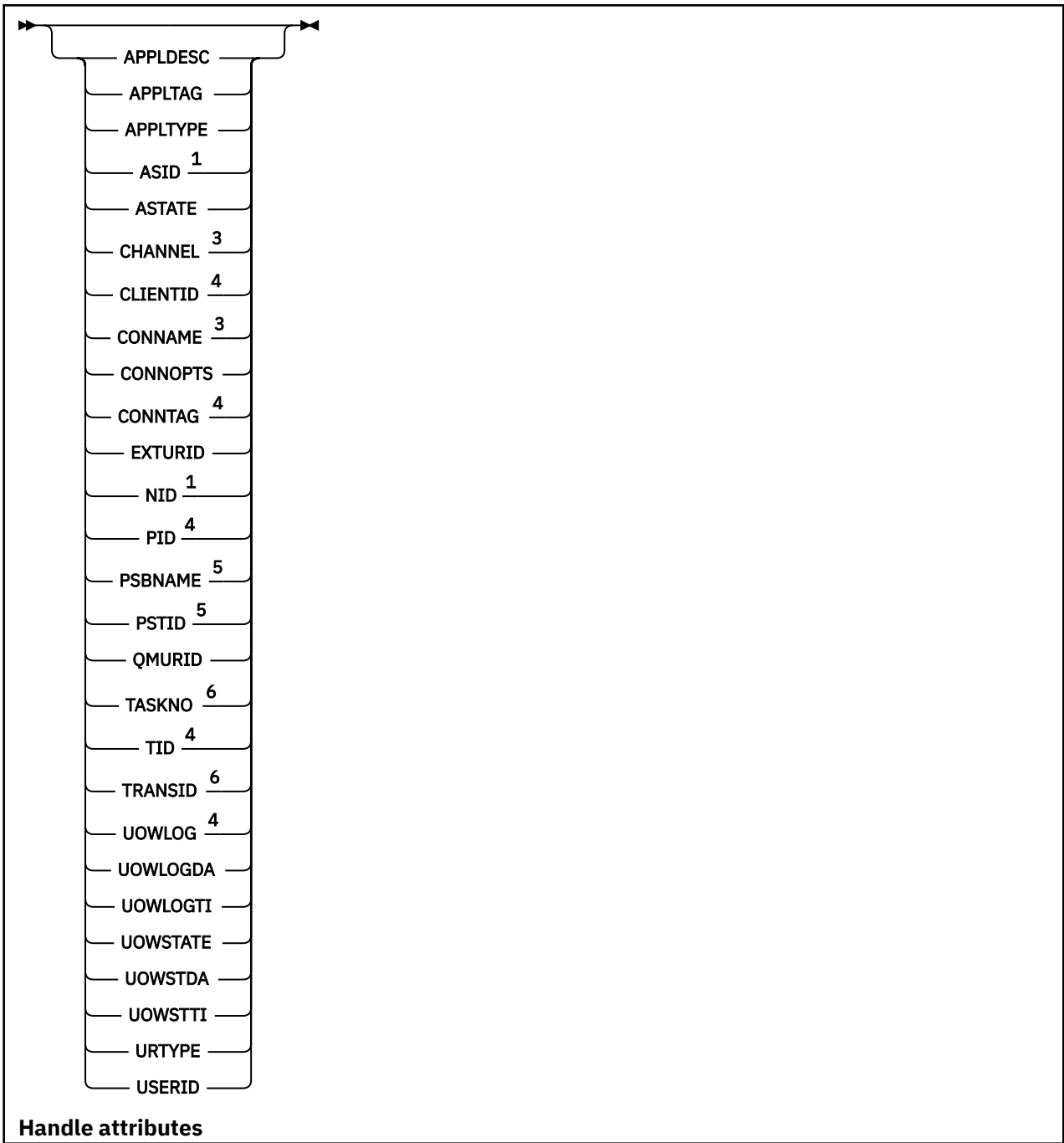
- [“Usage notes for DISPLAY CONN” on page 728](#)
- [“Parameter descriptions for DISPLAY CONN” on page 728](#)
- [“Connection attributes” on page 731](#)
- [“Handle attributes” on page 735](#)
- [“Full attributes” on page 739](#)

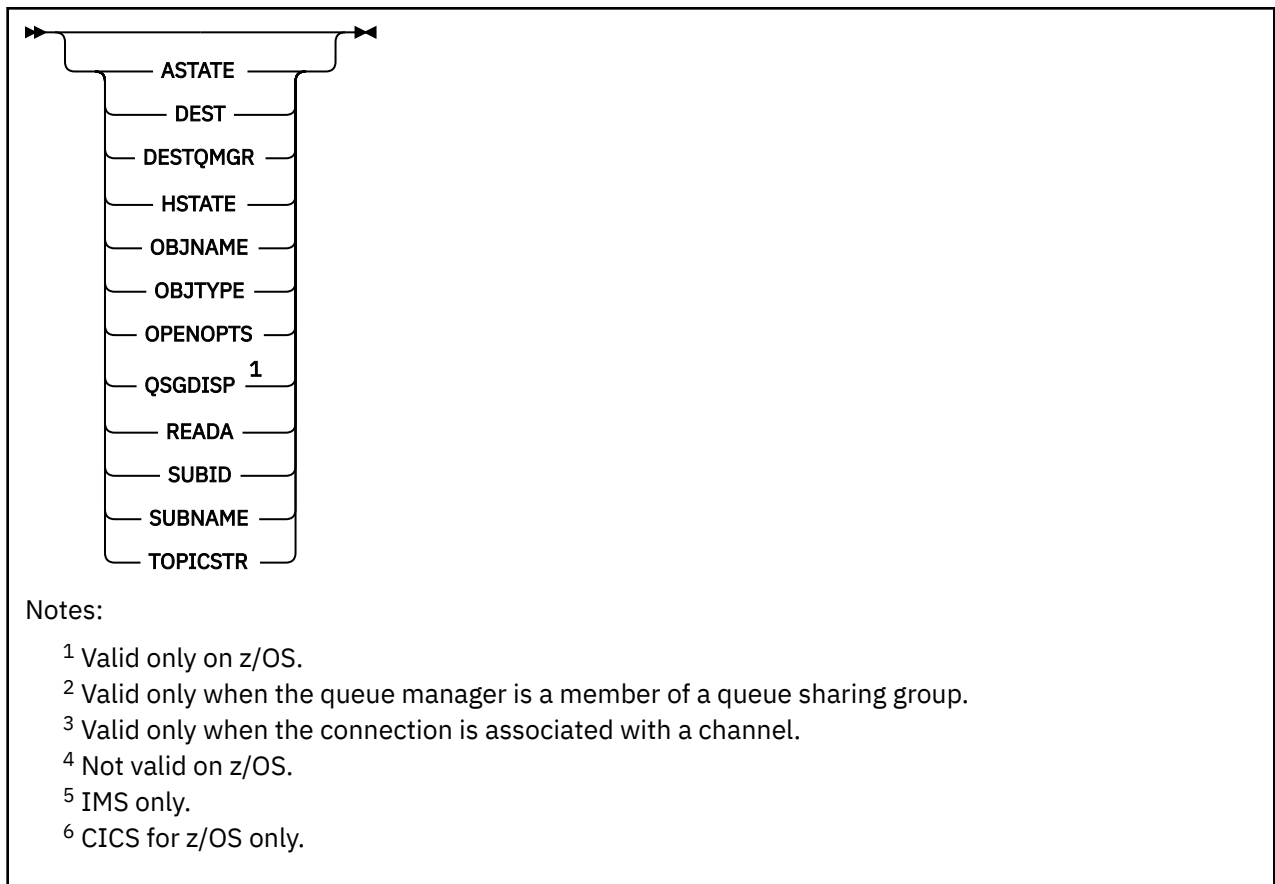
**Synonym:** DIS CONN

## DISPLAY CONN




### Connection attributes






## Usage notes for DISPLAY CONN

1.  This command is issued internally by IBM MQ on z/OS when taking a checkpoint, and when the queue manager is starting and stopping, so that a list of units of work that are in doubt at the time is written to the z/OS console log.
2. The TOPICSTR parameter might contain characters that cannot be translated into printable characters when the command output is displayed.

 On z/OS, these non-printable characters will be displayed as blanks.

 On Multiplatforms platforms using **runmqsc**, these non-printable characters will be displayed as dots.

3. The state of asynchronous consumers, ASTATE, reflects that of the server-connection proxy on behalf of the client application; it does not reflect the client application state.

From IBM MQ 8.0, there is a change to the data that is returned in the EXTURID field on the results shown for the **DISPLAY CONN runmqsc** command when there is no XA transaction associated with the connection. Prior to IBM MQ 8.0, if there is no XA transaction associated with the connection then within the EXTURID attribute the XA\_FORMATID field would be show as [00000000]. From IBM MQ 8.0, if there is no XA transaction associated with the connection, then the XA\_FORMATID value is shown as the empty string [].

## Parameter descriptions for DISPLAY CONN

You must specify a connection for which you want to display information. This can be a specific connection identifier or a generic connection identifier. A single asterisk (\*) can be used as a generic connection identifier to display information for all connections.



### **(generic-connid)**

The identifier of the connection definition for which information is to be displayed. A single asterisk (\*) specifies that information for all connection identifiers is to be displayed.

When an application connects to IBM MQ, it is given a unique 24-byte connection identifier (ConnectionId). The value for CONN is formed by converting the last eight bytes of the ConnectionId to its 16 -character hexadecimal equivalent.

### **WHERE**

Specify a filter condition to display only those connections that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Almost any parameter that can be used to display attributes for this **DISPLAY** command. However, you cannot use the **CMDSCOPE**, **EXTCONN**, **QSGDISP**, **TYPE**, and **EXTURID** parameters as filter keywords.

#### **operator**

This is used to determine whether a connection satisfies the filter value on the given filter keyword. The operators are:

##### **LT**

Less than

##### **GT**

Greater than

##### **EQ**

Equal to

##### **NE**

Not equal to

##### **LE**

Less than or equal to

##### **GE**

Greater than or equal to

##### **LK**

Matches a generic string that you provide as a *filter-value*

##### **NL**

Does not match a generic string that you provide as a *filter-value*

##### **CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which contain the specified item. You cannot use the **CONNOPTS** value MQCNO\_STANDARD\_BINDING with this operator.

##### **EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not contain the specified item. You cannot use the **CONNOPTS** value MQCNO\_STANDARD\_BINDING with this operator.

#### **filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value NONE on the **UOWSTATE** parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string in the **APPLTAG** parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL,

all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. Use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed.

## ALL

Specify this to display all the connection information of the requested type for each specified connection. This is the default if you do not specify a generic identifier, and do not request any specific parameters.

## **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which it was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use **CMDSCOPE** as a filter keyword.

## EXTCONN

The value for **EXTCONN** is based on the first sixteen bytes of the ConnectionId converted to its 32-character hexadecimal equivalent.

Connections are identified by a 24-byte connection identifier. The connection identifier comprises a prefix, which identifies the queue manager, and a suffix which identifies the connection to that queue manager. By default, the prefix is for the queue manager currently being administered, but you can specify a prefix explicitly by using the **EXTCONN** parameter. Use the **CONN** parameter to specify the suffix.

When connection identifiers are obtained from other sources, specify the fully qualified connection identifier (both **EXTCONN** and **CONN**) to avoid possible problems related to non-unique **CONN** values.

Do not specify both a generic value for **CONN** and a non-generic value for **EXTCONN**.


You cannot use **EXTCONN** as a filter keyword.

## TYPE

Specifies the type of information to be displayed. Values are:

### CONN

Connection information for the specified connection.

 On z/OS, this includes threads which may be logically or actually disassociated from a connection, together with those that are in-doubt and for which external intervention is needed to resolve them. These latter threads are those that **DIS THREAD TYPE(INDOUBT)** would show.

### HANDLE

Information relating to any objects opened by the specified connection.

\*

Display all available information relating to the connection.

**ALL**

Display all available information relating to the connection.

▶ **z/OS** On z/OS, if you specify **TYPE**(ALL/\*) and **WHERE**(xxxxx) you only get CONN or HANDLE information returned, based on the **WHERE** specification. That is, if the xxxxx is a condition relating to handle attributes then only handle attributes for the connection are returned.

**URDISP**

Specifies the unit of recovery disposition of connections to be displayed. Values are:

**ALL**

Display all connections. This is the default option.

**GROUP**

Display only those connections with a GROUP unit of recovery disposition.

**QMGR**

Display only those connections with a QMGR unit of recovery disposition.

**Connection attributes**

If **TYPE** is set to CONN, the following information is always returned for each connection that satisfies the selection criteria, except where indicated:

- Connection identifier (**CONN** parameter)
- Type of information returned (**TYPE** parameter)

The following parameters can be specified for **TYPE**(**CONN**) to request additional information for each connection. If a parameter is specified that is not relevant for the connection, operating environment, or type of information requested, that parameter is ignored.

**APPLDESC**

A string containing a description of the application connected to the queue manager, where it is known. If the application is not recognized by the queue manager the description returned is blank.

**APPLTAG**

A string containing the tag of the application connected to the queue manager. It is one of the following:

- ▶ **z/OS** z/OS batch job name
- ▶ **z/OS** TSO USERID
- CICS APPLID
- ▶ **z/OS** IMS region name
- Channel initiator job name
- ▶ **IBM i** IBM i job name
- ▶ **Linux** ▶ **AIX** UNIX process

**Notes:**

- ▶ **Linux** On Linux, if the process name exceeds 15 characters, only the first 15 characters are shown.
- ▶ **AIX** On AIX, if the process name exceeds 28 characters, only the first 28 characters are shown.
- ▶ **Windows** Windows process

**Note:** This consists of the full program path and executable file name. If it is more than 28 characters long, only the last 28 characters are shown.

- Internal queue manager process name

## APPLTYPE

A string indicating the type of the application that is connected to the queue manager. It is one of the following:

### BATCH

Application using a batch connection

### RRSBATCH

RRS-coordinated application using a batch connection

### CICS

CICS transaction

### IMS

IMS transaction

### CHINIT

Channel initiator

### OS400

An IBM i application

### SYSTEM

Queue manager

### SYSTEMEXT

Application performing an extension of function that is provided by the queue manager

### UNIX

A Linux or AIX application

### USER

A user application

### WINDOWSNT

A Windows application

### ASID

A 4-character address-space identifier of the application identified by **APPLTAG**. It distinguishes duplicate values of **APPLTAG**.

This parameter is returned only on z/OS when the **APPLTYPE** parameter does not have the value SYSTEM.

This parameter is valid only on z/OS.

## ASTATE

The state of asynchronous consumption on this connection handle.

Possible values are:

### SUSPENDED

An MQCTL call with the Operation parameter set to MQOP\_SUSPEND has been issued against the connection handle so that asynchronous message consumption is temporarily suspended on this connection.

### STARTED

An MQCTL call with the Operation parameter set to MQOP\_START has been issued against the connection handle so that asynchronous message consumption can proceed on this connection.

### STARTWAIT

An MQCTL call with the Operation parameter set to MQOP\_START\_WAIT has been issued against the connection handle so that asynchronous message consumption can proceed on this connection.

**STOPPED**

An MQCTL call with the Operation parameter set to MQOP\_STOP has been issued against the connection handle so that asynchronous message consumption cannot currently proceed on this connection.

**NONE**

No MQCTL call has been issued against the connection handle. Asynchronous message consumption cannot currently proceed on this connection.

**CHANNEL**

The name of the channel that owns the connection. If there is no channel associated with the connection, this parameter is blank.

 **CLIENTID**

The client ID of the client that is using the connection. If there is no client ID associated with the connection, this parameter is blank.

**CONNAME**

The connection name associated with the channel that owns the connection. If there is no channel associated with the connection, this parameter is blank.

**CONNOPTS**

The connect options currently in force for this application connection. Possible values are:

- MQCNO\_ACCOUNTING\_MQI\_DISABLED
- MQCNO\_ACCOUNTING\_MQI\_ENABLED
- MQCNO\_ACCOUNTING\_Q\_DISABLED
- MQCNO\_ACCOUNTING\_Q\_ENABLED
- MQCNO\_FASTPATH\_BINDING
- MQCNO\_HANDLE\_SHARE\_BLOCK
- MQCNO\_HANDLE\_SHARE\_NO\_BLOCK
- MQCNO\_HANDLE\_SHARE\_NONE
- MQCNO\_ISOLATED\_BINDING
- MQCNO\_RECONNECT
- MQCNO\_RECONNECT\_Q\_MGR
- MQCNO\_RESTRICT\_CONN\_TAG\_Q\_MGR
- MQCNO\_RESTRICT\_CONN\_TAG\_QSG
- MQCNO\_SERIALIZE\_CONN\_TAG\_Q\_MGR
- MQCNO\_SERIALIZE\_CONN\_TAG\_QSG
- MQCNO\_SHARED\_BINDING
- MQCNO\_STANDARD\_BINDING

If you are using an IBM MQ 9.1.2 or later client, the values displayed for MQCNO\_RECONNECT and MQCNO\_RECONNECT\_Q\_MGR are the effective reconnect options. If you are using an earlier client version, the values displayed are whatever the application specifies, whether they are currently taking effect or not.

You cannot use the value MQCNO\_STANDARD\_BINDING as a filter value with the CT and EX operators on the **WHERE** parameter.

 **CONNTAG**

The connection tag associated with this connection, formatted as a readable string in the local codepage for the RUNMQSC.

**Note:** The *CONNTAG* is treated as string data, so it can be filtered using the syntax `WHERE (CONNTAG LK 'generic_tag*')`.

## EXTURID

The external unit of recovery identifier associated with this connection. Its format is determined by the value of **URTYPE**.

You cannot use **EXTURID** as a filter keyword.

### z/OS NID

Origin identifier, set only if the value of **UOWSTATE** is UNRESOLVED. This is a unique token identifying the unit of work within the queue manager. It is of the form `origin-node.origin-urid` where

- `origin-node` identifies the originator of the thread, except in the case where **APPLTYPE** is set to RRSBATCH, when it is omitted.
- `origin-urid` is the hexadecimal number assigned to the unit of recovery by the originating system for the specific thread to be resolved.

This parameter is valid only on z/OS.

## PID

Number specifying the process identifier of the application that is connected to the queue manager.

z/OS This parameter is not valid on z/OS.

### z/OS PSBNAME

The 8-character name of the program specification block (PSB) associated with the running IMS transaction. You can use the **PSBNAME** and **PSTID** to purge the transaction using IMS commands. It is valid on z/OS only.

This parameter is returned only when the **APPLTYPE** parameter has the value IMS.

### z/OS PSTID

The 4-character IMS program specification table (PST) region identifier for the connected IMS region. It is valid on z/OS only.

This parameter is returned only when the **APPLTYPE** parameter has the value IMS.

## QMURID

The queue manager unit of recovery identifier.

z/OS On z/OS, this is an 8-byte log RBA, displayed as 16 hexadecimal characters.

Multi On Multiplatforms, this is an 8-byte transaction identifier, displayed as `m.n` where `m` and `n` are the decimal representation of the first and last 4 bytes of the transaction identifier.

z/OS You can use **QMURID** as a filter keyword. On z/OS, you must specify the filter value as a hexadecimal string.

Multi On platforms other than z/OS, you must specify the filter value as a pair of decimal numbers separated by a period (.). You can only use the EQ, NE, GT, LT, GE, or LE filter operators.

z/OS However, on z/OS, if log shunting has taken place, as indicated by message CSQR026I, instead of the RBA you have to use the URID from the message.

### z/OS TASKNO

A 7-digit CICS task number. This number can be used in the CICS command "CEMT SET TASK(taskno) PURGE" to end the CICS task. This parameter is valid on z/OS only.

This parameter is returned only when the **APPLTYPE** parameter has the value CICS.

## TID

Number specifying the thread identifier within the application process that has opened the specified queue.

z/OS This parameter is not valid on z/OS.

### ▶ z/OS **TRANSID**

A 4-character CICS transaction identifier. This parameter is valid only on z/OS.

This parameter is returned only when the **APPLTYPE** parameter has the value CICS.

### ▶ Multi **UOWLOG**

The file name of the extent to which the transaction associated with this connection first wrote.

▶ Multi This parameter is valid only on [Multiplatforms](#).

### **UOWLOGDA**

The date that the transaction associated with the current connection first wrote to the log.

### **UOWLOGTI**

The time that the transaction associated with the current connection first wrote to the log.

### **UOWSTATE**

The state of the unit of work. It is one of the following:

#### **NONE**

There is no unit of work.

#### **ACTIVE**

The unit of work is active.

#### **PREPARED**

The unit of work is in the process of being committed.

### ▶ z/OS **UNRESOLVED**

The unit of work is in the second phase of a two-phase commit operation. IBM MQ holds resources on its behalf and external intervention is required to resolve it. This might be as simple as starting the recovery coordinator (such as CICS, IMS, or RRS) or it might involve a more complex operation such as using the **RESOLVE INDOUBT** command. The UNRESOLVED value can occur only on z/OS.

### **UOWSTDA**

The date that the transaction associated with the current connection was started.

### **UOWSTTI**

The time that the transaction associated with the current connection was started.

### **URTYPE**

The type of unit of recovery as seen by the queue manager. It is one of the following:

- ▶ z/OS CICS (valid only on z/OS)
- XA
- ▶ z/OS RRS (valid only on z/OS)
- ▶ z/OS IMS (valid only on z/OS)
- QMGR

**URTYPE** identifies the **EXTURID** type and not the type of the transaction coordinator. When **URTYPE** is QMGR, the associated identifier is in **QMURID** (and not **EXTURID**).

### **USERID**

The user identifier associated with the connection.

This parameter is not returned when **APPLTYPE** has the value SYSTEM.

## **Handle attributes**

If **TYPE** is set to HANDLE, the following information is always returned for each connection that satisfies the selection criteria, except where indicated:

- Connection identifier (**CONN** parameter)

- Read ahead status (**DEFREADA** parameter)
- Type of information returned (**TYPE** parameter)
- Handle status (**HSTATE**)
- Object name (**OBJNAME** parameter)
- Object type (**OBJTYPE** parameter)

The following parameters can be specified for **TYPE(HANDLE)** to request additional information for each queue. If a parameter is specified that is not relevant for the connection, operating environment, or type of status information requested, that parameter is ignored.

#### **ASTATE**

The state of the asynchronous consumer on this object handle.

Possible values are:

#### **ACTIVE**

An MQCB call has set up a function to call back to process messages asynchronously and the connection handle has been started so that asynchronous message consumption can proceed.

#### **INACTIVE**

An MQCB call has set up a function to call back to process messages asynchronously but the connection handle has not yet been started, or has been stopped or suspended, so that asynchronous message consumption cannot currently proceed.

#### **SUSPENDED**

The asynchronous consumption callback has been suspended so that asynchronous message consumption cannot currently proceed on this object handle. This can be either because an MQCB call with Operation MQOP\_SUSPEND has been issued against this object handle by the application, or because it has been suspended by the system. If it has been suspended by the system, as part of the process of suspending asynchronous message consumption the callback function will be called with the reason code that describes the problem resulting in suspension. This will be reported in the Reason field in the MQCBC structure that is passed to the callback function.

For asynchronous message consumption to proceed, the application must issue an MQCB call with the Operation parameter set to MQOP\_RESUME.

#### **SUSPTEMP**

The asynchronous consumption callback has been temporarily suspended by the system so that asynchronous message consumption cannot currently proceed on this object handle. As part of the process of suspending asynchronous message consumption, the callback function will be called with the reason code that describes the problem resulting in suspension. This will be reported in the Reason field in the MQCBC structure passed to the callback function.

The callback function will be called again when asynchronous message consumption is resumed by the system, when the temporary condition has been resolved.

#### **NONE**

An MQCB call has not been issued against this handle, so no asynchronous message consumption is configured on this handle.

#### **DEST**

The destination queue for messages that are published to this subscription. This parameter is only relevant for handles of subscriptions to topics. It is not returned for other handles.

#### **DESTQMGR**

The destination queue manager for messages that are published to this subscription. This parameter is relevant only for handles of subscriptions to topics. It is not returned for other handles. If DEST is a queue that is hosted on the local queue manager, this parameter will contain the local queue manager name. If DEST is a queue that is hosted on a remote queue manager, this parameter will contain the name of the remote queue manager.

#### **HSTATE**

The state of the handle.



Possible values are:

**ACTIVE**

An API call from this connection is currently in progress for this object. If the object is a queue, this condition can arise when an MQGET WAIT call is in progress.

If there is an MQGET SIGNAL outstanding, then this does not mean, by itself, that the handle is active.

**INACTIVE**

No API call from this connection is currently in progress for this object. If the object is a queue, this condition can arise when no MQGET WAIT call is in progress.


**OBJNAME**

The name of an object that the connection has open.

**OBJTYPE**

The type of the object that the connection has open. If this handle is that of a subscription to a topic, then the **SUBID** parameter identifies the subscription. You can then use the **DISPLAY SUB** command to find all the details about the subscription.

It is one of the following:

- QUEUE
- PROCESS
- QMGR
-  STGCLASS (valid only on z/OS)
- NAMELIST
- CHANNEL
- AUTHINFO
- TOPIC

**OPENOPTS**

The open options currently in force for the connection for the object. This parameter is not returned for a subscription. Use the value in the **SUBID** parameter and the **DISPLAY SUB** command to find the details about the subscription.

Possible values are:

**MQOO\_INPUT\_AS\_Q\_DEF**

Open queue to get messages using queue-defined default.

**MQOO\_INPUT\_SHARED**

Open queue to get messages with shared access.

**MQOO\_INPUT\_EXCLUSIVE**

Open queue to get messages with exclusive access.

**MQOO\_BROWSE**

Open queue to browse messages.

**MQOO\_OUTPUT**

Open queue or topic to put messages.

**MQOO\_INQUIRE**

Open queue to inquire attributes.

**MQOO\_SET**

Open queue to set attributes.

**MQOO\_BIND\_ON\_OPEN**

Bind handle to destination when queue is found.

**MQOO\_BIND\_NOT\_FIXED**

Do not bind to a specific destination.

**MQOO\_SAVE\_ALL\_CONTEXT**

Save context when message retrieved.

**MQOO\_PASS\_IDENTITY\_CONTEXT**

Allow identity context to be passed.

**MQOO\_PASS\_ALL\_CONTEXT**

Allow all context to be passed.

**MQOO\_SET\_IDENTITY\_CONTEXT**

Allow identity context to be set.

**MQOO\_SET\_ALL\_CONTEXT**

Allow all context to be set.

**MQOO\_ALTERNATE\_USER\_AUTHORITY**

Validate with specified user identifier.

**MQOO\_FAIL\_IF QUIESCING**

Fail if queue manager is quiescing.

 **QSGDISP**

Indicates the disposition of the object. It is valid on z/OS only. The value is one of the following:

**QMGR**

The object was defined with **QSGDISP(QMGR)**.

**COPY**

The object was defined with **QSGDISP(COPY)**.

**SHARED**

The object was defined with **QSGDISP(SHARED)**.

You cannot use **QSGDISP** as a filter keyword.

**READA**

The read ahead connection status.

Possible values are:

**NO**

Read ahead of non-persistent messages is not enabled for this object.

**YES**

Read ahead of non-persistent message is enabled for this object and is being used efficiently.

**BACKLOG**

Read ahead of non-persistent messages is enabled for this object. Read ahead is not being used efficiently because the client has been sent a large number of messages which are not being consumed.

**INHIBITED**

Read ahead was requested by the application but has been inhibited because of incompatible options specified on the first MQGET call.

**SUBID**

The internal, all-time unique identifier of the subscription. This parameter is relevant only for handles of subscriptions to topics. It is not returned for other handles.

Not all subscriptions show up in **DISPLAY CONN**; only those that have current handles open to the subscription show up. You can use the **DISPLAY SUB** command to see all subscriptions.

**SUBNAME**

The application's unique subscription name that is associated with the handle. This parameter is relevant only for handles of subscriptions to topics. It is not returned for other handles. Not all subscriptions will have a subscription name.

**TOPICSTR**

The resolved topic string. This parameter is relevant for handles with **OBJTYPE(TOPIC)**. For any other object type, this parameter is not returned.

## Full attributes

If **TYPE** is set to \*, or ALL, both Connection attributes and Handle attributes are returned for each connection that satisfies the selection criteria.

## **Multi** DISPLAY ENTAUTH (display entity authorizations) on Multiplatforms

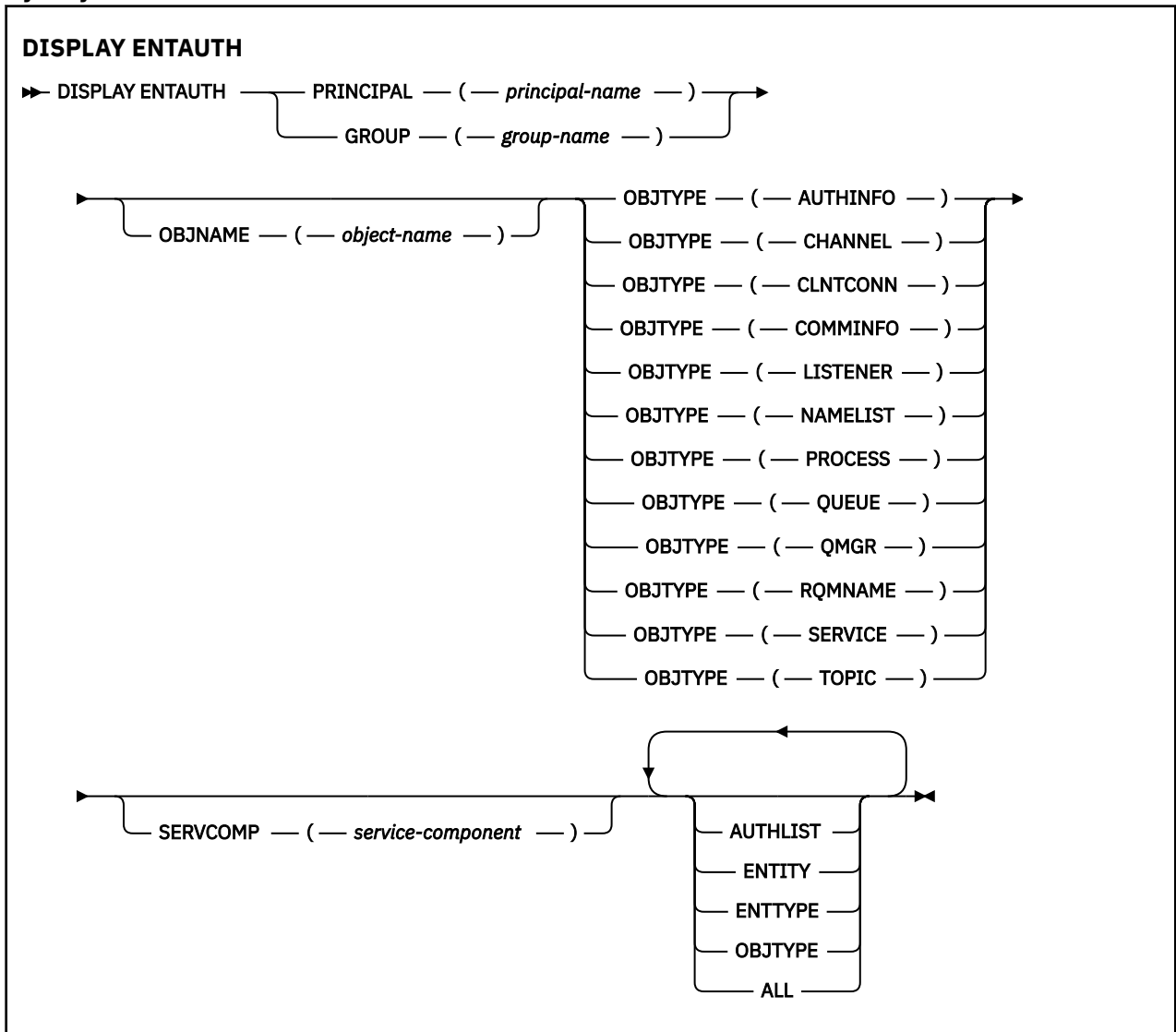
Use the MQSC command DISPLAY ENTAUTH to display the authorizations an entity has to a specified object.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions” on page 740](#)
- [“Requested parameters” on page 741](#)

**Synonym:** DIS ENTAUTH



## Parameter descriptions

### **PRINCIPAL**(*principal-name*)

A principal name. This is the name of a user for whom to retrieve authorizations to the specified object. On IBM MQ for Windows, the name of the principal can optionally include a domain name, specified in this format: `user@domain`.

You must specify either PRINCIPAL or GROUP.

### **GROUP**(*group-name*)

A group name. This is the name of the user group on which to make the inquiry. You can specify one name only and it must be the name of an existing user group.

**Windows** For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following formats:

```
GroupName@domain  
domain\GroupName
```

You must specify either PRINCIPAL or GROUP.

### **OBJNAME**(*object-name*)

The name of the object or generic profile for which to display the authorizations.

This parameter is required unless the OBJTYPE parameter is QMGR. This parameter can be omitted if the OBJTYPE parameter is QMGR.

### **OBJTYPE**

The type of object referred to by the profile. Specify one of the following values:

#### **AUTHINFO**

Authentication information record

#### **CHANNEL**

Channel

#### **CLNTCONN**

Client connection channel

#### **COMMINFO**

Communication information object

#### **LISTENER**

Listener

#### **NAMELIST**

Namelist

#### **PROCESS**

Process

#### **QUEUE**

Queue

#### **QMGR**

Queue manager

#### **RQMNAME**

Remote queue manager

#### **SERVICE**

Service

#### **TOPIC**

Topic

### **SERVCOMP**(*service-component*)

The name of the authorization service for which information is to be displayed.

If you specify this parameter, it specifies the name of the authorization service to which the authorizations apply. If you omit this parameter, the inquiry is made to the registered authorization services in turn in accordance with the rules for chaining authorization services.

#### ALL

Specify this value to display all of the authorization information available for the entity and the specified profile.

### Requested parameters

You can request the following information about the authorizations:

#### AUTHLIST

Specify this parameter to display the list of authorizations.

#### ENTITY

Specify this parameter to display the entity name.

#### ENTTYPE

Specify this parameter to display the entity type.

#### OBJTYPE

Specify this parameter to display the object type.

### z/OS **DISPLAY GROUP (display QSG information) on z/OS**

Use the MQSC command DISPLAY GROUP to display information about the queue sharing group to which the queue manager is connected. This command is valid only when the queue manager is a member of a queue sharing group.

### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DISPLAY GROUP” on page 741](#)
- [“Parameter descriptions for DISPLAY GROUP” on page 742](#)

**Synonym:** DIS GROUP



### Usage notes for DISPLAY GROUP

1. The response to the DISPLAY GROUP command is a series of messages containing information about the queue sharing group to which the queue manager is connected.

The following information is returned:

- The name of the queue sharing group
- Whether all the queue managers that belong to the group are active or inactive
- The names of all the queue managers that belong to the group.

- If you specify OBSMSGS (YES), whether queue managers in the group contain obsolete messages in Db2

## Parameter descriptions for DISPLAY GROUP

### OBSMSGS

Specifies whether the command additionally looks for obsolete messages in Db2. This is optional. Possible values are:

#### NO

Obsolete messages in Db2 are not looked for. This is the default value.

#### YES

Obsolete messages in Db2 are looked for and messages containing information about any found are returned.

## Multi **DISPLAY LISTENER (display listener information) on Multiplatforms**

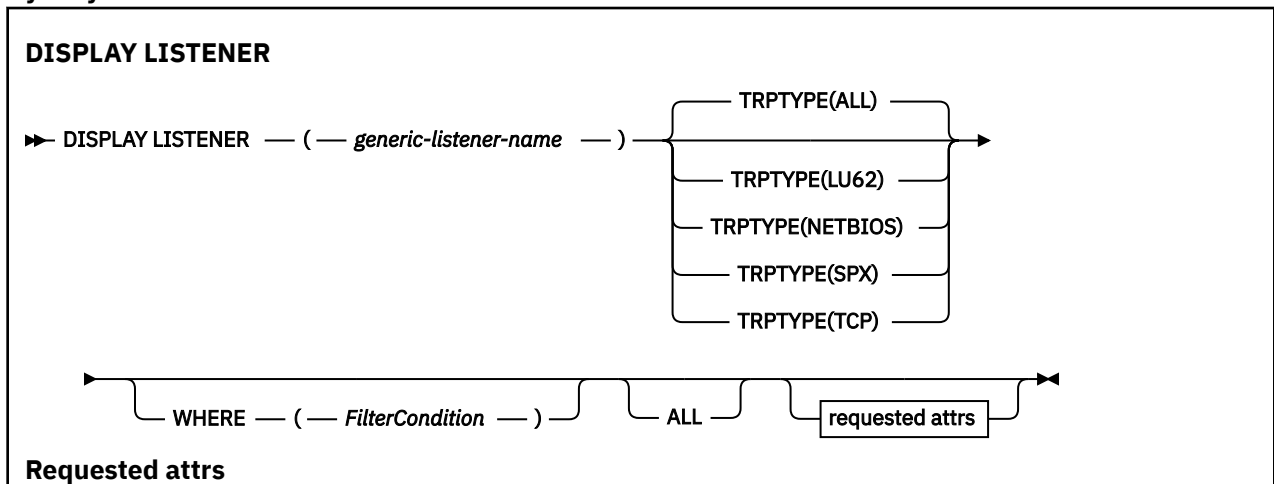
Use the MQSC command DISPLAY LISTENER to display information about a listener.

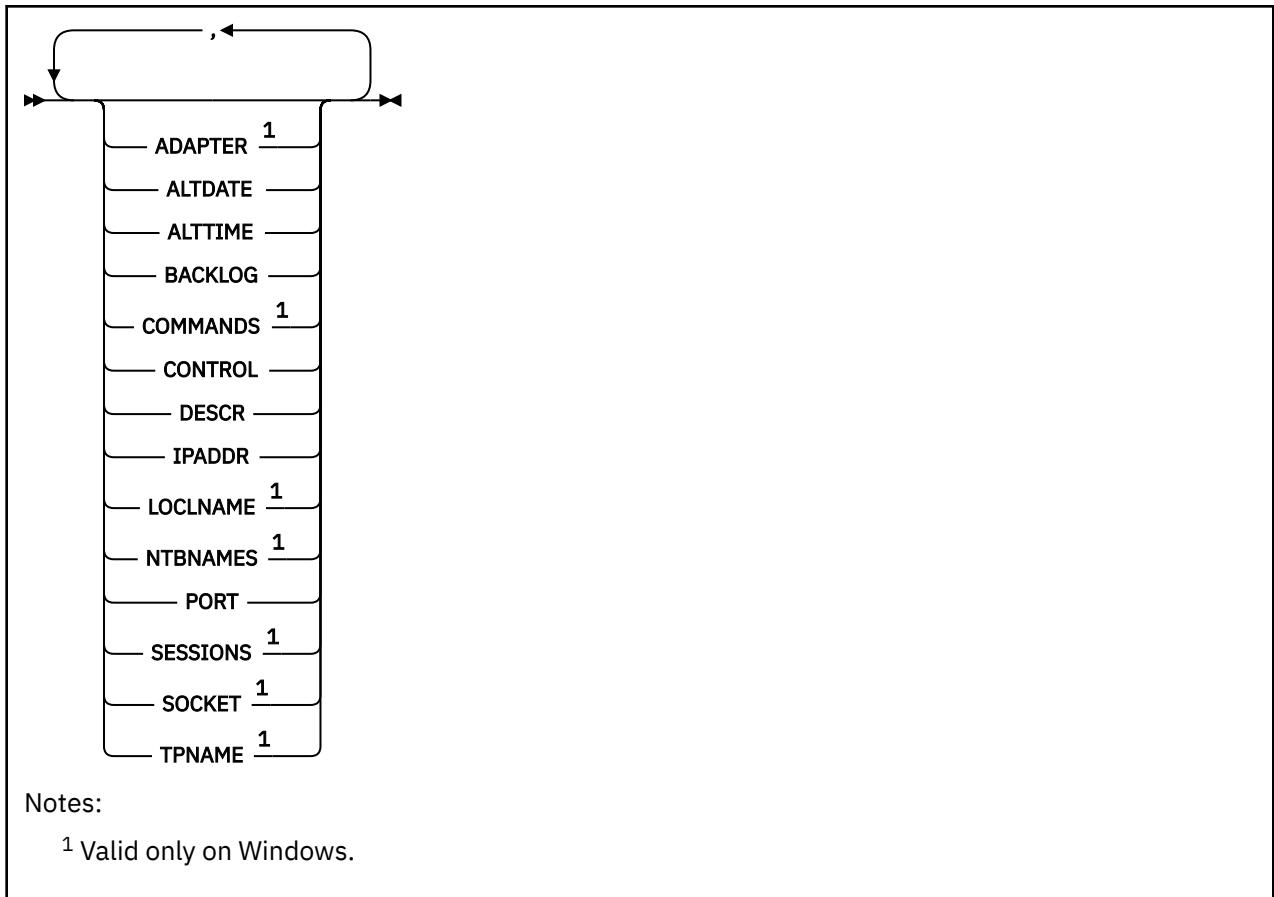
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Usage notes” on page 743](#)
- [“Keyword and parameter descriptions for DISPLAY LISTENER” on page 743](#)
- [“Requested parameters” on page 744](#)

**Synonym:** DIS LSTR





### Usage notes

The values displayed describe the current definition of the listener. If the listener has been altered since it was started, the currently running instance of the listener object may not have the same values as the current definition.

### Keyword and parameter descriptions for DISPLAY LISTENER

You must specify a listener for which you want to display information. You can specify a listener by using either a specific listener name or a generic listener name. By using a generic listener name, you can display either:

- Information about all listener definitions, by using a single asterisk (\*), or
- Information about one or more listeners that match the specified name.

#### ( *generic-listener-name* )

The name of the listener definition for which information is to be displayed. A single asterisk (\*) specifies that information for all listener identifiers is to be displayed. A character string with an asterisk at the end matches all listeners with the string followed by zero or more characters.

#### TRPTYPE

Transmission protocol. If you specify this parameter, it must follow directly after the *generic-listener-name* parameter. If you do not specify this parameter, a default of ALL is assumed. Values are:

##### ALL

This is the default value and displays information for all listeners.

##### LU62

Displays information for all listeners defined with a value of LU62 in their TRPTYPE parameter.

##### NETBIOS

Displays information for all listeners defined with a value of NETBIOS in their TRPTYPE parameter.

**SPX**

Displays information for all listeners defined with a value of SPX in their TRPTYPE parameter.

**TCP**

Displays information for all listeners defined with a value of TCP in their TRPTYPE parameter.

**WHERE**

Specify a filter condition to display information for those listeners that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

**filter-keyword**

Any parameter that can be used to display attributes for this DISPLAY command.

**operator**

This is used to determine whether a listener satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.
- A generic value. This is a character string, with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**ALL**

Specify this to display all the listener information for each specified listener. If this parameter is specified, any parameters that are requested specifically have no effect; all parameters are still displayed.

This is the default if you do not specify a generic identifier, and do not request any specific parameters.

**Requested parameters**

Specify one or more attributes that define the data to be displayed. The attributes can be specified in any order. Do not specify the same attribute more than once.



**ADAPTER**

The adapter number on which NetBIOS listens.

**ALTDATE**

The date on which the definition was last altered, in the form yyyy-mm-dd.

**ALTTIME**

The time at which the definition was last altered, in the form hh.mm.ss.

**BACKLOG**

The number of concurrent connection requests that the listener supports.

**COMMANDS**

The number of commands that the listener can use.

**CONTROL**

How the listener is to be started and stopped:

**MANUAL**

The listener is not to be started automatically or stopped automatically. It is to be controlled by use of the START LISTENER and STOP LISTENER commands.

**QMGR**

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

**STARTONLY**

The listener is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**DESCR**

Descriptive comment.

**IPADDR**

The listener's IP address.

**LOCLNAME**

The NetBIOS local name that the listener uses.

**NTBNAMES**

The number of names that the listener can use.

**PORT**

The port number for TCP/IP.

**SESSIONS**

The number of sessions that the listener can use.

**SOCKET**

SPX socket.

**TPNAME**

The LU6.2 transaction program name.

For more information on these parameters, see [“DEFINE LISTENER \(define a new listener\) on Multiplatforms”](#) on page 530.

 **DISPLAY LOG (display log information) on z/OS**

Use the MQSC command **DISPLAY LOG** to display log system parameters and information.

**Using MQSC commands on z/OS**

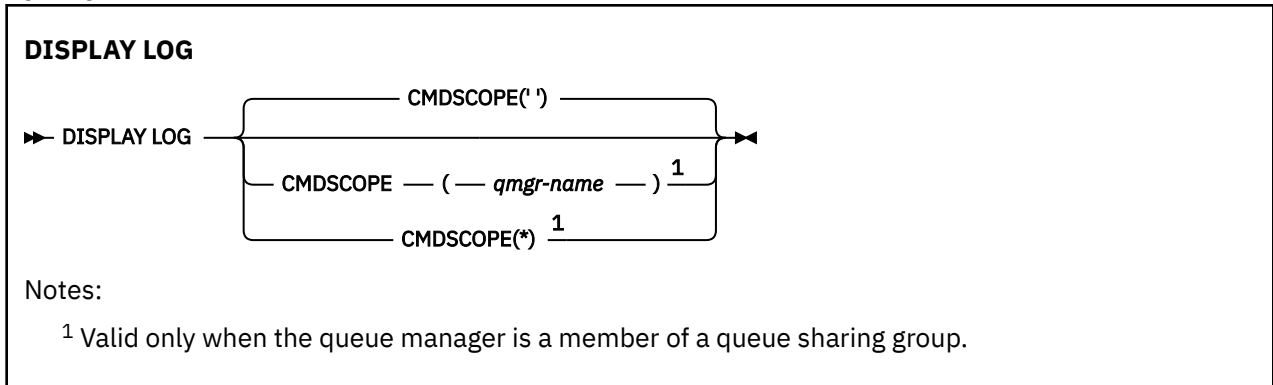
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [“Usage notes for DISPLAY LOG”](#) on page 746

- “Parameter descriptions for DISPLAY LOG” on page 746

**Synonym:** DIS LOG



## Usage notes for DISPLAY LOG

1. **DISPLAY LOG** returns a report that shows the initial log parameters, and the current values as changed by the **SET LOG** command:

- Whether log compression is active (COMPLOG).
- Whether writes to the active logs are made with zHyperWrite being enabled (ZHYWRITE)
- Length of time that an allowed archive read tape unit remains unused before it is deallocated (DEALLCT).
- Size of input buffer storage for active and archive log data sets (INBUFF).
- Size of output buffer storage for active and archive log data sets (OUTBUFF).
- Maximum number of dedicated tape units that can be set to read archive log tape volumes (MAXRTU).
- Maximum number of archive log volumes that can be recorded (MAXARCH).
- Maximum number of concurrent log offload tasks (MAXCNOFF)
- Whether archiving is on or off (OFFLOAD).
- Whether single or dual active logging is being used (TWOACTV).
- Whether single or dual archive logging is being used (TWOARCH).
- Whether single or dual BSDS is being used (TWOBSDS).
- Number of output buffers to be filled before they are written to the active log data sets (WRTHRSH).

It also returns a report about the status of the logs.

2. This command is issued internally by IBM MQ at the end of queue manager startup.

## Parameter descriptions for DISPLAY LOG

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## Multi **DISPLAY LSSTATUS (display listener status) on Multiplatforms**

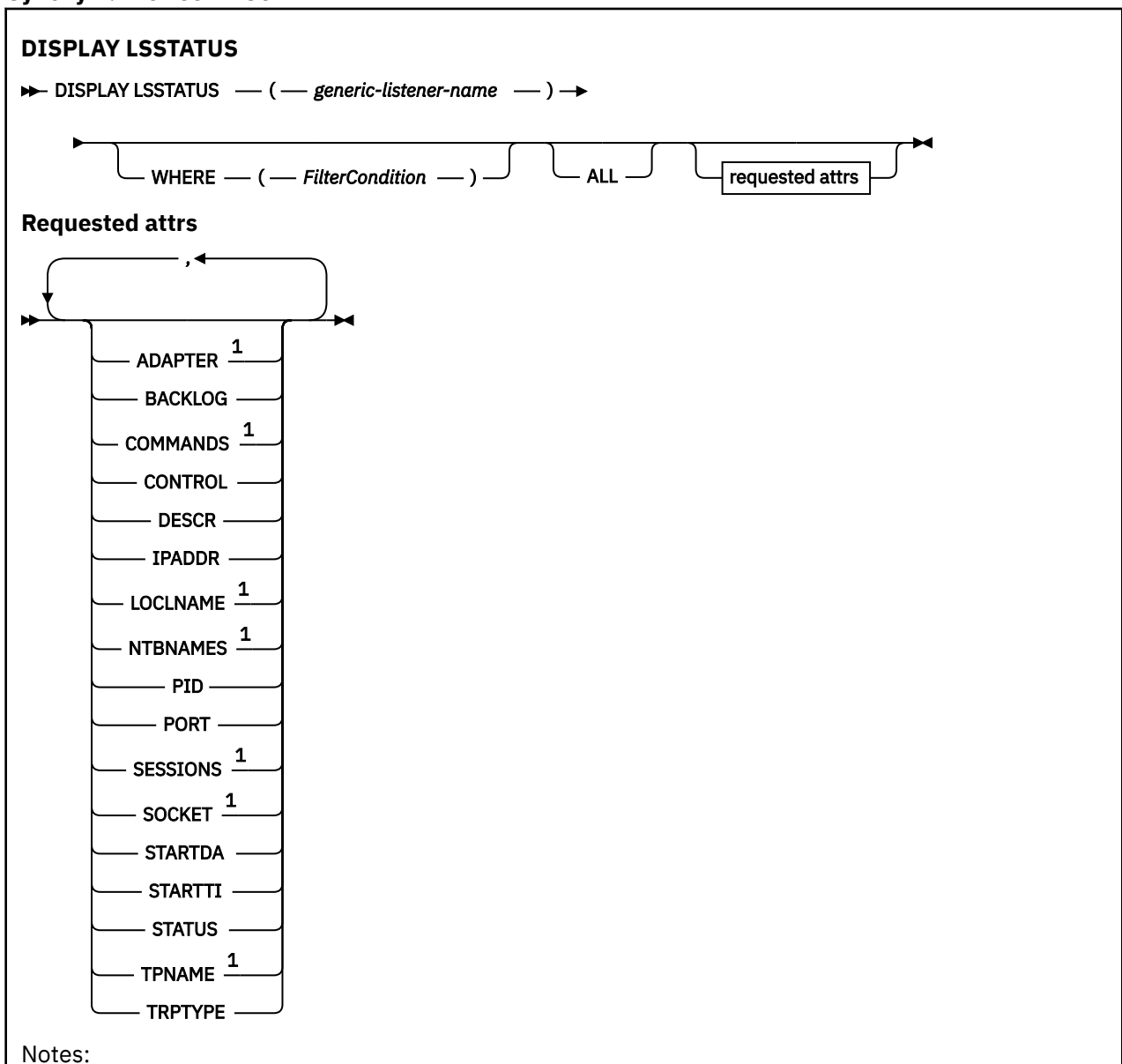
Use the MQSC command DISPLAY LSSTATUS to display status information for one or more listeners.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [“Keyword and parameter descriptions for DISPLAY LSSTATUS” on page 748](#)
- [“Requested parameters” on page 749](#)

**Synonym:** DIS LSSTATUS



<sup>1</sup> Valid only on Windows.

## Keyword and parameter descriptions for DISPLAY LSSTATUS

You must specify a listener for which you want to display status information. You can specify a listener by using either a specific listener name or a generic listener name. By using a generic listener name, you can display either:

- Status information for all listener definitions, by using a single asterisk (\*), or
- Status information for one or more listeners that match the specified name.

### ( *generic-listener-name* )

The name of the listener definition for which status information is to be displayed. A single asterisk (\*) specifies that information for all connection identifiers is to be displayed. A character string with an asterisk at the end matches all listeners with the string followed by zero or more characters.

### WHERE

Specify a filter condition to display information for those listeners that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Any parameter that can be used to display attributes for this DISPLAY command.

#### **operator**

This is used to determine whether a listener satisfies the filter value on the given filter keyword. The operators are:

#### **LT**

Less than

#### **GT**

Greater than

#### **EQ**

Equal to

#### **NE**

Not equal to

#### **LE**

Less than or equal to

#### **GE**

Greater than or equal to

#### **LK**

Matches a generic string that you provide as a *filter-value*

#### **NL**

Does not match a generic string that you provide as a *filter-value*

#### **filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.
- A generic value. This is a character string with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**ALL**

Display all the status information for each specified listener. This is the default if you do not specify a generic name, and do not request any specific parameters.

**Requested parameters**

Specify one or more attributes that define the data to be displayed. The attributes can be specified in any order. Do not specify the same attribute more than once.

**ADAPTER**

The adapter number on which NetBIOS listens.

**BACKLOG**

The number of concurrent connection requests that the listener supports.

**CONTROL**

How the listener is to be started and stopped:

**MANUAL**

The listener is not to be started automatically or stopped automatically. It is to be controlled by use of the START LISTENER and STOP LISTENER commands.

**QMGR**

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

**STARTONLY**

The listener is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**DESCR**

Descriptive comment.

**IPADDR**

The listener's IP address.

**LOCLNAME**

The NetBIOS local name that the listener uses.

**NTBNAMES**

The number of names that the listener can use.

**PID**

The operating system process identifier associated with the listener.

**PORT**

The port number for TCP/IP.

**SESSIONS**

The number of sessions that the listener can use.

**SOCKET**

SPX socket.

**STARTDA**

The date on which the listener was started.

**STARTTI**

The time at which the listener was started.

**STATUS**

The current status of the listener. It can be one of:

**RUNNING**

The listener is running.

**STARTING**

The listener is in the process of initializing.

## STOPPING

The listener is stopping.

## TPNAME

The LU6.2 transaction program name.

## TRPTYPE

Transport type.

For more information on these parameters, see [“DEFINE LISTENER \(define a new listener\) on Multiplatforms”](#) on page 530.

## **DISPLAY MAXSMGS (display maximum messages setting) on z/OS**

Use the MQSC command DISPLAY MAXSMGS to see the maximum number of messages that a task can get or put within a single unit of recovery.

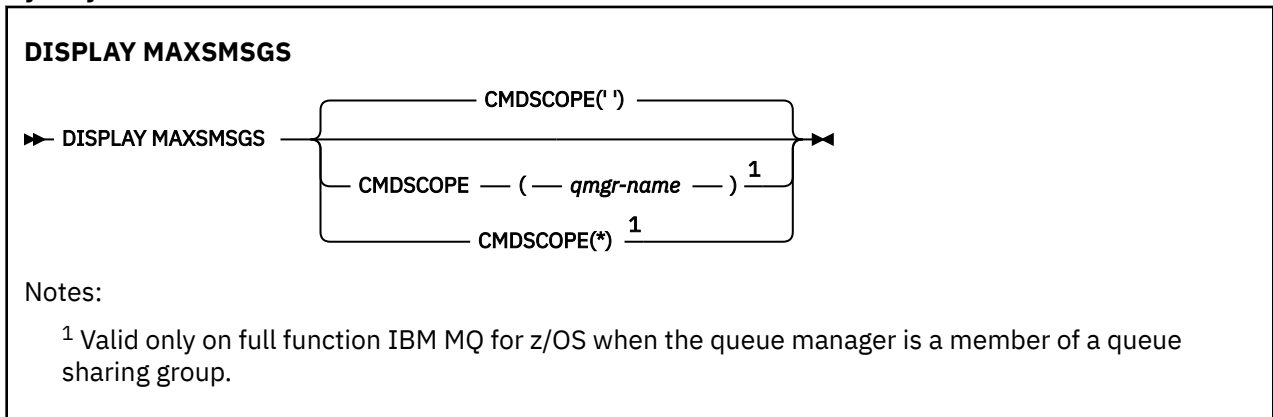
## Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes”](#) on page 750
- [“Parameter descriptions for DISPLAY MAXSMGS”](#) on page 750

**Synonym:** DIS MAXSM



## Usage notes

This command is valid only on z/OS and is retained for compatibility with earlier releases, although it can no longer be issued from the CSQINP1 initialization data set. You should use the MAXUMSGS parameter of the DISPLAY QMGR command instead.

## Parameter descriptions for DISPLAY MAXSMGS

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*


The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## DISPLAY NAMELIST (display a list of names)

Use the MQSC command DISPLAY NAMELIST to display the names in a namelist.

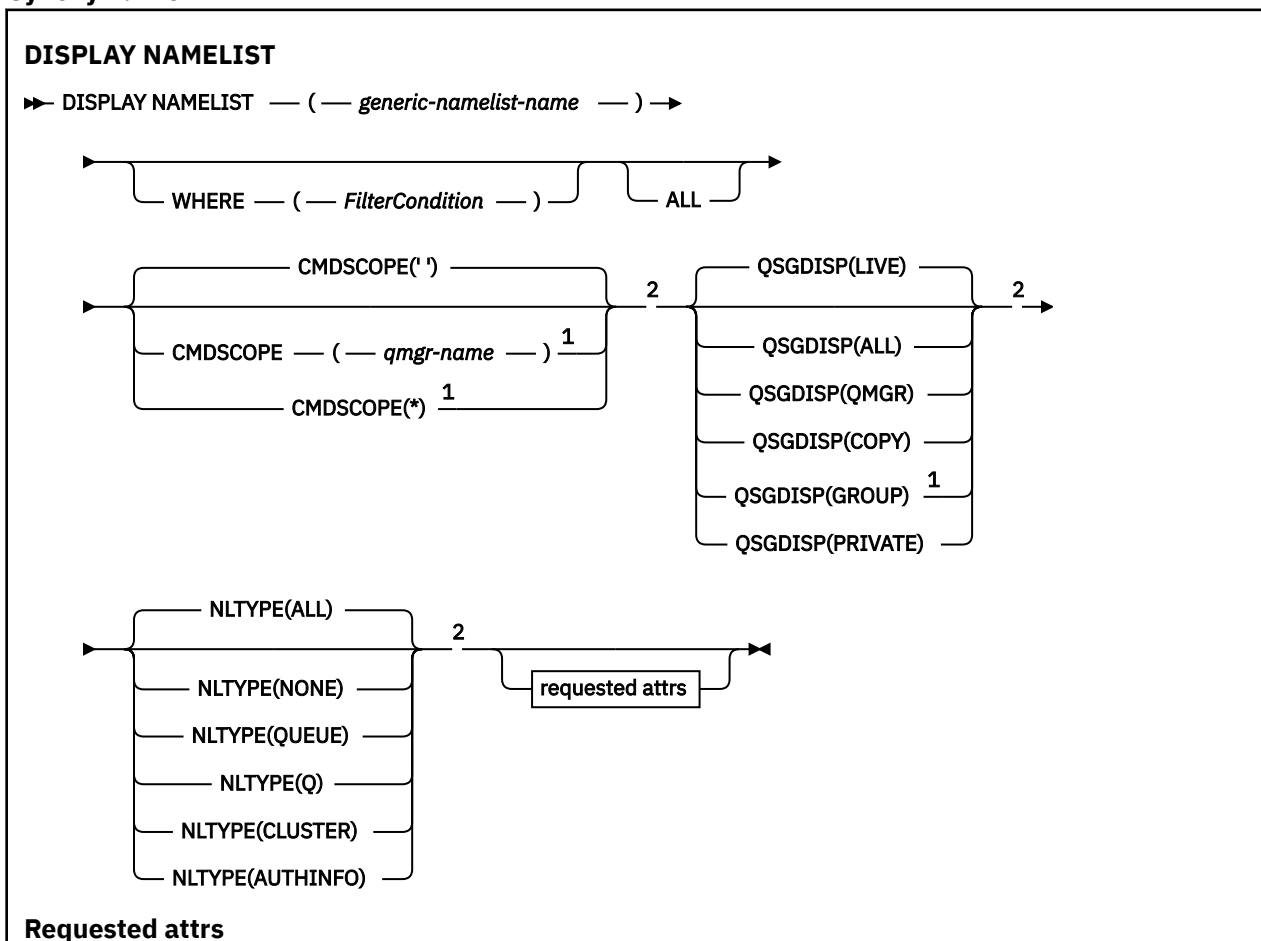
### Using MQSC commands

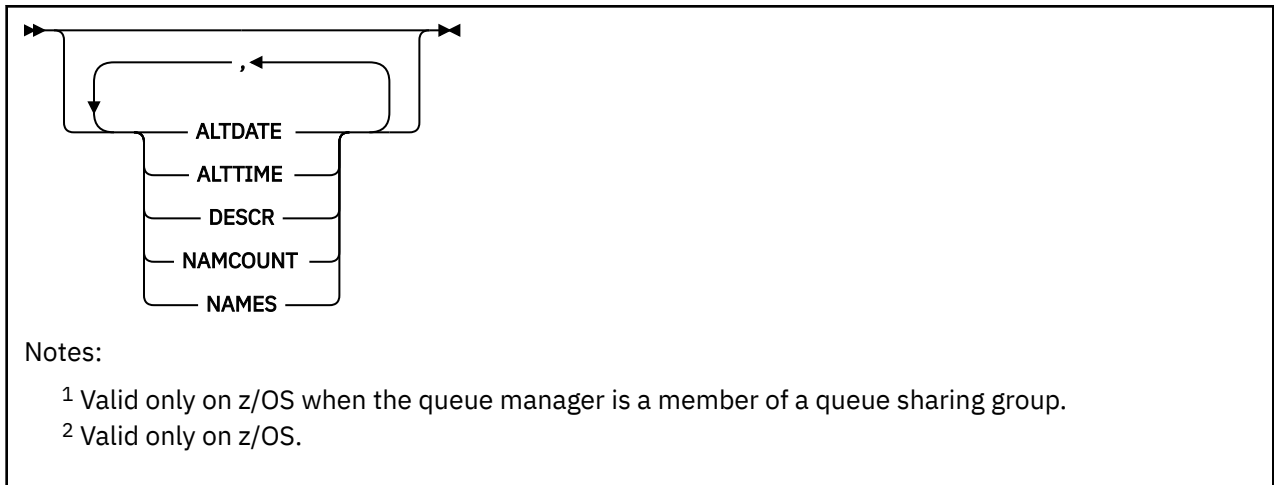
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY NAMELIST” on page 752](#)
- [“Requested parameters” on page 755](#)

**Synonym:** DIS NL





## Parameter descriptions for DISPLAY NAMELIST

You must specify the name of the namelist definition you want to display. This can be a specific namelist name or a generic namelist name. By using a generic namelist name, you can display either:

- All namelist definitions
- One or more namelists that match the specified name

### ( *generic-namelist-name* )

The name of the namelist definition to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk (\*) matches all namelists with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all namelists.

### WHERE

Specify a filter condition to display only those namelists that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### filter-keyword

Almost any parameter that can be used to display attributes for this DISPLAY command. However, you cannot use the CMDSCOPE or QSGDISP parameters as filter keywords. You cannot use NLTYPE as a filter keyword if you also use it to select namelists.

#### operator

This is used to determine whether a namelist satisfies the filter value on the given filter keyword. The operators are:

##### LT

Less than

##### GT

Greater than

##### EQ

Equal to

##### NE

Not equal to

##### LE

Less than or equal to

##### GE

Greater than or equal to

##### LK

Matches a generic string that you provide as a *filter-value*

##### NL

Does not match a generic string that you provide as a *filter-value*



**CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which contain the specified item.

**EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not contain the specified item.

**CTG**

Contains an item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which match the generic string.

**EXG**

Does not contain any item which matches a generic string that you provide as a *filter-value*. If the *filter-keyword* is a list, you can use this to display objects the attributes of which do not match the generic string.

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value NONE on the NLTYPE parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.


You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. The value can be explicit or, if it is a character value, it can be explicit or generic. If it is explicit, use CT or EX as the operator. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed. If it is generic, use CTG or EXG as the operator. If ABC\* is specified with the operator CTG, all items where one of the attribute values begins with ABC are listed.

**ALL**

Specify this to display all the parameters. If this parameter is specified, any parameters that are requested specifically have no effect; all the parameters are displayed.

This is the default if you do not specify a generic name, and do not request any specific parameters.

 On z/OS this is also the default if you specify a filter condition using the WHERE parameter, but on other platforms only requested attributes are displayed.

 **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## **QSGDISP**

Specifies the disposition of the objects for which information is to be displayed. Values are:

### **LIVE**

This is the default value and displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

### **ALL**

Displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP).

If QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

In a shared queue manager environment, use

```
DISPLAY NAMELIST(name) CMDSCOPE(*) QSGDISP(ALL)
```

to list ALL objects matching

```
name
```

in the queue sharing group without duplicating those in the shared repository.

### **COPY**

Display information only for objects defined with QSGDISP(COPY).

### **GROUP**

Display information only for objects defined with QSGDISP(GROUP). This is allowed only if there is a shared queue manager environment.

### **PRIVATE**

Display information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY). Note that QSGDISP(PRIVATE) displays the same information as QSGDISP(LIVE).

### **QMGR**

Display information only for objects defined with QSGDISP(QMGR).

QSGDISP displays one of the following values:

### **QMGR**

The object was defined with QSGDISP(QMGR).

### **GROUP**

The object was defined with QSGDISP(GROUP).

### **COPY**

The object was defined with QSGDISP(COPY).

You cannot use QSGDISP as a filter keyword.

## **NLTYPE**

Indicates the type of namelist to be displayed.

This parameter is valid only on z/OS.

**ALL**

Displays namelists of all types. This is the default.

**NONE**

Displays namelists of type NONE.

**QUEUE or Q**

Displays namelists that hold lists of queue names.

**CLUSTER**

Displays namelists that are associated with clustering.

**AUTHINFO**

Displays namelists that contain lists of authentication information object names.

## Requested parameters

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

The default, if no parameters are specified (and the ALL parameter is not specified) is that the object names, and, on z/OS, their NLTYPEs and QSGDISP are displayed.

**ALTDATE**

The date on which the definition was last altered, in the form yyyy-mm-dd

**ALTTIME**

The time at which the definition was last altered, in the form hh.mm.ss

**DESCR**

Description

**NAMCOUNT**

Number of names in the list

**NAMES**

List of names

See [“DEFINE NAMELIST \(define a list of names\)”](#) on page 536 for more information about the individual parameters.

## **DISPLAY POLICY (display a security policy) on Multiplatforms**

Use the MQSC command **DISPLAY POLICY** to display a security policy.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY POLICY”](#) on page 755

**DISPLAY POLICY**

► **DISPLAY POLICY** — ( — *policy-name* — ) ►

### Parameter descriptions for DISPLAY POLICY

**(*policy-name*)**

Specifies the policy name to be displayed.

The name of the policy to display is the same as the name of the queue that the policy controls. You can specify an asterisk to display all policy names.

**Note:** *policy-name* does not support wildcard characters to return multiple policies.

## Display policy behavior with specific policy names

When executing a **DISPLAY POLICY** command for a specific policy, for example `DISPLAY POLICY(Queue.1)`, a policy object is always returned even if one does not exist. When a policy object does not exist, the policy object returned is a default policy object that specifies plain text protection, that is, no signing or encryption of message data.


To view policy objects that exist, a `DISPLAY POLICY(*)` command must be run. This command returns all policy objects that exist.

### Related reference


[“SET POLICY \(set security policy\) on Multiplatforms” on page 938](#)

Use the MQSC command `SET POLICY` to set a security policy.

[“setmqspl \(set security policy\)” on page 216](#)

Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy.  On z/OS you use the command with the CSQOUTIL utility.

[“dspmqspl \(display security policy\)” on page 99](#)

Use the **dspmqspl** command to display a list of all policies and details of a named policy.  On z/OS you use the command with the CSQOUTIL utility.

### Related information


[Managing security policies in AMS](#)

## DISPLAY PROCESS (display process information)

Use the MQSC command `DISPLAY PROCESS` to display the attributes of one or more IBM MQ processes.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

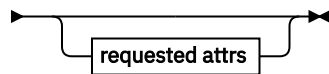
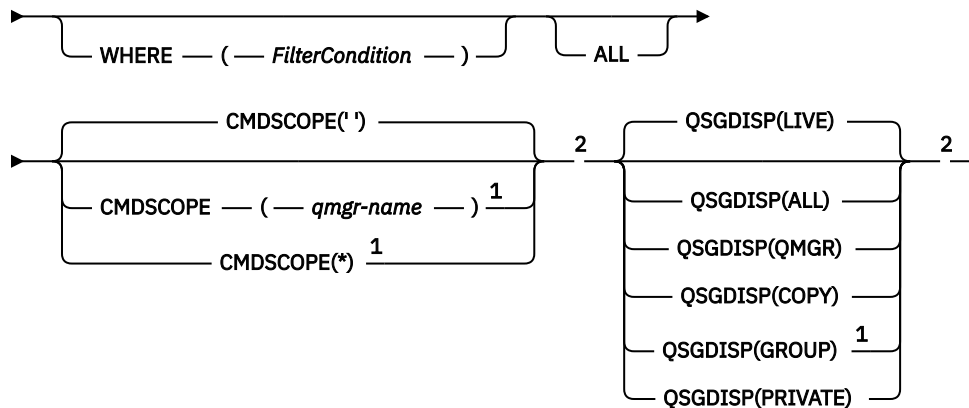
 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY PROCESS” on page 757](#)
- [“Requested parameters” on page 760](#)

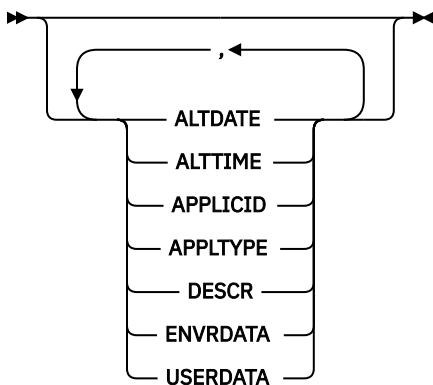
**Synonym:** DIS PRO

## DISPLAY PROCESS

►► DISPLAY PROCESS — ( — *generic-process-name* — ) ►►



### Requested attrs



Notes:

- <sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>2</sup> Valid only on z/OS.

## Parameter descriptions for DISPLAY PROCESS

You must specify the name of the process you want to display. This can be a specific process name or a generic process name. By using a generic process name, you can display either:

- All process definitions
- One or more processes that match the specified name

### (*generic-process-name*)


The name of the process definition to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk (\*) matches all processes with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all processes. The names must all be defined to the local queue manager.

### WHERE

Specify a filter condition to display only those process definitions that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

### filter-keyword

Almost any parameter that can be used to display attributes for this DISPLAY command.

 However, on z/OS, you cannot use the CMDSCOPE or QSGDISP parameters as filter keywords.

### operator

This is used to determine whether a process definition satisfies the filter value on the given filter keyword. The operators are:

#### LT

Less than

#### GT

Greater than

#### EQ

Equal to

#### NE

Not equal to

#### LE

Less than or equal to

#### GE

Greater than or equal to

#### LK

Matches a generic string that you provide as a *filter-value*

#### NL

Does not match a generic string that you provide as a *filter-value*

### filter-value

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value DEF on the APPLTYPE parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

### ALL

Specify this to display all the parameters. If this parameter is specified, any parameters that are requested specifically have no effect; all parameters are still displayed.

On the following platforms, this is the default if you do not specify a generic name and do not request any specific parameters:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

**z/OS** On z/OS this is also the default if you specify a filter condition using the WHERE parameter, but on other platforms only requested attributes are displayed.

### **z/OS** **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

''

The command runs on the queue manager on which it was entered. This is the default value.

#### ***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

### **z/OS** **QSGDISP**

Specifies the disposition of the objects for which information is to be displayed. Values are:

#### **LIVE**

This is the default value and displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

#### **ALL**

Displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP).

If QSGDISP(LIVE) is specified or defaulted, or if QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **COPY**

Display information only for objects defined with QSGDISP(COPY).

#### **GROUP**

Display information only for objects defined with QSGDISP(GROUP). This is allowed only if there is a shared queue manager environment.

#### **PRIVATE**

Display information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY). Note that QSGDISP(PRIVATE) displays the same information as QSGDISP(LIVE).

#### **QMGR**

Display information only for objects defined with QSGDISP(QMGR).

QSGDISP displays one of the following values:

#### **QMGR**

The object was defined with QSGDISP(QMGR).

#### **GROUP**

The object was defined with QSGDISP(GROUP).

#### **COPY**



The object was defined with QSGDISP(COPY).

You cannot use QSGDISP as a filter keyword.

## Requested parameters

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

The default, if no parameters are specified and the **ALL** parameter is not specified, is that:

-  On Multiplatforms, that is, on platforms other than z/OS, the object names are displayed.
-  On z/OS only, the object names and QSGDISP are displayed.

### ALTDATE

The date on which the definition was last altered, in the form yyyy-mm-dd

### ALTTIME

The time at which the definition was last altered, in the form hh.mm.ss

### APPLICID

Application identifier

### APPLTYPE

Application type. In addition to the values listed for this parameter in “Parameter descriptions for DEFINE PROCESS” on page 541, the value SYSTEM can be displayed. This indicates that the application type is a queue manager.

### DESCR

Description

### ENVRDATA

Environment data

### USERDATA

User data


See “[DEFINE PROCESS \(create a new process definition\)](#)” on page 540 for more information about individual parameters.

## DISPLAY PUBSUB (display publish/subscribe status information)

Use the MQSC command DISPLAY PUBSUB to display publish/subscribe status information for a queue manager.

## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

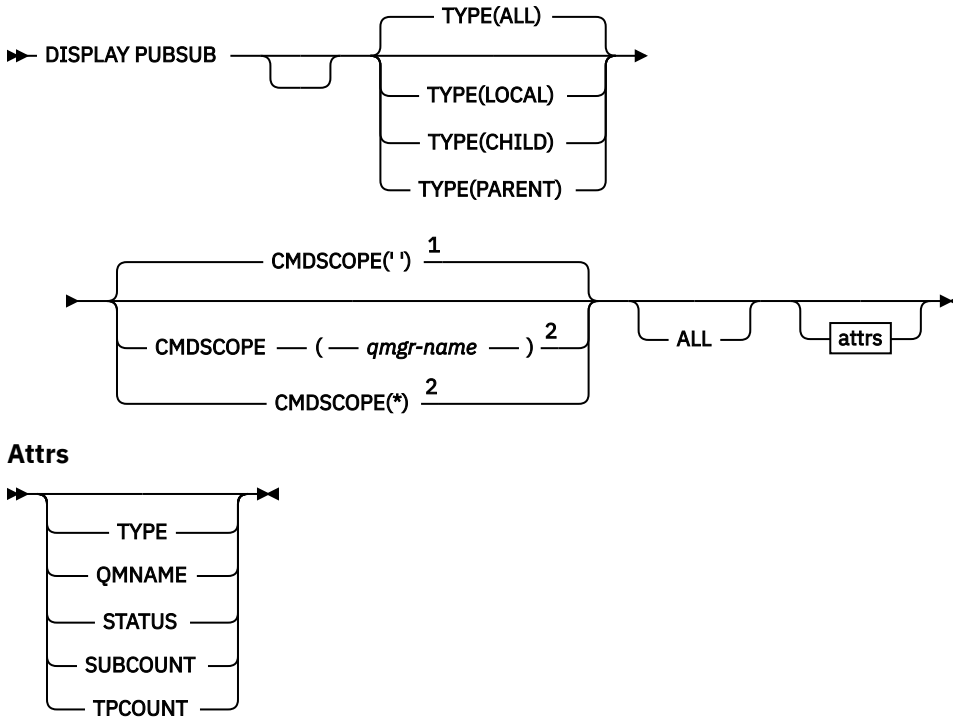
 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY PUBSUB” on page 761](#)
- [“Returned parameters” on page 762](#)

**Synonym:** None



## DISPLAY PUBSUB



### Notes:

<sup>1</sup> Valid only on z/OS.

<sup>2</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

## Parameter descriptions for DISPLAY PUBSUB

### TYPE

The type of publish/subscribe connections.

### ALL

Display the publish/subscribe status for this queue manager and for parent and child hierarchical connections.

### CHILD

Display the publish/subscribe status for child connections.

### LOCAL

Display the publish/subscribe status for this queue manager.

### PARENT

Display the publish/subscribe status for the parent connection.

### z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

## Returned parameters

A group of parameters is returned, containing the attributes TYPE, QMNAME, STATUS, SUBCOUNT, and TPCOUNT. This group is returned for the current queue manager if you set TYPE to LOCAL or ALL, for the parent queue manager if you set TYPE to PARENT or ALL, and for each child queue manager if you set TYPE to CHILD or ALL.

### TYPE

#### CHILD

A child connection.

#### LOCAL

Information for this queue manager.

#### PARENT


The parent connection.

### QMNAME

The name of the current queue manager or the remote queue manager connected as a parent or a child.

### STATUS

The status of the publish/subscribe engine or the hierarchical connection. The publish/subscribe engine is initializing and is not yet operational. If the queue manager is a member of a cluster (has at least one CLUSRCVR defined), it remains in this state until the cluster cache is available.

 On IBM MQ for z/OS, this requires that the Channel Initiator is running.

When TYPE is CHILD, the following values can be returned:

#### ACTIVE

The connection with the child queue manager is active.

#### ERROR

This queue manager is unable to initialize a connection with the child queue manager because of a configuration error. A message is produced in the queue manager logs to indicate the specific error. If you receive error message AMQ5821 or on z/OS systems CSQT821E, possible causes include:

- Transmit queue is full.
- Transmit queue put is disabled.

If you receive error message AMQ5814 or on z/OS systems CSQT814E, take the following actions:

- Check that the child queue manager is correctly specified.
- Ensure that broker is able to resolve the queue manager name of the child broker.

To resolve the queue manager name, at least one of the following resources must be configured:

- A transmission queue with the same name as the child queue manager name.
- A queue manager alias definition with the same name as the child queue manager name.
- A cluster with the child queue manager a member of the same cluster as this queue manager.

- A cluster queue manager alias definition with the same name as the child queue manager name.
- A default transmission queue.

After you have set up the configuration correctly, modify the child queue manager name to blank. Then set with the child queue manager name.

#### **STARTING**

Another queue manager is attempting to request that this queue manager become its parent.

If the child status remains in STARTING without progressing to ACTIVE, take the following actions:

- Check that the sender channel to child queue manager is running
- Check that the receiver channel from child queue manager is running

#### **STOPPING**

The queue manager is disconnecting.

If the child status remains in STOPPING, take the following actions:

- Check that the sender channel to child queue manager is running
- Check that the receiver channel from child queue manager is running

When TYPE is LOCAL, the following values can be returned:

#### **ACTIVE**

The publish/subscribe engine and the queued publish/subscribe interface are running. It is therefore possible to publish or subscribe using the application programming interface and the queues that are monitored by the queued publish/subscribe interface.

#### **COMPAT**

The publish/subscribe engine is running. It is therefore possible to publish or subscribe by using the application programming interface. The queued publish/subscribe interface is not running. Therefore, any message that is put to the queues that are monitored by the queued publish/subscribe interface are not acted upon by IBM MQ.

#### **ERROR**

The publish/subscribe engine has failed. Check your error logs to determine the reason for the failure.


#### **INACTIVE**

The publish/subscribe engine and the queued publish/subscribe interface are not running. It is therefore not possible to publish or subscribe using the application programming interface. Any publish/subscribe messages that are put to the queues that are monitored by the queued publish/subscribe interface are not acted upon by IBM MQ.

If inactive and you want to start the publish/subscribe engine use the command **ALTER QMGR PSMODE(ENABLED)**.

#### **STARTING**

The publish/subscribe engine is initializing and is not yet operational. If the queue manager is a member of a cluster, that is, it has at least one CLUSRCVR defined, it remains in this state until the cluster cache is available.

 On IBM MQ for z/OS, this requires that the Channel Initiator is running.

#### **STOPPING**

The publish/subscribe engine is stopping.


When TYPE is PARENT, the following values can be returned:

#### **ACTIVE**


The connection with the parent queue manager is active.

#### **ERROR**

This queue manager is unable to initialize a connection with the parent queue manager because of a configuration error. A message is produced in the queue manager logs to indicate the specific

error. If you receive error message AMQ5821,  or on z/OS systems CSQT821E, possible causes include:

- Transmit queue is full.
- Transmit queue put is disabled.

If you receive error message AMQ5814,  or error message CSQT814E on z/OS systems, take the following actions:

- Check that the parent queue manager is correctly specified.
- Ensure that broker is able to resolve the queue manager name of the parent broker.

To resolve the queue manager name, at least one of the following resources must be configured:

- A transmission queue with the same name as the parent queue manager name.
- A queue manager alias definition with the same name as the parent queue manager name.
- A cluster with the parent queue manager a member of the same cluster as this queue manager.
- A cluster queue manager alias definition with the same name as the parent queue manager name.
- A default transmission queue.

After you have set up the configuration correctly, modify the parent queue manager name to blank. Then set with the parent queue manager name.

#### **REFUSED**

The connection has been refused by the parent queue manager. This might be caused by the following:

- The parent queue manager already has a child queue manager with the same name as this queue manager.
- The parent queue manager has used the command RESET QMGR TYPE(PUBSUB) CHILD to remove this queue manager as one of its children.

#### **STARTING**

The queue manager is attempting to request that another queue manager become its parent.

If the parent status remains in STARTING without progressing to ACTIVE, take the following actions:

- Check that the sender channel to parent queue manager is running
- Check that the receiver channel from parent queue manager is running

#### **STOPPING**

The queue manager is disconnecting from its parent.

If the parent status remains in STOPPING, take the following actions:

- Check that the sender channel to parent queue manager is running
- Check that the receiver channel from parent queue manager is running

#### **SUBCOUNT**

When TYPE is LOCAL, the total number of subscriptions against the local tree is returned. When TYPE is CHILD or PARENT, queue manager relations are not inquired and the value NONE is returned.

#### **TPCOUNT**

When TYPE is LOCAL, the total number of topic nodes in the local tree is returned. When TYPE is CHILD or PARENT, queue manager relations are not inquired and the value NONE is returned.

## DISPLAY QMGR (display queue manager settings)

Use the MQSC command **DISPLAY QMGR** to display the queue manager parameters for this queue manager.

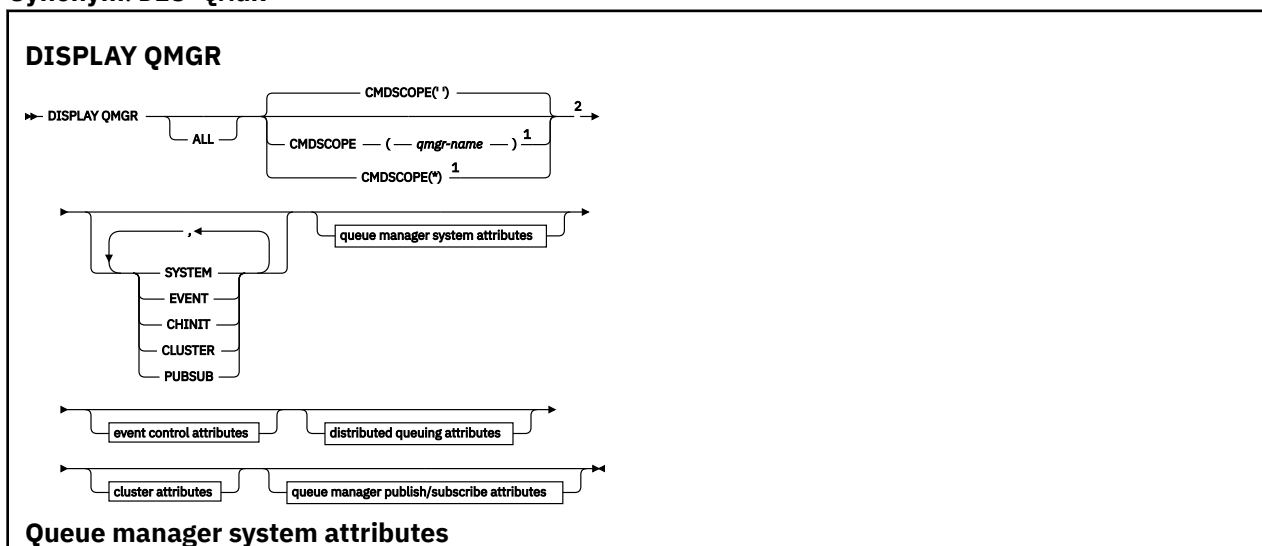
### Using MQSC commands

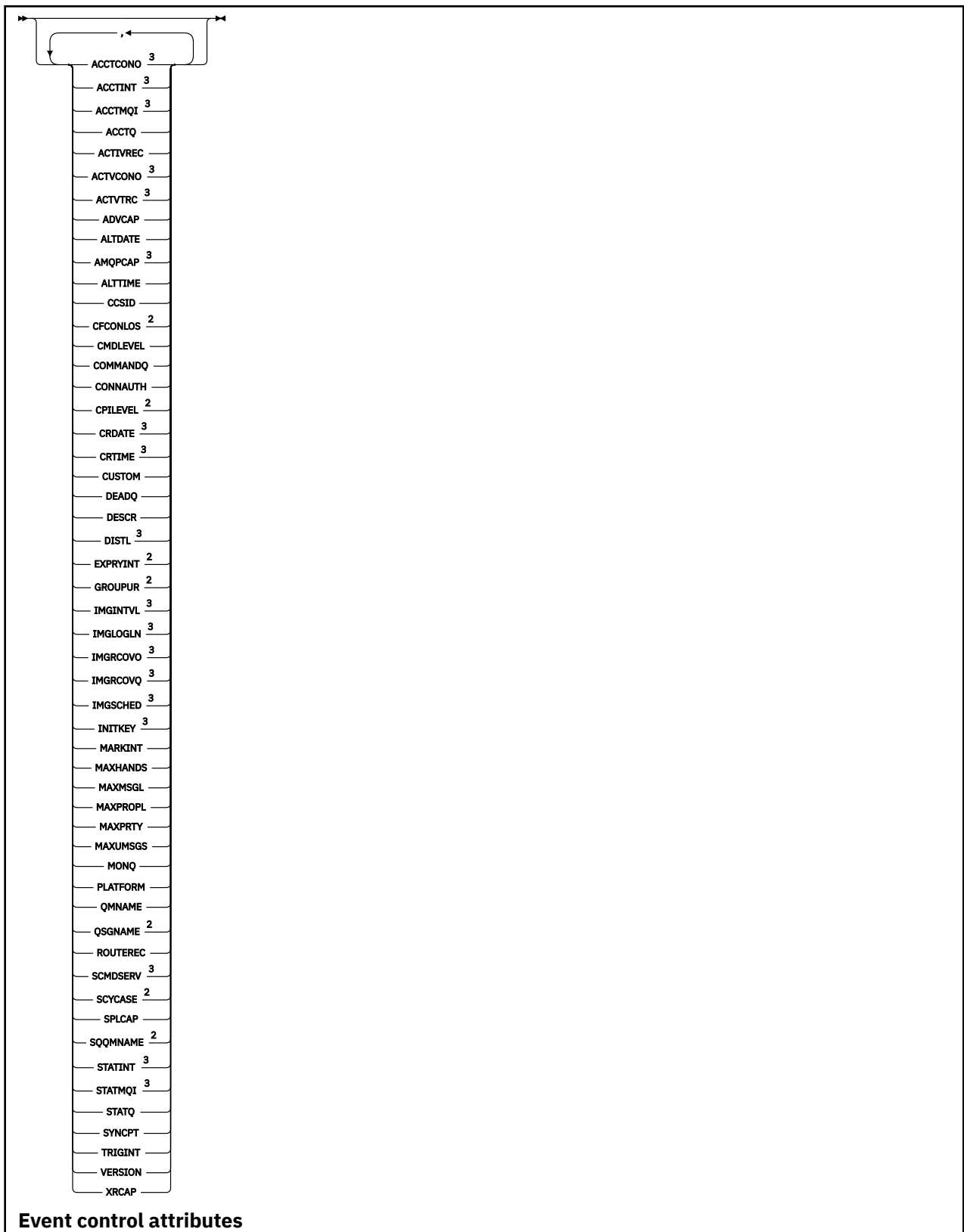
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

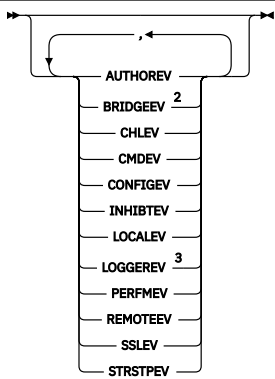
**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY QMGR” on page 769](#)
- [“Requested parameters” on page 770](#)

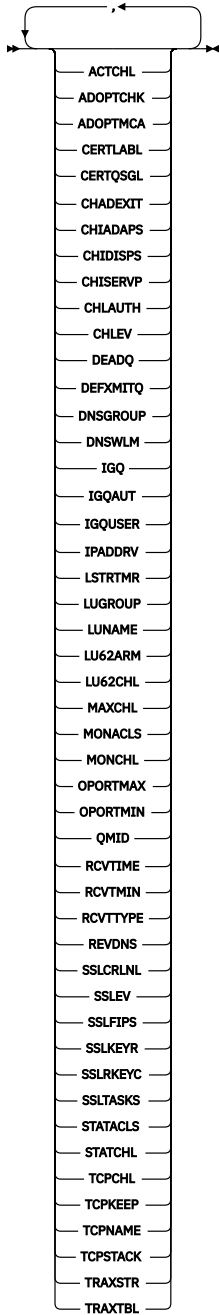
**Synonym: DIS QMGR**



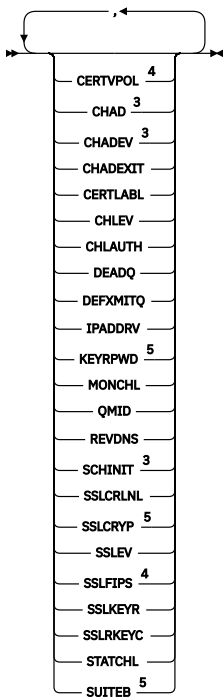




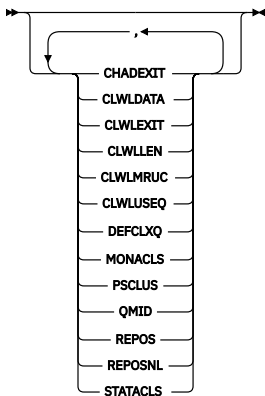
**Distributed queuing attributes for z/OS**



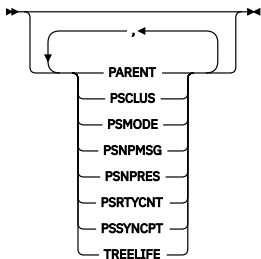
**Distributed queuing attributes for other platforms**



### Cluster attributes



### Queue manager publish/subscribe attributes



### Notes:


- 1 Valid only on z/OS when the queue manager is a member of a queue sharing group.
- 2 Valid only on z/OS.
- 3 Not valid on z/OS.
- 4 Not valid on IBM i.
- 5 Valid only on AIX, Linux, and Windows.



## Parameter descriptions for DISPLAY QMGR

### ALL

Specify this parameter to display all the parameters. If this parameter is specified, any parameters that are requested specifically are ineffective; all parameters are still displayed.

 On [Multiplatforms](#), this parameter is the default if you do not request any specific parameters.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This command is the default value.

#### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of running this command is the same as entering the command on every queue manager in the queue sharing group.

### SYSTEM

Specify this parameter to display the set of queue manager system attributes that are available in the Queue manager system attrs list. See [“Requested parameters” on page 770](#) for information about these parameters.

If you specify this parameter, any request you make to display individual parameters within this set is ineffective.

### EVENT

Specify this parameter to display the set of event control attributes that are available in the Event control attrs list. See [“Requested parameters” on page 770](#) for information about these parameters.

If you specify this parameter, any request you make to display individual parameters within this set is ineffective.

### CHINIT

Specify this parameter to display the set of attributes relating to distributed queuing that are available in the Distributed queuing attrs list. You can also specify DQM to display the same set of attributes. See [“Requested parameters” on page 770](#) for information about these parameters.

If you specify this parameter, any request you make to display individual parameters within this set is ineffective.

### CLUSTER

Specify this parameter to display the set of attributes relating to clustering that are available in the Cluster attrs list. See [“Requested parameters” on page 770](#) for information about these parameters.

If you specify this parameter, any request you make to display individual parameters within this set is ineffective.

### PUBSUB

Specify this parameter to display the set of attributes relating to publish/subscribe that are available in the Queue manager pub/sub attrs list. See [“Requested parameters” on page 770](#) for information about these parameters.

If you specify this parameter, any request you make to display individual parameters within this set is ineffective.

## Requested parameters

**Note:** If no parameters are specified (and the **ALL** parameter is not specified or defaulted), the queue manager name is returned.

You can request the following information for the queue manager:

### Multi ACCTCONO

Whether the settings of the **ACCTQMQUI** and **ACCTQ** queue manager parameters can be overridden. This parameter is valid only on [Multiplatforms](#).

### Multi ACCTINT

The interval at which intermediate accounting records are written. This parameter is valid only on [Multiplatforms](#).

### Multi ACCTMQI

Whether accounting information is to be collected for MQI data. This parameter is valid only on [Multiplatforms](#).

### ACCTQ

Whether accounting data collection is to be enabled for queues.

### z/OS ACTCHL

The maximum number of channels that can be active at any time.

This parameter is valid only on z/OS.

### ACTIVREC

Whether activity reports are to be generated if requested in the message.

### Multi ACTVCONO

Whether the settings of the **ACTVTRC** queue manager parameter can be overridden. This parameter is valid only on [Multiplatforms](#).

### Multi ACTVTRC

Whether IBM MQ MQI application activity tracing information is to be collected. See [Setting ACTVTRC to control collection of activity trace information](#). This parameter is valid only on [Multiplatforms](#).

### z/OS ADOPTCHK

Which elements are checked to determine whether an MCA is adopted when a new inbound channel is detected with the same name as an already active MCA.

This parameter is valid only on z/OS.

### z/OS ADOPTMCA

Whether an orphaned MCA instance is to be restarted when a new inbound channel request matching the **ADOPTCHK** parameters is detected.

This parameter is valid only on z/OS.

### MQ Adv. ADVCAP

Whether IBM MQ Advanced extended capabilities are available for a queue manager.

**z/OS** On z/OS, the queue manager sets the value to be ENABLED, only if the value of **QMGRPROD** is ADVANCEDVUE. For any other value of **QMGRPROD**, or if **QMGRPROD** is not set, the queue manager sets the value to DISABLED. If **ADVCAP** is ENABLED you must be entitled to IBM MQ Advanced for z/OS Value Unit Edition (VUE). See [“START QMGR \(start queue manager\) on z/OS” on page 953 and Installing IBM MQ Advanced for z/OS Value Unit Edition](#) for more information.

**Multi** On other platforms, the queue manager sets the value to be ENABLED, only if you have installed Managed File Transfer, XR, Advanced Message Security or RDQM. If you have not installed Managed File Transfer, XR, Advanced Message Security or RDQM, **ADVCAP** is set to DISABLED. If **ADVCAP** is ENABLED, you must be entitled to IBM MQ Advanced. The list of installable components that enable **ADVCAP** might change in future releases. For more information, see [IBM MQ components and features](#) and [Installing IBM MQ Advanced for Multiplatforms](#).

#### **ALTDATE**

The date on which the definition was last altered, in the form *yyyy-mm-dd*.

#### **ALTTIME**

The time at which the definition was last altered, in the form *hh.mm.ss*.

#### **AMQPCAP**

Whether AMQP capabilities are available for a queue manager.

#### **AUTHOREV**

Whether authorization events are generated.

#### **z/OS** **BRIDGEEV**

On z/OS only, whether IMS bridge events are generated.

#### **CCSID**

Coded character set identifier. This parameter applies to all character string fields defined by the application programming interface (API), including the names of objects, and the creation date and time of each queue. It does not apply to application data carried as the text of messages.

#### **CERTLABL**

Specifies the certificate label that this queue manager used.

#### **z/OS** **CERTQSG**

Specifies the queue sharing group (QSG) certificate label.

This parameter is valid only on z/OS.

#### **ALW** **CERTVPOL**

Specifies which TLS certificate validation policy is used to validate digital certificates received from remote partner systems. This attribute can be used to control how strictly the certificate chain validation conforms to industry security standards. For more information about certificate validation policies, see [Certificate validation policies in IBM MQ](#).

This parameter is valid only on AIX, Linux, and Windows.

#### **z/OS** **CFCONLOS**

Specifies the action to be taken when the queue manager loses connectivity to the administration structure, or any CF structure with **CFCONLOS** set to ASQMGR.

This parameter is valid only on z/OS.

#### **Multi** **CHAD**

Whether auto-definition of receiver and server-connection channels is enabled.

**z/OS** This parameter is not valid on z/OS.

#### **Multi** **CHADEV**

Whether auto-definition events are enabled.

**z/OS** This parameter is not valid on z/OS.

#### **CHADEXIT**

The name of the channel auto-definition exit.

#### **z/OS** **CHIADAPS**

The number of adapter subtasks to use to process IBM MQ calls.

This parameter is valid only on z/OS.

**z/OS CHIDISPS**

The number of dispatchers to use for the channel initiator.

This parameter is valid only on z/OS.

**CHISERV**

This field is reserved for IBM use only.

**CHLAUTH**

Whether channel authentication records are checked.

**CHLEV**

Whether channel events are generated.

**CLWLEXIT**

The name of the cluster workload exit.

**CLWLDATA**

The data passed to the cluster workload exit.

**Windows z/OS AIX CLWLEN**

The maximum number of bytes of message data that is passed to the cluster workload exit.

**Linux** This parameter is not valid on Linux.

**CLWLMRUC**

The maximum number of outbound cluster channels.

**CLWLUSEQ**

The behavior of MQPUTs for queues where **CLWLUSEQ** has a value of QMGR.

**CMDEV**

Whether command events are generated.

**CMDLEVEL**

Command level. This indicates the level of system control commands supported by the queue manager.

**COMMANDQ**

The name of the system-command input queue. Suitably authorized applications can put commands on this queue.

**CONFIGEV**

Whether configuration events are generated.

**CONNAUTH**

The name of an authentication information object that is used to provide the location of user ID and password authentication.

**CPILEVEL**

Reserved, this value has no significance.

**CRDATE**

The date on which the queue manager was created (in the form *yyyy-mm-dd*).

**CRTIME**

The time at which the queue manager was created (in the form *hh.mm.ss*).

**CUSTOM**

This attribute is reserved for the configuration of new features before separate attributes have been introduced. It can contain the values of zero or more attributes as pairs of attribute name and value in the form *NAME (VALUE)*.

**DEADQ**

The name of the queue to which messages are sent if they cannot be routed to their correct destination (the dead-letter queue or undelivered-message queue). The default is blanks.

For example, messages are put on this queue when:

- A message arrives at a queue manager, destined for a queue that is not yet defined on that queue manager
- A message arrives at a queue manager, but the queue for which it is destined cannot receive it because, possibly:
  - The queue is full
  - The queue is inhibited for puts
  - The sending node does not have authority to put the message on the queue
- An exception message must be generated, but the queue named is not known to that queue manager

**Note:** Messages that have passed their expiry time are not transferred to this queue when they are discarded.

If the dead-letter queue is not defined, or full, or unusable for some other reason, a message that would have been transferred to it by a message channel agent is retained instead on the transmission queue.

If a dead-letter queue or undelivered-message queue is not specified, all blanks are returned for this parameter.

## DEFCLXQ

The **DEFCLXQ** attribute controls which transmission queue is selected by default by cluster-sender channels to get messages from, to send the messages to cluster-receiver channels.

## SCTQ

All cluster-sender channels send messages from `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. The `correlID` of messages placed on the transmission queue identifies which cluster-sender channel the message is destined for.

SCTQ is set when a queue manager is defined. **DEFCLXQ** was not present.

## CHANNEL

Each cluster-sender channel sends messages from a different transmission queue. Each transmission queue is created as a permanent dynamic queue from the model queue `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`.

If the queue manager attribute, **DEFCLXQ**, is set to CHANNEL, the default configuration is changed to cluster-sender channels being associated with individual cluster transmission queues. The transmission queues are permanent-dynamic queues created from the model queue `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`. Each transmission queue is associated with one cluster-sender channel. As one cluster-sender channel services a cluster transmission queue, the transmission queue contains messages for only one queue manager in one cluster. You can configure clusters so that each queue manager in a cluster contains only one cluster queue. In this case, the message traffic from a queue manager to each cluster queue is transferred separately from messages to other queues.

## DEFXMITQ

Default transmission queue name. This parameter is the transmission queue on which messages, destined for a remote queue manager, are put if there is no other suitable transmission queue defined.

## DESCR

Description.

### Multi **DISTL**

Whether distribution lists are supported by the queue manager.

**z/OS** This parameter is not valid on z/OS.

### **z/OS** **DNSGROUP**

This parameter is no longer used. See [z/OS: WLM/DNS no longer supported](#). This parameter is valid only on z/OS

**z/OS DNSWLM**

This parameter is no longer used. See [z/OS: WLM/DNS no longer supported](#). This parameter is valid only on z/OS.

**z/OS EXPRYINT**

On z/OS only, the approximate interval between scans for expired messages.

**z/OS GROUPUR**

On z/OS only, whether XA client applications are allowed to connect to this queue manager with a GROUP unit of recovery disposition.

**IMGINTVL**

The target frequency with which the queue manager automatically writes media images.

**z/OS** This parameter is not valid on z/OS.

**IMGLOGLN**

The target amount of recovery log written by which the queue manager automatically writes media images.

**z/OS** This parameter is not valid on z/OS.

**IMGRCOVO**

Whether specified objects are recoverable from a media image, if linear logging is being used.

**z/OS** This parameter is not valid on z/OS.

**IMGRCOVQ**

Whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used.

**z/OS** This parameter is not valid on z/OS.

**IMGSCHED**

Whether the queue manager automatically writes media images.

**z/OS** This parameter is not valid on z/OS.

**z/OS IGQ**

On z/OS only, whether intra-group queuing is to be used.

**z/OS IGQAUT**

On z/OS only, displays the type of authority checking used by the intra-group queuing agent.

**z/OS IGQUSER**

On z/OS only, displays the user ID used by the intra-group queuing agent.

**INHIBTEV**

Whether inhibit events are generated.

**Multi V 9.3.0 V 9.3.0 INITKEY**

Initial key for the password protection system. The value appears as \*\*\*\*\* when set to a custom value. The value appears as ' ' when the default key is in use.

**z/OS** This parameter is not valid on z/OS.

**IPADDRV**

Whether to use an IPv4 or IPv6 IP address for a channel connection in ambiguous cases.

**V 9.3.0 ALW V 9.3.0 KEYRPWD**

The password for the TLS key repository. The value appears as \*\*\*\*\*.

This parameter is valid only on AIX, Linux, and Windows.

**LOCALEV**

Whether local error events are generated.

### Multi **LOGGEREV**

Whether recovery log events are generated. This parameter is valid only on [Multiplatforms](#).

### z/OS **LSTRTMR**

The time interval, in seconds, between attempts by IBM MQ to restart the listener after an APPC or TCP/IP failure.

This parameter is valid only on z/OS.

### z/OS **LUGROUP**

The generic LU name to be used by the LU 6.2 listener that handles inbound transmissions for the queue sharing group.

This parameter is valid only on z/OS.

### z/OS **LUNAME**

The name of the LU to use for outbound LU 6.2 transmissions.

This parameter is valid only on z/OS.

### z/OS **LU62ARM**

The suffix of the APPCPM member of SYS1.PARMLIB. This suffix nominates the LUADD for this channel initiator. When automatic restart manager (ARM) restarts the channel initiator, the z/OS command SET APPC= xx is issued.

This parameter is valid only on z/OS.

### z/OS **LU62CHL**

The maximum number of channels that can be current, or clients that can be connected, that use the LU 6.2 transmission protocol. If the value of LU62CHL is zero, the LU 6.2 transmission protocol is not used.

This parameter is valid only on z/OS.

### **MARKINT**

The mark browse interval in milliseconds.



**Attention:** This value should not be below the default of 5000.

### z/OS **MAXCHL**

The maximum number of channels that can be current (including server-connection channels with connected clients).

This parameter is valid only on z/OS.

### **MAXHANDS**

The maximum number of open handles that any one connection can have at any one time.

### **MAXMSGL**

The maximum message length that can be handled by the queue manager. Individual queues or channels might have a smaller maximum than the value of this parameter.

### **MAXPROPL (integer)**

The maximum length of property data in bytes that can be associated with a message.

### **MAXPRTY**

The maximum priority. This value is 9.

### **MAXUMSGS**

Maximum number of uncommitted messages within one sync point. The default value is 10000.

MAXUMSGS has no effect on MQ Telemetry. MQ Telemetry tries to batch requests to subscribe, unsubscribe, send, and receive messages from multiple clients into batches of work within a transaction.

**MONACLS**

Whether online monitoring data is to be collected for auto-defined cluster-sender channels, and, if so, the rate of data collection.

**MONCHL**

Whether online monitoring data is to be collected for channels, and, if so, the rate of data collection.

**MONQ**

Whether online monitoring data is to be collected for queues, and, if so, the rate of data collection.

**z/OS OPORTMAX**

The maximum value in the range of port numbers to be used when binding outgoing channels.

This parameter is valid only on z/OS.

**z/OS OPORTMIN**

The minimum value in the range of port numbers to be used when binding outgoing channels.

This parameter is valid only on z/OS.

**PARENT**

The name of the queue manager to which this queue manager is connected hierarchically as its child.

**PERFMEV**

Whether performance-related events are generated.

**PLATFORM**

The architecture of the platform on which the queue manager is running. The value of this parameter is:

- **z/OS** MVS (for z/OS platforms)
- NSK
- OS2
- OS400
- APPLIANCE
- UNIX
- WINDOWSNT

**PSCLUS**

Controls whether this queue manager participates in publish subscribe activity across any clusters in which it is a member. No clustered topic objects can exist in any cluster when modifying from ENABLED to DISABLED.

**PSMODE**

Controls whether the publish/subscribe engine and the queued publish/subscribe interface are running, and therefore controls whether applications can publish or subscribe by using the application programming interface and the queues that are monitored by the queued publish/subscribe interface.

**PSNPMSG**

If the queued publish/subscribe interface cannot process a non-persistent input message it might attempt to write the input message to the dead-letter queue (depending on the report options of the input message). If the attempt to write the input message to the dead-letter queue fails, and the MQRO\_DISCARD\_MSG report option was specified on the input message or PSNPMSG=DISCARD, the broker discards the input message. If PSNPMSG=KEEP is specified, the interface only discards the input message if the MQRO\_DISCARD\_MSG report option was set in the input message.

**PSNPRES**

If the queued publish/subscribe interface attempts to generate a response message in response to a non-persistent input message, and the response message cannot be delivered to the reply-to queue, this attribute indicates whether the interface tries to write the undeliverable message to the dead-letter queue or whether to discard the message.



## PSRTYCNT

When the queued publish/subscribe interface fails to process a command message under sync point (for example a publish message that cannot be delivered to a subscriber because the subscriber queue is full and it is not possible to put the publication on the dead letter queue), the unit of work is backed out and the command tries this number of times again before the broker attempts to process the command message according to its report options instead.

## PSSYNCPT

If this attribute is set to IFPER, when the queued publish/subscribe interface reads a publish or delete publication messages from a stream queue during normal operation then it specifies MQGMO\_SYNCPOINT\_IF\_PERSISTENT. This value makes the queued pubsub daemon receive non-persistent messages outside sync point. If the daemon receives a publication outside sync point, the daemon forwards that publication to subscribers known to it outside sync point.

## QMID

The internally generated unique name of the queue manager.

## QMNAME

The name of the local queue manager. See [Rules for naming IBM MQ objects](#).

## QSGNAME

The name of the queue sharing group to which the queue manager belongs, or blank if the queue manager is not a member of a queue sharing group. You can use queue sharing groups only on z/OS.

## RCVTIME

The approximate length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to an inactive state. The value of this parameter is the numeric value qualified by **RCVTTYPE**.

This parameter is valid only on z/OS.

## RCVTMIN

The minimum length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to an inactive state.

This parameter is valid only on z/OS.

## RCVTTYPE

The qualifier to apply to the value in **RCVTIME**.

This parameter is valid only on z/OS.

## REMOTEEV

Whether remote error events are generated.

## REPOS

The name of a cluster for which this queue manager is to provide a repository manager service.

## REPOSNL

The name of a list of clusters for which this queue manager is to provide a repository manager service.

## REVDNS

Whether reverse lookup of the host name from a Domain Name Server (DNS) is done for the IP address from which a channel has connected.

## ROUTEREC

Whether trace-route information is to be recorded if requested in the message.

## SCHINIT

Whether the channel initiator is to be started automatically when the queue manager starts.

 This parameter is not valid on z/OS.

## SCMDSERV

Whether the command server is to be started automatically when the queue manager starts.

z/OS

This parameter is not valid on z/OS.

z/OS

### SCYCASE

Whether the security profiles are uppercase or mixed case.

This parameter is valid only on z/OS.

If this parameter has been altered but the **REFRESH SECURITY** command has not yet been issued, the queue manager might not be using the case of profiles you expect. Use **DISPLAY SECURITY** to verify which case of profiles is actually in use.

### SPLCAP

Indicates if Advanced Message Security (AMS) capabilities are available to the queue manager. If the AMS component is installed for the version of IBM MQ that the queue manager is running under, the attribute has a value ENABLED. If the AMS component is not installed, the value is DISABLED.

z/OS

### SQQMNAME

When a queue manager makes an MQOPEN call for a shared queue and the queue manager that is specified in the **ObjectQmgrName** parameter of the MQOPEN call is in the same queue sharing group as the processing queue manager, the **SQQMNAME** attribute specifies whether the **ObjectQmgrName** is used or whether the processing queue manager opens the shared queue directly.

This parameter is valid only on z/OS.

### SSLRNL

Indicates the namelist of AUTHINFO objects being used for the queue manager for certificate revocation checking.

Only authentication information objects with types of CRLLDAP or OCSP are allowed in the namelist referred to by **SSLRNL**. Any other type results in an error message when the list is processed and is subsequently ignored.

ALW

### SSLCRYP

Indicates the name of the parameter string being used to configure the cryptographic hardware present on the system. The PKCS #11 password appears as xxxxxx. This is valid only on AIX, Linux, and Windows.

### SSLEV

Whether TLS events are generated.

### SSLFIPS

Whether only FIPS-certified algorithms are to be used if cryptography is processed in IBM MQ rather than in the cryptographic hardware itself.

### SSLKEYR

Indicates the name of the Secure Sockets Layer key repository.

### SSLRKEYC

Indicates the number of bytes to be sent and received within an TLS conversation before the secret key is renegotiated.

z/OS

### SSLTASKS

On z/OS only, indicates the number of server subtasks to use for processing TLS calls.

### STATACLS

Whether statistics data is to be collected for auto-defined cluster-sender channels, and, if so, the rate of data collection.

### STATCHL

It determines whether statistics data is to be collected for channels, and, if so, the rate of data collection.

**Multi** **STATINT**

The interval at which statistics monitoring data is written to the monitoring queue. This parameter is valid only on [Multiplatforms](#).

**Multi** **STATMQI**

Whether statistics monitoring data is to be collected for the queue manager. This parameter is valid only on [Multiplatforms](#).

**V 9.3.0** **STATQ**

Whether statistics data is to be collected for queues.

**STRSTPEV**

Whether start and stop events are generated.

**SUITEB**

Whether Suite B compliant cryptography is used. For more information about Suite B configuration and its effect on TLS channels, see [NSA Suite B Cryptography in IBM MQ](#).

**SYNCPT**

Whether sync point support is available with the queue manager. This is a read only queue manager attribute.

**z/OS** **TCPCHL**

The maximum number of channels that can be current, or clients that can be connected, that use the TCP/IP transmission protocol. If zero, the TCP/IP transmission protocol is not used.

This parameter is valid only on z/OS.

**z/OS** **TCPKEEP**

Whether the KEEPALIVE facility is to be used to check that the other end of the connection is still available. If it is unavailable, the channel is closed.

This parameter is valid only on z/OS.

**z/OS** **TCPNAME**

The name of the preferred TCP/IP stack to be used in a CINET multiple stack environment. In INET single stack environments the channel initiator uses the only available TCP/IP stack.

This parameter is valid only on z/OS.

**z/OS** **TCPSTACK**

Whether the channel initiator uses only the TCP/IP stack specified in TCPNAME, or can optionally bind to any of the TCP/IP stacks defined in a CINET multiple stack environment.

This parameter is valid only on z/OS.

**z/OS** **TRAXSTR**

Whether channel initiator trace starts automatically.

This parameter is valid only on z/OS.

**z/OS** **TRAXTBL**

The size, in megabytes, of the trace data space of the channel initiator.

This parameter is valid only on z/OS.

**TREELIFE**

The lifetime of non-administrative topics.

**TRIGINT**

The trigger interval.

**VERSION**

The version of the IBM MQ installation that the queue manager is associated with. The version has the format VVRRMMFF:

VV: Version  
RR: Release  
MM: Maintenance level  
FF: Fix level

### XRCAP

Whether MQ Telemetry capability is supported by the queue manager.

For more information about these parameters, see [“ALTER QMGR \(alter queue manager settings\)”](#) on page 348.

### Related tasks

[Displaying and altering queue manager attributes](#)

## **Multi** **DISPLAY QMSTATUS (display queue manager status) on Multiplatforms**

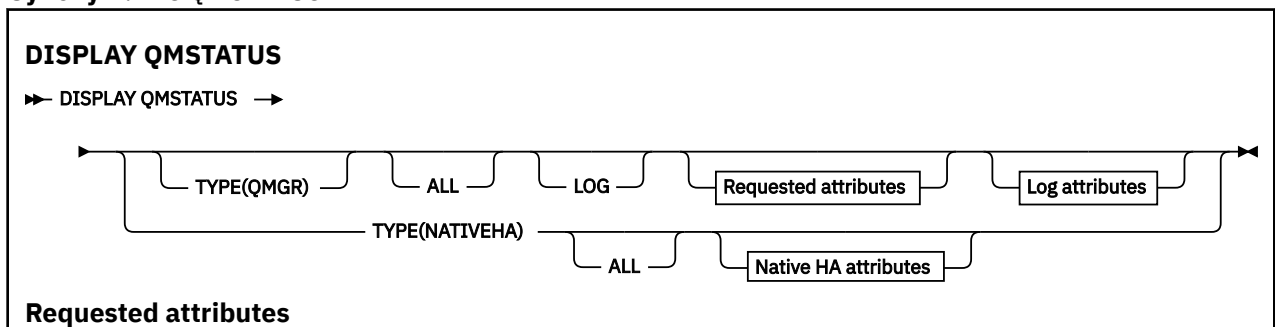
Use the MQSC command **DISPLAY QMSTATUS** to display status information associated with this queue manager.

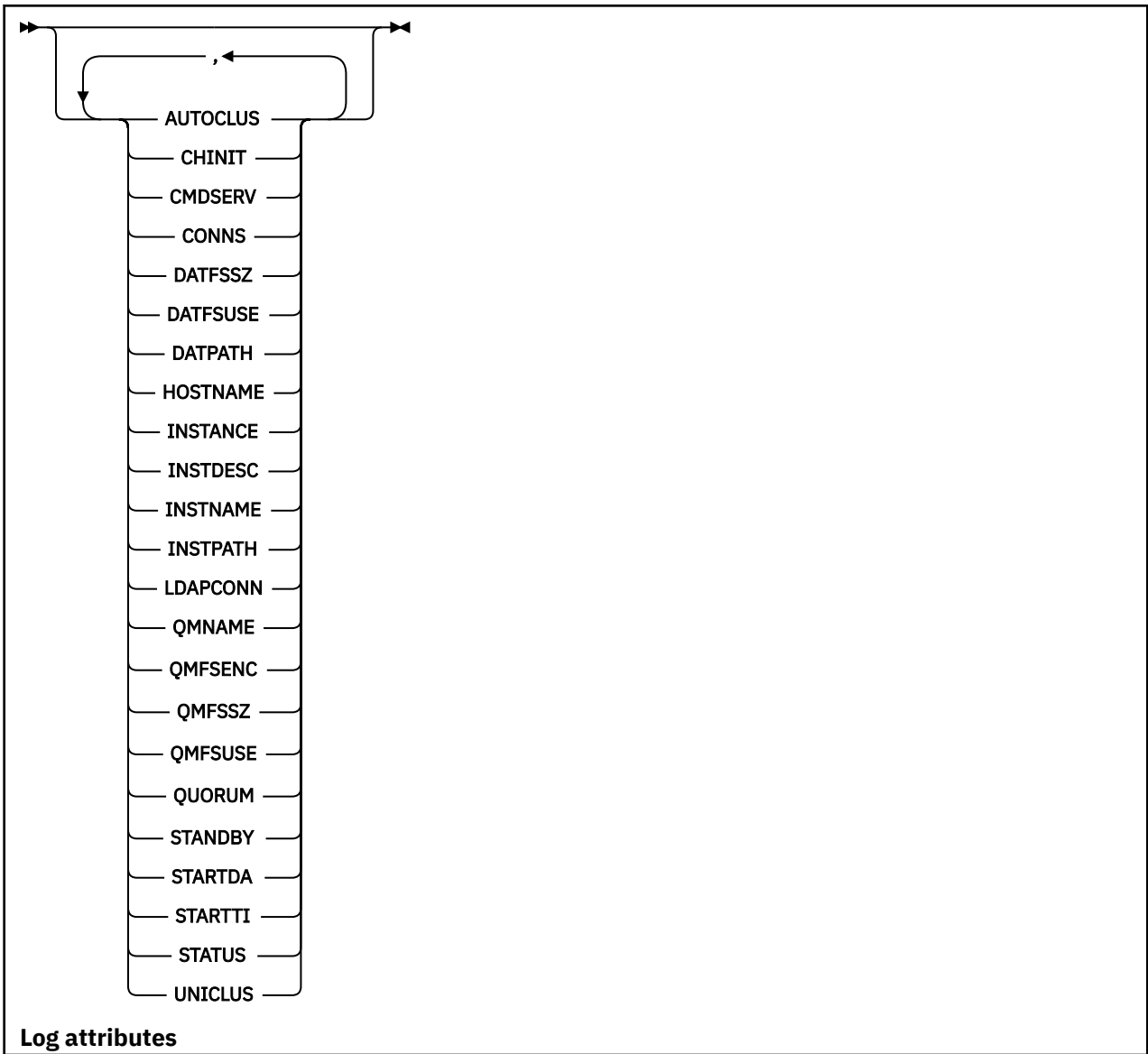
### Using MQSC commands

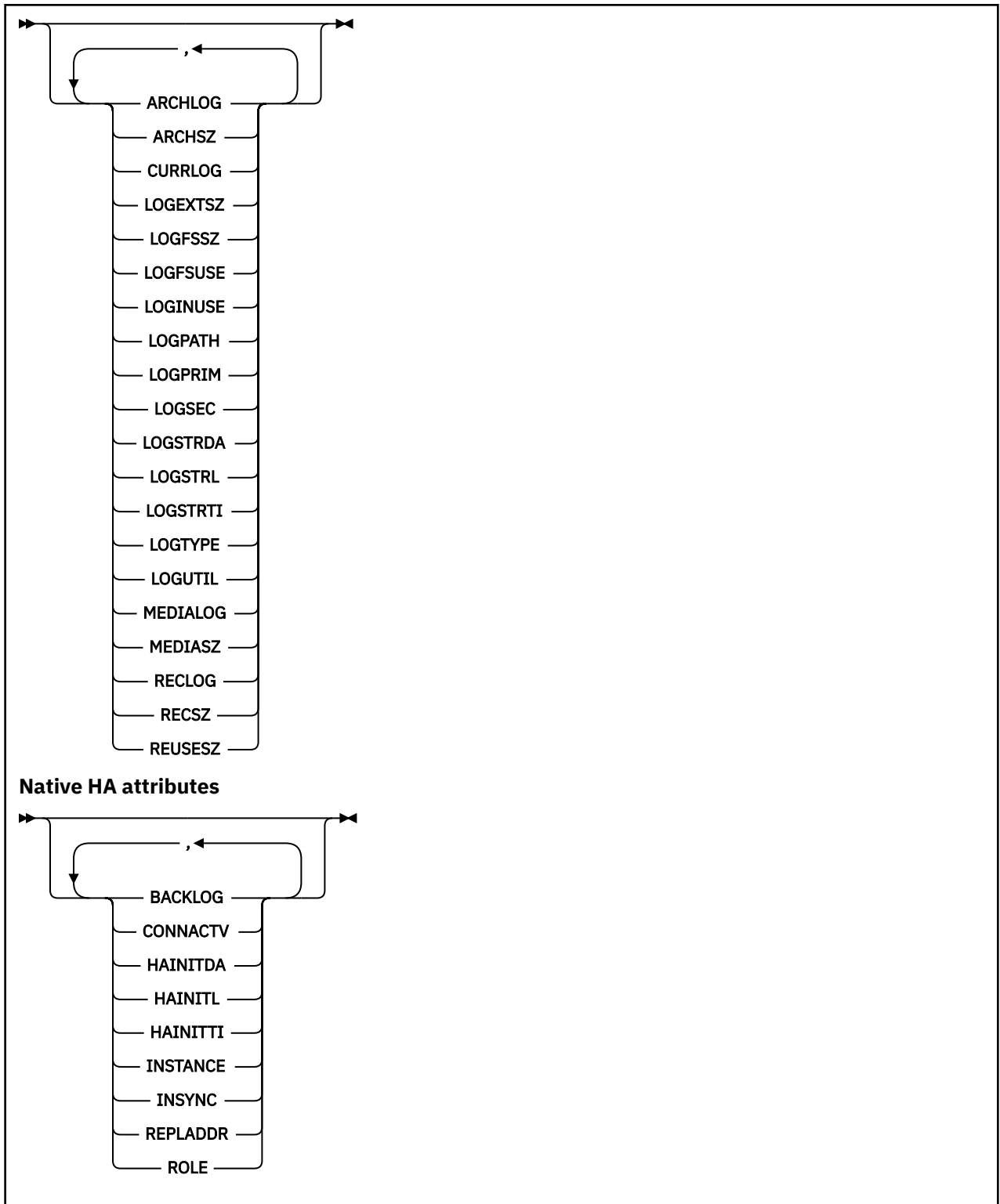
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY QMSTATUS”](#) on page 782
- [“Requested parameters”](#) on page 783

**Synonym:** DIS QMSTATUS







## Parameter descriptions for DISPLAY QMSTATUS

### ALL

Specify this parameter to display all the parameters. If this parameter is specified, any parameters that are requested specifically have no effect; all parameters are still displayed.

This parameter is the default if you do not request any specific parameters.

## LOG

Specify this parameter to display all the **LOG** parameters. If this parameter is specified, any **LOG** parameters that are requested specifically have no effect; all parameters are still displayed.

**V 9.3.2** For parameters for **DISPLAY QMSTATUS TYPE(NATIVEHA)**, see [“Attributes for TYPE \(NATIVEHA\)”](#) on page 788.

## Requested parameters

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

### ARCHLOG

Name of the oldest log extent for which the queue manager is waiting for archive notification. This parameter is:

- Available only on queue managers using archive log management.
- Blank, if the queue manager is not using archive log management, or if the queue manager has no extents waiting for notification.

**IBM i** This parameter is not valid on IBM i.

### ARCHSZ

The amount of space occupied, in megabytes, by log extents no longer required for restart or media recovery, but waiting to be archived.

Note that this value impacts the total space used by the queue manager for log extents.

This parameter is available only on queue managers using archive log management. If the queue manager is not using archive log management, this parameter is zero.

**IBM i** This parameter is not valid on IBM i.

### **V 9.3.2** AUTOCLUS

Whether the queue manager is a member of an automatic cluster. Is one of the following:

#### **NONE**

Does not belong to an automatic cluster.

#### **UNIFORM**

Is part of a uniform cluster.

See [Using automatic cluster configuration](#).

### CHINIT

The status of the channel initiator reading SYSTEM.CHANNEL.INITQ. It is one of the following:

#### **STOPPED**

The channel initiator is not running.

#### **STARTING**

The channel initiator is in the process of initializing and is not yet operational.

#### **RUNNING**

The channel initiator is fully initialized and is running.

#### **STOPPING**

The channel initiator is stopping.

### CMDSERV

The status of the command server. It is one of the following:

**STOPPED**

The command server is not running.

**STARTING**

The command server is in the process of initializing and is not yet operational.

**RUNNING**

The command server is fully initialized and is running.

**STOPPING**

The command server is stopping.

**CONNS**

The current number of connections to the queue manager.

**CURRLOG**

The name of the log extent being written to at the time that the **DISPLAY QMSTATUS** command is processed. If the queue manager is using circular logging, and this parameter is explicitly requested, a blank string is displayed.

**V 9.3.2 DATFSSZ**

The size of the dedicated queue manager data file system in MB, rounded up. If the queue manager data and recovery log are on the same file system, the value is **SHARED** (see QMFSSZ).

**V 9.3.2 DATFSUSE**

The percentage of the dedicated queue manager data file system that is full, rounded up to the nearest percent. If the queue manager data and recovery log are on the same file system, the value is **SHARED** (see QMFSUSE).

**V 9.3.2 DATPATH**

The path of the queue manager data directory.

**V 9.3.2 HOSTNAME**

The name of the host that the queue manager is running on from the perspective of IBM MQ. Usually, this is the value reported by the O/S but it can be overridden by the MQS\_IPC\_HOST environment variable.

**V 9.3.2 INSTANCE**

The name of the local Native HA instance when the queue manager is part of a Native HA group. It is blank otherwise.

**INSTDESC**

Description of the installation associated with the queue manager.

**INSTNAME**

Name of the installation associated with the queue manager.

**INSTPATH**

Path of the installation associated with the queue manager.

**LDAPCONN**

The status of the connection to the LDAP server. It is one of the following:

**CONNECTED**

The queue manager currently has a connection to the LDAP server.

**ERROR**

The queue manager attempted to make a connection to the LDAP server and failed.



## INACTIVE

The queue manager is not configured to use an LDAP server or has not yet made a connection to the LDAP server.

**Note:** The **LDAPCONN** status within **DISPLAY QMSTATUS** is a single status for the whole queue manager, reflecting only the most recent actions performed with the LDAP server. There are multiple connections to the LDAP server, one per queue manager agent process. **LDAPCONN** reflects the status from the most recent LDAP connection across the agents of the whole queue manager. If the error is temporary, and quickly clears, then the **ERROR** status will be short-lived. Always look in the [queue manager error logs](#) to see more details of any LDAP connectivity failures.

### V 9.3.2 LOGEXTSZ

The size of each log file (or the threshold of the currently attached journal receiver on IBM i) in KB.

### V 9.3.2 LOGFSSZ

The size of the dedicated recovery log file system in MB, rounded up. If the queue manager data and recovery log are on the same file system, the value is **SHARED** (see QMFSSZ).

### V 9.3.2 LOGFSUSE

The percentage of the dedicated recovery log file system that is full, rounded up to the nearest percent. If the queue manager data and recovery log are on the same file system, the value is **SHARED** (see QMFSUSE).

## LOGINUSE

The percentage of the primary log space in use for restart recovery at this point in time.

A value of 100 or greater indicates the queue manager might have allocated, and be using, secondary log files, probably due to long-lived transactions at this point in time.

▶ **IBM i** This parameter is not valid on IBM i.

## LOGPATH

Identifies the directory where log files are created by the queue manager.

### V 9.3.2 LOGPRIM

The number of primary log files.

▶ **IBM i** This parameter is not valid on IBM i.

### V 9.3.2 LOGSEC

The maximum number of secondary log files.

▶ **IBM i** This parameter is not valid on IBM i.

### V 9.3.2 LOGSTRDA

The date for the last log record recovered when the queue manager started.

▶ **IBM i** This parameter is not valid on IBM i.

### V 9.3.2 LOGSTRL

The log sequence number (LSN) for the last log record recovered when the queue manager started, in `<nnnnn:nnnnn:nnnnn:nnnnn>` format.

▶ **IBM i** This parameter is not valid on IBM i.

### V 9.3.2 LOGSTRTI

The time for the last log record recovered when the queue manager was started.

**IBM i** This parameter is not valid on IBM i.

### V 9.3.2 LOGTYPE

The type of logging, is one of the following values:

#### **CIRCULAR**

Uses circular logging.

#### **LINEAR**

Uses linear logging.

#### **REPLICATED**

Uses replicated logging.

For information about logging types, see [Types of logging](#).

### **LOGUTIL**

A percentage estimate of how well the queue manager workload is contained within the primary log space.

If the value is consistently above 100 you might want to investigate whether there are long-lived transactions, or if the number of primary files is not sufficient for the workload.

If the utilization continues to rise, eventually requests for most further operations requiring log activity will be refused, together with an MQRC\_RESOURCE\_PROBLEM return code being returned to the application. Transactions might be backed out.

**IBM i** This parameter is not valid on IBM i.

### **MEDIALOG**

The name of the oldest log extent required by the queue manager to perform media recovery. If the queue manager is using circular logging, and this parameter is explicitly requested, a blank string is displayed.

### **MEDIASZ**

Size of the log data required for media recovery in megabytes.

This value shows how much log that must be read for media recovery and directly impacts the time taken for this operation.

This is zero for a circular logging queue manager. The size is typically reduced by taking more frequent media images of objects.

**IBM i** This parameter is not valid on IBM i.

### V 9.3.2 QMFSENC

Indicates whether the queue manager file system is encrypted or not. Is set to one of the following values:

#### **NO**

The file system is not encrypted.

#### **YES**

The file system is encrypted.

#### **UNKNOWN**

The encryption status is not known.

**MQ Appliance** Returned on the IBM MQ Appliance only.

### V 9.3.2 QMFSSZ

The total size of the queue manager data and recovery log file systems in MB, rounded up. If the queue manager data and recovery log are on the same file system, the value is the size of that file system.

### V 9.3.2 QMFSUSE

The percentage of the queue manager data and recovery log file systems that are full, rounded up to the nearest percent. If the queue manager data and recovery log are on the same file system, the value will be the usage of that file system.

### QMNAME

The name of the queue manager. This parameter is always returned.

### V 9.3.2 QUORUM

Two numeric values separated by a forward slash (X/Y) if the queue manager is part of a Native HA group, or blank otherwise. The two values give the following information:

- X is how many of the configured instances are in-sync with the active instance.
- Y is the total number of configured instances of the queue manager.

### RECLOG

The name of the oldest log extent required by the queue manager to perform restart recovery. If the queue manager is using circular logging, and this parameter is explicitly requested, a blank string is displayed.

### RECSZ

Size of the log data required for restart recovery in megabytes.

This value shows how much log that must be read for restart recovery and directly impacts the time taken for this operation.

 This parameter is not valid on IBM i.

### REUSESZ

This attribute is valid only on automatic or archive log management queue managers.

The amount of space occupied, in megabytes, by log extents available to be reused.

This value impacts the total space used by the queue manager for log extents.

The size is automatically managed by the queue manager, but if necessary you can request reductions using the **RESET QMGR TYPE (REDUCELOG)** command.

 This parameter is not valid on IBM i.

### STANDBY

Whether a standby instance is permitted. It is one of the following:

#### NOPERMIT

Standby instances are not permitted.

#### PERMIT

Standby instances are permitted.

### STARTDA

The date on which the queue manager was started (in the form yyyy-mm-dd).

### STARTTI

The time at which the queue manager was started (in the form hh.mm.ss).

## STATUS

The status of the queue manager. It is one of the following:

### STARTING

The queue manager is in the process of initializing.

### RUNNING

The queue manager is fully initialized and is running.

### QUIESCING

The queue manager is quiescing.

## V 9.3.2 UNICLUS

The name of the uniform cluster that the queue manager is a member of, or blank otherwise.

## Attributes for TYPE (NATIVEHA)

### V 9.3.2

When you specify TYPE(NATIVEHA) you can specify ALL to have all attributes returned, alternatively you can specify one or more of the following parameters:

### BACKLOG (for REPLICAs instances only)

How many KB of recovery log data the active instance has written that have not yet been acknowledged by the named instance. This is how 'out of sync' the named instance currently is. The value is rounded up to a 1 KB boundary. Set to **UNKNOWN** if the information is not available.

### CONNECTV (for REPLICAs instances only)

Whether the instance currently has a pair of active connections to the active instance. Set to **YES** or **NO**.

### HAINITDA (for ACTIVE instances only)

The date of the last log record recovered when the Native HA group initially became active.

### HAINITL (for ACTIVE instances only)

The log sequence number (LSN) of the last log record recovered when the Native HA group initially became active, in the format `<nnnnn:nnnnn:nnnnn:nnnnn>`.

### HAINITTI (for ACTIVE instances only)

The time of the last log record recovered when the Native HA group initially became active.

## INSTANCE

The name of the Native HA instance.

### INSYNC (for REPLICAs instances only)

Whether this instance is currently considered in-sync with the active instance. Set to **YES** or **NO**.

## REPLADDR

The network address and port to use when sending data to and from the specified instance.

## ROLE

The current role of the instance in the Native HA group. Can be **ACTIVE**, **REPLICA**, or **UNKNOWN**.

See [Native HA](#) for a description of these roles.

If you use the **DISPLAY QMSTATUS TYPE (NATIVEHA)** command in an environment where Native HA is not supported or configured, the following message is returned:

```
AMQ5708E: Native HA not available.
```

## DISPLAY QSTATUS (display queue status)

Use the MQSC command DISPLAY QSTATUS to display the status of one or more queues.

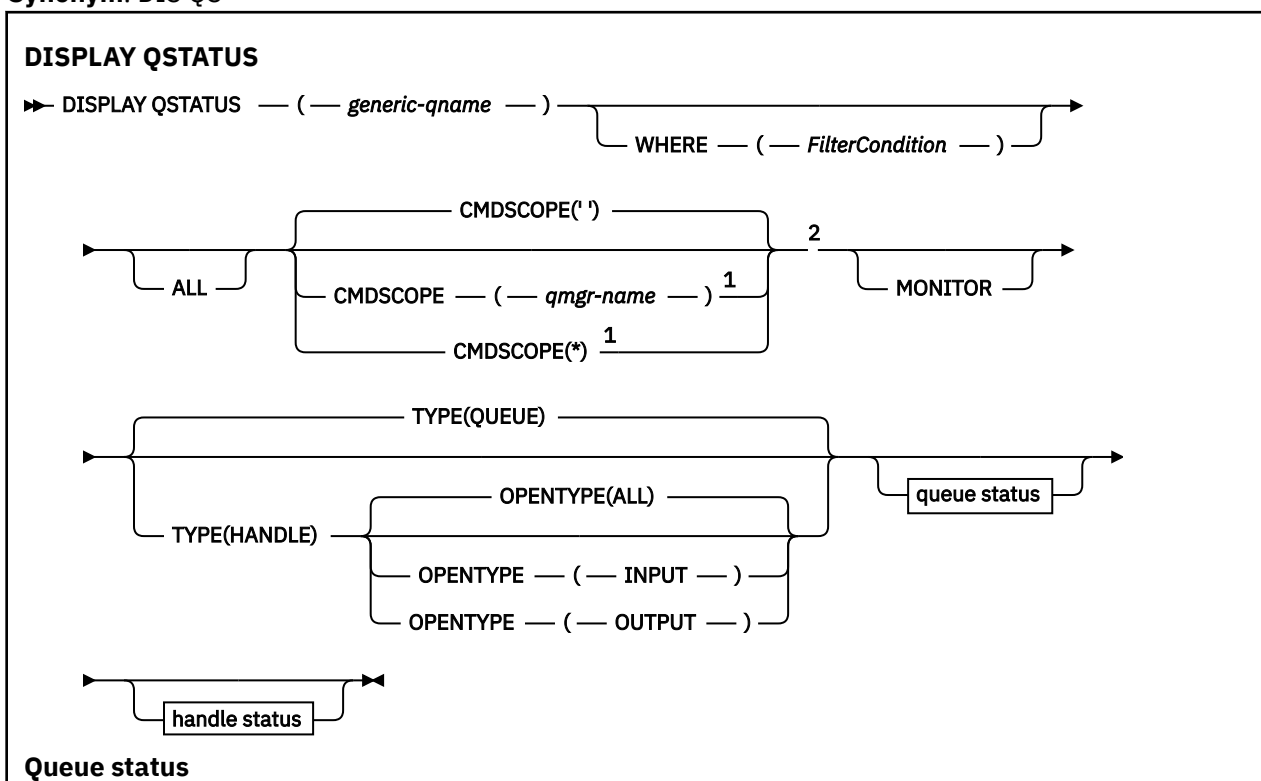
### Using MQSC commands

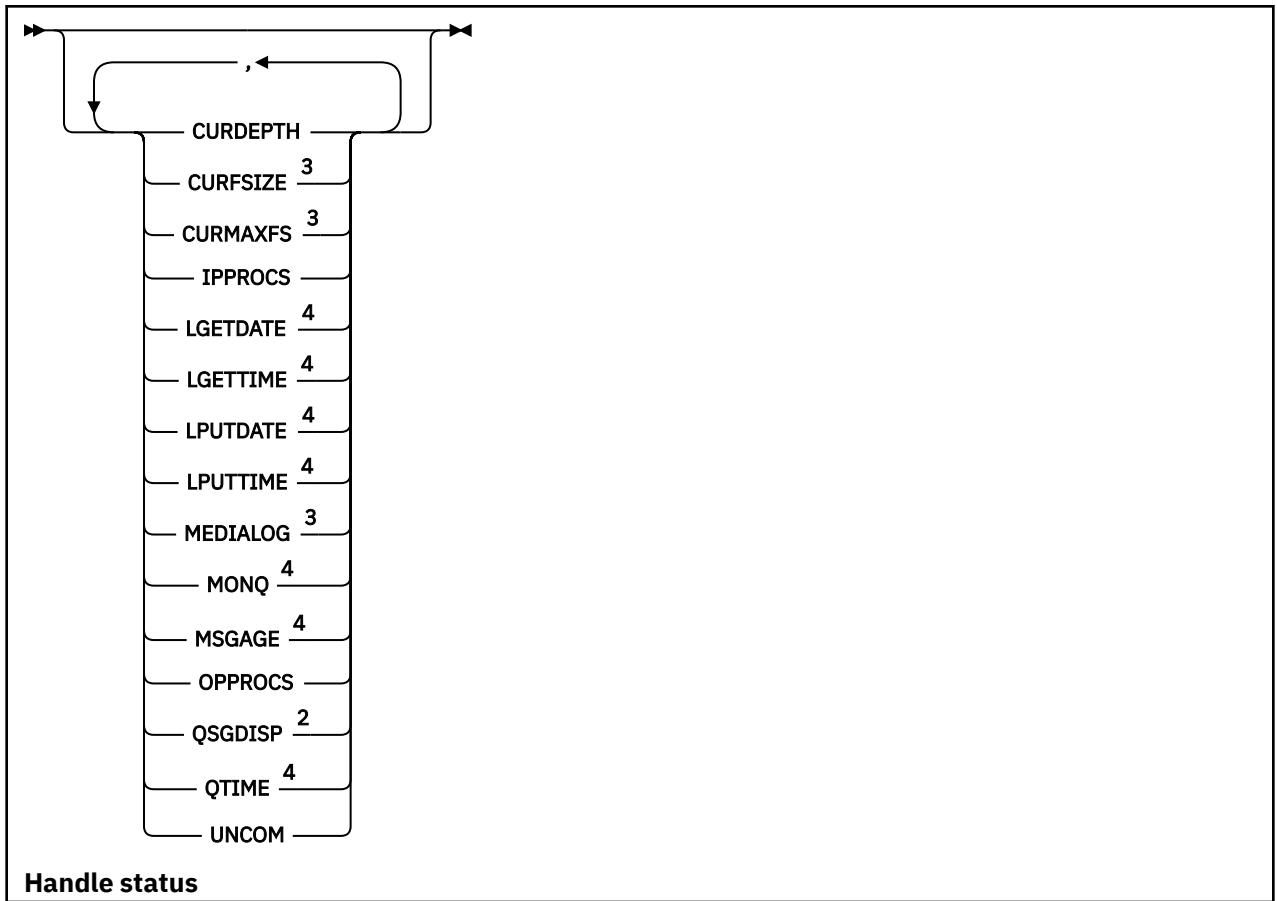
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

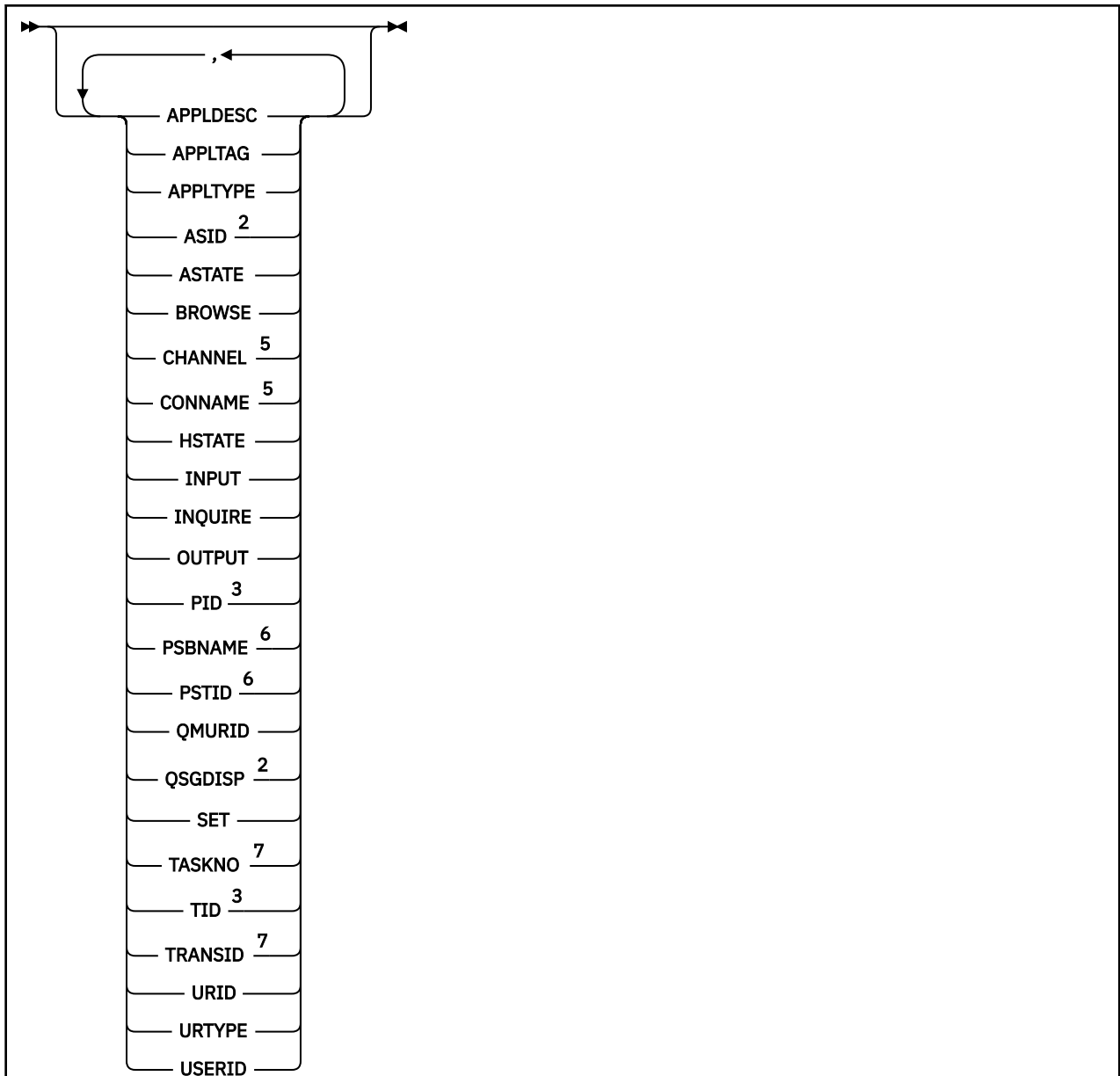
**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DISPLAY QSTATUS” on page 791](#)
- [“Parameter descriptions for DISPLAY QSTATUS” on page 792](#)
- [“Queue status” on page 794](#)
- [“Handle status” on page 797](#)

**Synonym:** DIS QS







Notes:

- 1 Valid only on z/OS when the queue manager is a member of a queue sharing group.
- 2 Valid on z/OS only.
- 3 Not valid on z/OS.
- 4 Also displayed by selection of the MONITOR parameter.
- 5 Channel initiator only
- 6 IMS only
- 7 CICS only

**Usage notes for DISPLAY QSTATUS**

The state of asynchronous consumers, ASTATE, reflects that of the server-connection proxy on behalf of the client application; it does not reflect the client application state.

## Parameter descriptions for DISPLAY QSTATUS

You must specify the name of the queue for which you want to display status information. This name can either be a specific queue name or a generic queue name. By using a generic queue name you can display either:

- Status information for all queues, or
- Status information for one or more queues that match the specified name and other selection criteria

You must also specify whether you want status information about:

- Queues
- Handles that are accessing the queues

**Note:** You cannot use the DISPLAY QSTATUS command to display the status of an alias queue or remote queue. If you specify the name of one of these types of queue, no data is returned. You can, however, specify the name of the local queue or transmission queue to which the alias queue or remote queue resolves.

### ( *generic-qname* )

The name of the queue for which status information is to be displayed. A trailing asterisk (\*) matches all queues with the specified stem followed by zero or more characters. An asterisk (\*) on its own matches all queues.

### WHERE

Specify a filter condition to display status information for queues that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Almost any parameter that can be used to display attributes for this DISPLAY command. However, you cannot use the CMDSCOPE, MONITOR, OPENTYPE, QSGDISP, QTIME, TYPE, or URID parameters as filter keywords.

#### **operator**

The operator is used to determine whether a queue satisfies the filter value on the given filter keyword. The operators are:

#### **LT**

Less than

#### **GT**

Greater than

#### **EQ**

Equal to

#### **NE**

Not equal to

#### **LE**

Less than or equal to

#### **GE**

Greater than or equal to

#### **LK**

Matches a generic string that you provide as a *filter-value*

#### **NL**

Does not match a generic string that you provide as a *filter-value*

#### **CT**

Contains a specified item. If the *filter-keyword* is a list, you can use this filter to display objects whose attributes contain the specified item.

#### **EX**

Does not contain a specified item. If the *filter-keyword* is a list, you can use this filter to display objects whose attributes do not contain the specified item.



### **filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this value can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value NO on the UNCOM parameter), you can only use EQ or NE.

- A generic value. This value is a character string (such as the character string in the APPLTAG parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

- An item in a list of values. The operator must be CT or EX. If it is a character value, it can be explicit or generic. For example, if the value DEF is specified with the operator CT, all items where one of the attribute values is DEF are listed. If ABC\* is specified, all items where one of the attribute values begins with ABC are listed.

### **ALL**

Display all the status information for each specified queue.

This value is the default if you do not specify a generic name, and do not request any specific parameters.

**z/OS** On z/OS, this value is also the default if you specify a filter condition using the WHERE parameter, but on other platforms only the requested attributes are displayed.

### **z/OS CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group. It is valid on z/OS only.

..

The command runs on the queue manager on which it was entered. This value is the default.

#### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this value is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

### **MONITOR**

Specify this value to return the set of online monitoring parameters. These are LGETDATE, LGETTIME, LPUTDATE, LPUTTIME, MONQ, MSGAGE, and QTIME. If you specify this parameter, any of the monitoring parameters that you request specifically have no effect; all monitoring parameters are still displayed.

### **OPENTYPE**

Restricts the queues selected to queues which have handles with the specified type of access:

**ALL**

Selects queues that are open with any type of access. This value is the default if the OPENTYPE parameter is not specified.

**INPUT**

Selects queues that are open for input only. This option does not select queues that are open for browse.

**OUTPUT**

Selects queues that are open only for output.

The OPENTYPE parameter is valid only if TYPE(HANDLE) is also specified.

You cannot use OPENTYPE as a filter keyword.

**TYPE**

Specifies the type of status information required:

**QUEUE**

Status information relating to queues is displayed. This value is the default if the TYPE parameter is not specified.



**HANDLE**

Status information relating to the handles that are accessing the queues is displayed.

You cannot use TYPE as a filter keyword.

**Queue status**

For queue status, the following information is always returned for each queue that satisfies the selection criteria, except where indicated:

- Queue name
- Type of information returned (TYPE parameter)
-  Current queue depth (CURDEPTH parameter)
-  On z/OS only, the queue sharing group disposition (QSGDISP parameter)

The following parameters can be specified for TYPE(Queue) to request additional information for each queue. If a parameter is specified that is not relevant for the queue, operating environment, or type of status information requested, that parameter is ignored.

**CURDEPTH**

The current depth of the queue, that is, the number of messages on the queue, including both committed messages and uncommitted messages.

 **CURFSIZE**

Indicates the current size of the queue file in megabytes, rounded up to the nearest megabyte.

For a new queue with default attributes, the value of CURFSIZE is 1.

 **CURMAXFS**

Indicates the current maximum size the queue file can grow to, rounded up to the nearest megabyte, given the current block size in use on a queue.

The use of this field is two fold:

- If you set MAXFSIZE(DEFAULT) for the current block size, CURMAXFS shows the actual value that DEFAULT equates to.
- If CURMAXFS does not match MAXFSIZE, you know the queue must be drained in order to adopt a bigger granularity.


**IPPROCS**

The number of handles that are currently open for input for the queue (either input-shared or input-exclusive). This number does not include handles that are open for browse.

For shared queues, the number returned applies only to the queue manager generating the reply. The number is not the total for all the queue managers in the queue sharing group.

### **LGETDATE**

The date on which the last message was retrieved from the queue since the queue manager started. A message being browsed does not count as a message being retrieved. When no get date is available, perhaps because no message has been retrieved from the queue since the queue manager was started, the value is shown as a blank.


 For queues with QSGDISP(SHARED), the value shown is for measurements collected on this queue manager only.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONQ is set to a value other than OFF for this queue.

### **LGETTIME**

The time at which the last message was retrieved from the queue since the queue manager started. A message being browsed does not count as a message being retrieved. When no get time is available, perhaps because no message has been retrieved from the queue since the queue manager was started, the value is shown as a blank.


 For queues with QSGDISP(SHARED), the value shown is for measurements collected on this queue manager only.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONQ is set to a value other than OFF for this queue.

### **LPUTDATE**

The date on which the last message was put to the queue since the queue manager started. When no put date is available, perhaps because no message has been put to the queue since the queue manager was started, the value is shown as a blank.


 For queues with QSGDISP(SHARED), the value shown is for measurements collected on this queue manager only.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONQ is set to a value other than OFF for this queue.

### **LPUTTIME**

The time at which the last message was put to the queue since the queue manager started. When no put time is available, perhaps because no message has been put to the queue since the queue manager was started, the value is shown as a blank.

 For queues with QSGDISP(SHARED), the value shown is for measurements collected on this queue manager only.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONQ is set to a value other than OFF for this queue.

**Note:** Moving the system clock backwards should be avoided in case the LPUTTIME is being used to monitor the messages. The LPUTTIME of a queue is only updated when a message that arrives on the queue has a PutTime greater than the existing value of LPUTTIME. Because the PutTime of the message is less than the existing LPUTTIME of the queue in this case, the time is left unchanged.

### **MEDIALOG**

The log extent or journal receiver needed for media recovery of the queue. On queue managers on which circular logging is in place, MEDIALOG is returned as a null string.

This parameter is valid only on Multiplatforms.

### **MONQ**

Current level of monitoring data collection for the queue.

This parameter is also displayed when you specify the MONITOR parameter.

### **MSGAGE**

Age, in seconds, of the oldest message on the queue. The maximum displayable value is 999999999; if the age exceeds this value, 999999999 is displayed.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONQ is set to a value other than OFF for this queue.

### **OPPROCS**

This is the number of handles that are currently open for output for the queue.

For shared queues, the number returned applies only to the queue manager generating the reply. The number is not the total for all the queue managers in the queue sharing group.

### **z/OS QSGDISP**

Indicates the disposition of the queue. The value displayed is one of the following:

#### **QMGR**

The object was defined with QSGDISP(QMGR).

#### **COPY**

The object was defined with QSGDISP(COPY).

#### **SHARED**

The object was defined with QSGDISP(SHARED).

This parameter is valid on z/OS only.

For shared queues, if the CF structure used by the queue is unavailable or has failed, the status information might be unreliable.

You cannot use QSGDISP as a filter keyword.

### **QTIME**

Interval, in microseconds, between messages being put on the queue and then being destructively read. The maximum displayable value is 999999999; if the interval exceeds this value, 999999999 is displayed.

The interval is measured from the time that the message is placed on the queue until it is destructively retrieved by an application and, therefore, includes any interval caused by a delay in committing by the putting application.

Two values are displayed and these are recalculated only when messages are processed:

- A value based on the last few messages processed
- A value based on a larger sample of the recently processed messages

These values depend on the configuration and behavior of your system, as well as the levels of activity within it, and serve as an indicator that your system is performing normally. A significant variation in these values might indicate a problem with your system. For queues with QSGDISP(SHARED), the values shown are for measurements collected on this queue manager only.

This parameter is also displayed when you specify the MONITOR parameter.

A value is only displayed for this parameter if MONQ is set to a value other than OFF for this queue.

### **UNCOM**

Indicates whether there are any uncommitted changes (puts and gets) pending for the queue. The value displayed is one of the following:

#### **YES**

On z/OS, there are one or more uncommitted changes pending.

#### **NO**

There are no uncommitted changes pending.

*n*

► **Multi** On Multiplatforms, an integer value indicating how many uncommitted changes are pending.

For shared queues, the value returned applies only to the queue manager generating the reply. The value does not apply to all the queue managers in the queue sharing group.

## Handle status

For handle status, the following information is always returned for each queue that satisfies the selection criteria, except where indicated:

- Queue name
- Type of information returned (TYPE parameter)
- ► **Multi** User identifier (USERID parameter) - not returned for APPLTYPE(SYSTEM)

**Note:** ► **z/OS** Returned only if requested on z/OS

- ► **Multi** Application tag (APPLTAG parameter)
- Application type (APPLTYPE parameter)
- ► **z/OS** On z/OS only, the queue sharing group disposition (QSGDISP parameter)

The following parameters can be specified for TYPE(HANDLE) to request additional information for each queue. If a parameter that is not relevant is specified for the queue, operating environment, or type of status information requested, that parameter is ignored.

### APPLDESC

A string containing a description of the application connected to the queue manager, where it is known. If the application is not recognized by the queue manager the description returned is blank.

### APPLTAG

A string containing the tag of the application connected to the queue manager. It is one of the following:

- ► **z/OS** z/OS batch job name
- ► **z/OS** TSO USERID
- CICS APPLID
- IMS region name
- Channel initiator job name
- ► **IBM i** IBM i job name
- ► **Linux** ► **AIX** Linux or AIX process
- ► **Windows** Windows process

**Note:** The returned value consists of the full program path and executable file name. If it is more than 28 characters long, only the first 28 characters are shown.

- Internal queue manager process name

Application name represents the name of the process or job that has connected to the queue manager. In the instance that this process or job is connected via a channel, the application name represents the remote process or job rather than the local channel process or job name.

### APPLTYPE

A string indicating the type of the application that is connected to the queue manager. It is one of the following:

**BATCH**

Application using a batch connection

**RRSBATCH**

RRS-coordinated application using a batch connection

**CICS**

CICS transaction

**IMS**

IMS transaction

**CHINIT**

Channel initiator

**SYSTEM**

Queue manager

**SYSTEMEXT**

Application performing an extension of function that is provided by the queue manager

**USER**

A user application

**z/OS ASID**

A four-character address-space identifier of the application identified by APPLTAG. It distinguishes duplicate values of APPLTAG.

This parameter is returned only when the queue manager owning the queue is running on z/OS, and the APPLTYPE parameter does not have the value SYSTEM.

**ASTATE**

The state of the asynchronous consumer on this queue.

Possible values are:

**ACTIVE**

An MQCB call has set up a function to call back to process messages asynchronously and the connection handle has been started so that asynchronous message consumption can proceed.

**INACTIVE**

An MQCB call has set up a function to call back to process messages asynchronously but the connection handle has not yet been started, or has been stopped or suspended, so that asynchronous message consumption cannot currently proceed.

**SUSPENDED**

The asynchronous consumption call-back has been suspended so that asynchronous message consumption cannot currently proceed on this queue. This can be either because an MQCB call with Operation MQOP\_SUSPEND has been issued against this object handle by the application, or because it has been suspended by the system. If it has been suspended by the system, as part of the process of suspending asynchronous message consumption the call-back function is initiated with the reason code that describes the problem resulting in suspension. This code is reported in the Reason field in the MQCBC structure that is passed to the call-back function.

For asynchronous message consumption to proceed, the application must issue an MQCB call with the Operation parameter set to MQOP\_RESUME.

**SUSPTEMP**

The asynchronous consumption call-back has been temporarily suspended by the system so that asynchronous message consumption cannot currently proceed on this queue. As part of the process of suspending asynchronous message consumption, the call-back function is called with the reason code that describes the problem resulting in suspension. This code is reported in the Reason field in the MQCBC structure passed to the call-back function.

The call-back function is initiated again when asynchronous message consumption is resumed by the system, when the temporary condition has been resolved.

**NONE**

An MQCB call has not been issued against this handle, so no asynchronous message consumption is configured on this handle.

**BROWSE**

Indicates whether the handle is providing browse access to the queue. The value is one of the following:

**YES**

The handle is providing browse access.

**NO**

The handle is not providing browse access.

**CHANNEL**

The name of the channel that owns the handle. If there is no channel associated with the handle, this parameter is blank.

This parameter is returned only when the handle belongs to the channel initiator.

**CONNNAME**

The connection name associated with the channel that owns the handle. If there is no channel associated with the handle, this parameter is blank.

This parameter is returned only when the handle belongs to the channel initiator.

**HSTATE**

Whether an API call is in progress.

Possible values are:

**ACTIVE**

An API call from a connection is currently in progress for this object. For a queue, this condition can arise when an MQGET WAIT call is in progress.

If there is an MQGET SIGNAL outstanding, then this value does not mean, by itself, that the handle is active.

**INACTIVE**

No API call from a connection is currently in progress for this object. For a queue, this condition can arise when no MQGET WAIT call is in progress.

**INPUT**

Indicates whether the handle is providing input access to the queue. The value is one of the following:

**SHARED**

The handle is providing shared-input access.

**EXCL**

The handle is providing exclusive-input access.

**NO**

The handle is not providing input access.

**INQUIRE**

Indicates whether the handle currently provides inquire access to the queue. The value is one of the following:

**YES**

The handle provides inquire access.

**NO**

The handle does not provide inquire access.

**OUTPUT**

Indicates whether the handle is providing output access to the queue. The value is one of the following:

**YES**

The handle is providing output access.

**NO**

The handle is not providing output access.

**PID**

Number specifying the process identifier of the application that has opened the specified queue.

**z/OS** This parameter is not valid on z/OS.

**z/OS PSBNAME**

The eight characters long name of the program specification block (PSB) associated with the running IMS transaction. You can use the PSBNAME and PSTID to purge the transaction using IMS commands. It is valid on z/OS only.

This parameter is returned only when the APPLTYPE parameter has the value IMS.

**z/OS PSTID**

The four character IMS program specification table (PST) region identifier for the connected IMS region. It is valid on z/OS only.

This parameter is returned only when the APPLTYPE parameter has the value IMS.

**QMURID**

The queue manager unit of recovery identifier. On z/OS, this value is an 8-byte log RBA, displayed as 16 hexadecimal characters. On platforms other than z/OS, this value is an 8-byte transaction identifier, displayed as m.n where m and n are the decimal representation of the first and last 4 bytes of the transaction identifier.

You can use QMURID as a filter keyword. On z/OS, you must specify the filter value as a hexadecimal string. On platforms other than z/OS, you must specify the filter value as a pair of decimal numbers separated by a period (.). You can only use the EQ, NE, GT, LT, GE, or LE filter operators.

**z/OS QSGDISP**

Indicates the disposition of the queue. It is valid on z/OS only. The value is one of the following:

**QMGR**

The object was defined with QSGDISP(QMGR).

**COPY**

The object was defined with QSGDISP(COPY).

**SHARED**

The object was defined with QSGDISP(SHARED).

You cannot use QSGDISP as a filter keyword.

**SET**

Indicates whether the handle is providing set access to the queue. The value is one of the following:

**YES**

The handle is providing set access.

**NO**

The handle is not providing set access.

**z/OS TASKNO**

A seven-digit CICS task number. This number can be used in the CICS command "CEMT SET TASK(taskno) PURGE" to end the CICS task. This parameter is valid on z/OS only.

This parameter is returned only when the APPLTYPE parameter has the value CICS.

**TID**

Number specifying the thread identifier within the application process that has opened the specified queue.



**z/OS** This parameter is not valid on z/OS.

An asterisk indicates that this queue was opened using a shared connection.

For further information about shared connections see [Shared \(thread independent\) connections with MQCONNX](#).

**z/OS** **TRANSID**

A four-character CICS transaction identifier. This parameter is valid on z/OS only.

This parameter is returned only when the APPLTYPE parameter has the value CICS.

#### **URID**

The external unit of recovery identifier associated with the connection. It is the recovery identifier known in the external syncpoint coordinator. Its format is determined by the value of URTYPE.

You cannot use URID as a filter keyword.

#### **URTYPE**

The type of unit of recovery as seen by the queue manager. It is one of the following:

- CICS (valid only on z/OS )
- XA
- RRS (valid only on z/OS )
- IMS (valid only on z/OS )
- QMGR

URTYPE identifies the EXTURID type and not the type of the transaction coordinator. When URTYPE is QMGR, the associated identifier is in QMURID (and not URID).

#### **USERID**

The user identifier associated with the handle.

This parameter is not returned when APPLTYPE has the value SYSTEM.

## **DISPLAY QUEUE (display queue attributes)**

Use the MQSC command **DISPLAY QUEUE** to display the attributes of one or more queues of any type.

### **Using MQSC commands**

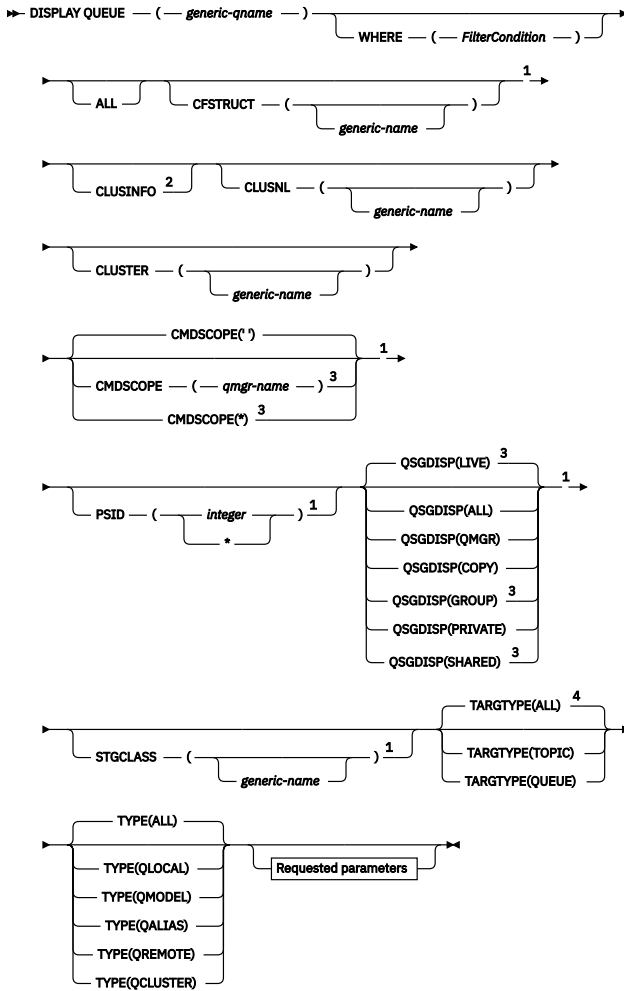
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 804](#)
- [“Parameter descriptions for DISPLAY QUEUE” on page 804](#)
- [“Requested parameters” on page 808](#)

**Synonym: DIS Q**

## DISPLAY QUEUE



### Requested parameters

ACCTQ
ALTDAT
ALTTIME
BOQNAME
BOTHRESH
CAPEXPY 5
CLCHNAME
CLUSDATE
CLUSQMgr
CLUSQT
CLUSTIME
CLWLPRTY
CLWLRANK
CLWLUSEQ
CRDATE
CRTIME
CURDEPTH
CUSTOM
DEFBIND
DEFPRESP
DEFPRTY
DEFPST
DEFREADA
DEFSOPT
DEFTYPE
DESCR
DISTL 5
GET
HARDENBO
IMGRCOVQ 5
INDXTYPE 1
INITQ
IPPROCS
MAXDEPTH
MAXFSIZE 6
MAXMSGL
MONQ
MSGDLVSQ
NPMCLASS
OPPROCS
PROCESS
PROPCTL
PUT
QDEPTHHI
QDEPTHLO
QDPHIEV
QDPLEV
QDPMAXEV
QMID
QSVCEV
QSVCINT
QTYPE
RETINTVL
RNAME
RQMNAME
SCOPE 6
SHARE
STATQ
STREAMQ
STRMQOS
TARGET
TARGETYPE
TPIPE 1
TRIGDATA
TRIGDPH
TRIGGER
TRIGMPRI
TRIGTYPE
USAGE
XMITQ

Notes:

<sup>1</sup> Valid only on z/OS.

<sup>2</sup> On z/OS, you cannot issue this from CSQINP2.

<sup>3</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

<sup>4</sup> Valid only on an alias queue.

<sup>5</sup> Not valid on z/OS.

<sup>6</sup> Not valid on z/OS or IBM i.

## Usage notes

1. You can use the following commands (or their synonyms) as an alternative way to display these attributes.

- **DISPLAY QALIAS**
- **DISPLAY QCLUSTER**
- **DISPLAY QLOCAL**
- **DISPLAY QMODEL**
- **DISPLAY QREMOTE**

These commands produce the same output as the **DISPLAY QUEUE TYPE** (*queue-type*) command. If you enter the commands this way, do not use the **TYPE** parameter.

2. **z/OS** On z/OS, the channel initiator must be running before you can display information about cluster queues (using **TYPE (QCLUSTER)** or the **CLUSINFO** parameter).
3. The command might not show every clustered queue in the cluster when issued on a partial repository, because the partial repository only knows about a queue once it has tried to use it.

## Parameter descriptions for DISPLAY QUEUE

You must specify the name of the queue definition you want to display. This can be a specific queue name or a generic queue name. By using a generic queue name, you can display either:

- All queue definitions
- One or more queues that match the specified name

### *queue-name*

The local name of the queue definition to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk \* matches all queues with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all queues.

### WHERE

Specify a filter condition to display only those queues that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Almost any parameter that can be used to display attributes for this **DISPLAY** command. However, you cannot use the **z/OS** **CMDSCOPE**, **QDPHIEV**, **QDPLOEV**, **QDPMAXEV**, **z/OS** **QSGDISP**, or **QSVCIIEV** parameters as filter keywords. You cannot use **z/OS** **CFSTRUCT**, **CLUSTER**, **z/OS** **PSID**, **z/OS** **STGCLASS**, or **CLUSNL** if these are also used to select queues. Queues of a type for which the filter keyword is not a valid attribute are not displayed.

#### **operator**

This is used to determine whether a queue satisfies the filter value on the given filter keyword. The operators are:

##### **LT**

Less than

##### **GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value QALIAS on the CLUSQT parameter), you can only use EQ or NE. For the parameters HARDENBO, SHARE, and TRIGGER, use either EQ YES or EQ NO.


- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**ALL**

Specify this to display all the attributes. If this parameter is specified, any attributes that are also requested specifically have no effect; all attributes are still displayed.

On all platforms, this is the default if you do not specify a generic name and do not request any specific attributes.

 On z/OS this is also the default if you specify a filter condition using the WHERE parameter, but on other platforms only requested attributes are displayed.


 **CFSTRUCT ( *generic-name* )**

This parameter is optional and limits the information displayed to those queues where the value of the coupling facility structure is specified in brackets.

The value can be a generic name. If you do not enter a value for this parameter, **CFSTRUCT** is treated as a requested parameter.

**CLUSINFO**

This requests that, in addition to information about attributes of queues defined on this queue manager, information about these and other queues in the cluster that match the selection criteria is displayed. In this case, there might be multiple queues with the same name displayed. The cluster information is obtained from the repository on this queue manager.


 Note that, on z/OS, you cannot issue DISPLAY QUEUE CLUSINFO commands from CSQINP2.

**CLUSNL ( *generic-name* )**

This is optional, and limits the information displayed if entered with a value in brackets:

- For queues defined on the local queue manager, only those with the specified cluster list. The value can be a generic name. Only queue types for which **CLUSNL** is a valid parameter are restricted in this way; other queue types that meet the other selection criteria are displayed.
- For cluster queues, only those belonging to clusters in the specified cluster list if the value is not a generic name. If the value is a generic name, no restriction is applied to cluster queues.

If you do not enter a value to qualify this parameter, it is treated as a requested parameter, and cluster list information is returned about all the queues displayed.

**Note:**  If the disposition requested is SHARED, CMDSCOPE must be blank or the local queue manager.

### **CLUSTER ( *generic-name* )**

This is optional, and limits the information displayed to queues with the specified cluster name if entered with a value in brackets. The value can be a generic name. Only queue types for which **CLUSTER** is a valid parameter are restricted in this way by this parameter; other queue types that meet the other selection criteria are displayed.

If you do not enter a value to qualify this parameter, it is treated as a requested parameter, and cluster name information is returned about all the queues displayed.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if QSGDISP is set to GROUP or SHARED.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### ***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use **CMDSCOPE** as a filter keyword.

### **PSID ( *integer* )**

The identifier of the page set where a queue resides. This is optional. Specifying a value limits the information displayed to queues that have an active association to the specified page set. The value consists of two numeric characters, in the range 00 - 99. An asterisk \* on its own specifies all page set identifiers. If you do not enter a value, page set information is returned about all the queues displayed.

The page set identifier is displayed only if there is an active association of the queue to a page set, that is, after the queue has been the target of an MQPUT request. The association of a queue to a page set is not active when:

- The queue is just defined
- The STGCLASS attribute of the queue is altered, and there is no subsequent MQPUT request to the queue
- The queue manager is restarted and there are no messages on the queue

This parameter is valid only on z/OS.

Specifies the disposition of the objects for which information is to be displayed. Values are:

**LIVE**

This is the default value and displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY). If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, also display information for objects defined with QSGDISP(SHARED).

**ALL**

Display information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP) or QSGDISP(SHARED).

In a shared queue manager environment:

```
DISPLAY QUEUE(name) CMDSCOPE(*) QSGDISP(ALL)
```

The command lists objects matching name in the queue sharing group, without duplicating those in the shared repository.

**COPY**

Display information only for objects defined with QSGDISP(COPY).

**GROUP**

Display information only for objects defined with QSGDISP(GROUP). This is allowed only if there is a shared queue manager environment.

**PRIVATE**

Display information only for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

**QMGR**

Display information only for objects defined with QSGDISP(QMGR).

**SHARED**

Display information only for objects defined with QSGDISP(SHARED). This is allowed only in a shared queue manager environment.

**Note:** For cluster queues, this is always treated as a requested parameter. The value returned is the disposition of the real queue that the cluster queue represents.

If QSGDISP(LIVE) is specified or defaulted, or if QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

**Note:** In the QSGDISP(LIVE) case, this occurs only where a shared and a non-shared queue have the same name; such a situation should not occur in a well-managed system.

**QSGDISP** displays one of the following values:

**QMGR**

The object was defined with QSGDISP(QMGR).

**GROUP**

The object was defined with QSGDISP(GROUP).

**COPY**

The object was defined with QSGDISP(COPY).

**SHARED**

The object was defined with QSGDISP(SHARED).

You cannot use **QSGDISP** as a filter keyword.

**z/OS** **STGCLASS ( generic-name )**

This is optional, and limits the information displayed to queues with the storage class specified if entered with a value in brackets. The value can be a generic name.

If you do not enter a value to qualify this parameter, it is treated as a requested parameter, and storage class information is returned about all the queues displayed.

This parameter is valid only on z/OS.

**TARGETTYPE ( target-type )**

This is optional and specifies the target type of the alias queue you want to be displayed.

**TYPE ( queue-type )**

This is optional, and specifies the type of queues you want to be displayed. If you specify ALL, which is the default value, all queue types are displayed; this includes cluster queues if CLUSINFO is also specified.

As well as ALL, you can specify any of the queue types allowed for a **DEFINE** command: QALIAS, QLOCAL, QMODEL, QREMOTE, or their synonyms, as follows:

**QALIAS**

Alias queues

**QLOCAL**

Local queues

**QMODEL**

Model queues

**QREMOTE**

Remote queues

You can specify a queue type of QCLUSTER to display only cluster queue information. If QCLUSTER is specified, any selection criteria specified by the CFSTRUCT, STGCLASS, or PSID parameters are ignored. Note that you cannot issue **DISPLAY QUEUE TYPE(QCLUSTER)** commands from CSQINP2.

**Multi** On Multiplatforms, QTYPE ( type ) can be used as a synonym for this parameter.

The queue name and queue type **z/OS** (and, on z/OS, the queue disposition) are always displayed.

**Requested parameters**

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

Most parameters are relevant only for queues of a particular type or types. Parameters that are not relevant for a particular type of queue cause no output, nor is an error raised.

The following table shows the parameters that are relevant for each type of queue. There is a brief description of each parameter after the table, but for more information, see the **DEFINE** command for each queue type.

<i>Table 173. Parameters that can be returned by the <b>DISPLAY QUEUE</b> command</i>					
	<b>Local queue</b>	<b>Model queue</b>	<b>Alias queue</b>	<b>Remote queue</b>	<b>Cluster queue</b>
<u>ACCTQ</u>	✓	✓	N/A	N/A	N/A
<u>ALTDAT</u>	✓	✓	✓	✓	✓
<u>ALTTIME</u>	✓	✓	✓	✓	✓
<u>BOQNAME</u>	✓	✓	N/A	N/A	N/A
<u>BOTHRESH</u>	✓	✓	N/A	N/A	N/A







Table 173. Parameters that can be returned by the **DISPLAY QUEUE** command (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
<u>CAPEXPRT</u>	✓	✓	✓	✓	✓
<u>CFSTRUCT</u>	✓	✓	N/A	N/A	N/A
<u>CLCHNAME</u>	✓	✓	N/A	N/A	N/A
<u>CLUSDATE</u>	N/A	N/A	N/A	N/A	✓
<u>CLUSNL</u>	✓	N/A	✓	✓	N/A
<u>CLUSQMGR</u>	N/A	N/A	N/A	N/A	✓
<u>CLUSQT</u>	N/A	N/A	N/A	N/A	✓
<u>CLUSTER</u>	✓	N/A	✓	✓	✓
<u>CLUSTIME</u>	N/A	N/A	N/A	N/A	✓
<u>CLWLPRTY</u>	✓	N/A	✓	✓	✓
<u>CLWLRANK</u>	✓	N/A	✓	✓	✓
<u>CLWLUSEQ</u>	✓	N/A	N/A	N/A	N/A
<u>CRDATE</u>	✓	✓	N/A	N/A	N/A
<u>CRTIME</u>	✓	✓	N/A	N/A	N/A
<u>CURDEPTH</u>	✓	N/A	N/A	N/A	N/A
<u>CUSTOM</u>	✓	✓	✓	✓	✓
<u>DEFBIND</u>	✓	N/A	✓	✓	✓
<u>DEFPRESP</u>	✓	✓	✓	✓	✓
<u>DEFPRTY</u>	✓	✓	✓	✓	✓
<u>DEFPSIST</u>	✓	✓	✓	✓	✓
<u>DEFREADA</u>	✓	✓	✓	N/A	N/A
<u>DEFLOPT</u>	✓	✓	N/A	N/A	N/A
<u>DEFTYPE</u>	✓	✓	N/A	N/A	N/A
<u>DESCR</u>	✓	✓	✓	✓	✓
<u>DISTL</u>	✓	✓	N/A	N/A	N/A
<u>GET</u>	✓	✓	✓	N/A	N/A
<u>HARDENBO</u>	✓	✓	N/A	N/A	N/A
<u>IMGRCOVQ</u>	✓	✓	N/A	N/A	N/A

Table 173. Parameters that can be returned by the **DISPLAY QUEUE** command (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
<u>INDXTYPE</u>	✓	✓	N/A	N/A	N/A
<u>INITQ</u>	✓	✓	N/A	N/A	N/A
<u>IPPROCS</u>	✓	N/A	N/A	N/A	N/A
<u>MAXDEPTH</u>	✓	✓	N/A	N/A	N/A
<u>MAXFSIZE</u>	✓	✓	N/A	N/A	N/A
<u>MAXMSGL</u>	✓	✓	N/A	N/A	N/A
<u>MONQ</u>	✓	✓	N/A	N/A	N/A
<u>MSGDLVSQ</u>	✓	✓	N/A	N/A	N/A
<u>NPMCLASS</u>	✓	✓	N/A	N/A	N/A
<u>OPPROCS</u>	✓		N/A	N/A	N/A
<u>PROCESS</u>	✓	✓	N/A	N/A	N/A
<u>PROPCTL</u>	✓	✓	✓	N/A	N/A
<u>PSID</u>	✓	N/A	N/A	N/A	N/A
<u>PUT</u>	✓	✓	✓	✓	✓
<u>QDEPTHHI</u>	✓	✓	N/A	N/A	N/A
<u>QDEPTHLO</u>	✓	✓	N/A	N/A	N/A
<u>QDPHIEV</u>	✓	✓	N/A	N/A	N/A
<u>QDPLOEV</u>	✓	✓	N/A	N/A	N/A
<u>QDPMAXEV</u>	✓	✓	N/A	N/A	N/A
<u>QMID</u>	N/A	N/A	N/A	N/A	✓
<u>QSGDISP</u>	✓	✓	✓	✓	✓
<u>QSVCIEV</u>	✓	✓	N/A	N/A	N/A
<u>QSVCINT</u>	✓	✓	N/A	N/A	N/A
<u>QTYPE</u>	✓	✓	✓	✓	✓
<u>RETINTVL</u>	✓	✓	N/A	N/A	N/A
<u>RNAME</u>	N/A	N/A	N/A	✓	N/A
<u>RQMNAME</u>	N/A	N/A	N/A	✓	N/A
<u>SCOPE</u>	✓	N/A	✓	✓	N/A

Table 173. Parameters that can be returned by the **DISPLAY QUEUE** command (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
<u>SHARE</u>	✓	✓	N/A	N/A	N/A
<u>STATQ</u>	✓	✓	N/A	N/A	N/A
<u>STGCLASS</u>	✓	✓	N/A	N/A	N/A
  <b>STREAMQ</b>	✓	✓	N/A	N/A	N/A
  <b>STRMQOS</b>	✓	✓	N/A	N/A	N/A
<u>TARGET</u>	N/A	N/A	✓	N/A	N/A
<u>TARGETTYPE</u>	N/A	N/A	✓	N/A	N/A
<u>TPIPE</u>	✓	N/A	N/A	N/A	N/A
<u>TRIGDATA</u>	✓	✓	N/A	N/A	N/A
<u>TRIGDPTH</u>	✓	✓	N/A	N/A	N/A
<u>TRIGGER</u>	✓	✓	N/A	N/A	N/A
<u>TRIGMPRI</u>	✓	✓	N/A	N/A	N/A
<u>TRIGTYPE</u>	✓	✓	N/A	N/A	N/A
<u>USAGE</u>	✓	✓	N/A	N/A	N/A
<u>XMITQ</u>	N/A	N/A	N/A	✓	N/A

**ACCTQ**

Whether accounting (on z/OS, thread-level and queue-level accounting) data collection is to be enabled for the queue.

**ALTDATE**

The date on which the definition or information was last altered, in the form yyyy-mm-dd.

**ALTTIME**

The time at which the definition or information was last altered, in the form hh.mm.ss.

**BOQNAME**

Backout requeue name.

**BOTHRESH**

Backout threshold.



**CAPEXPY(*integer*)**

The maximum time, expressed in tenths of a second, until a message put on an object handle opened using this object in the resolution path, remains in the system until it becomes eligible for expiry processing.

**Important:**  You cannot use the **CAPEXPY** attribute introduced in IBM MQ 9.3.1 with a cluster if the full repository is on z/OS.

#### **CLCHNAME**

**CLCHNAME** is the generic name of the cluster-sender channels that use this queue as a transmission queue. The attribute specifies which cluster-sender channels send messages to a cluster-receiver channel from this cluster transmission queue.

#### **CLUSDATE**

The date on which the definition became available to the local queue manager, in the form yyyy-mm-dd.

#### **CLUSNL**

The namelist that defines the cluster that the queue is in.

#### **CLUSQGR**

The name of the queue manager that hosts the queue.

#### **CLUSQT**

Cluster queue type. This can be:

##### **QALIAS**

The cluster queue represents an alias queue.

##### **QLOCAL**

The cluster queue represents a local queue.

##### **QMGR**

The cluster queue represents a queue manager alias.

##### **QREMOTE**

The cluster queue represents a remote queue.

#### **CLUSTER**

The name of the cluster that the queue is in.

#### **CLUSTIME**

The time at which the definition became available to the local queue manager, in the form hh.mm.ss.

#### **CLWLPRTY**

The priority of the queue for the purposes of cluster workload distribution.

#### **CLWLRANK**

The rank of the queue for the purposes of cluster workload distribution.

#### **CLWLUSEQ**

Whether puts are allowed to other queue definitions apart from local ones.

#### **CRDATE**


The date on which the queue was defined (in the form yyyy-mm-dd).

#### **CRTIME**

The time at which the queue was defined (in the form hh.mm.ss).

#### **CURDEPTH**

Current depth of queue.

 On z/OS, **CURDEPTH** is returned as zero for queues defined with a disposition of GROUP. It is also returned as zero for queues defined with a disposition of SHARED if the CF structure that they use is unavailable or has failed.

Messages put on a queue count toward the current depth as they are put. Messages got from a queue do not count toward the current depth. This is true whether operations are done under syncpoint or not. Commit has no effect on current depth. Therefore:

- Messages put under syncpoint (but not yet committed) are included in the current depth.
- Messages got under syncpoint (but not yet committed) are not included in the current depth.

## CUSTOM

This attribute is reserved for the configuration of new features before separate attributes have been introduced. It can contain the values of zero or more attributes as pairs of attribute name and value in the form NAME (VALUE).

## DEFBIND

Default message binding.

## DEFPRESP

Default put response; defines the behavior that should be used by applications when the put response type in the **MQPMO** options has been set to MQPMO\_RESPONSE\_AS\_Q\_DEF.

## DEFPRTY

Default priority of the messages put on the queue.

## DEFPSIST

Whether the default persistence of messages put on this queue is set to NO or YES. NO means that messages are lost across a restart of the queue manager.

## DEFREADA

This specifies the default read ahead behavior for non-persistent messages delivered to the client.

## DEFSOPT

Default share option on a queue opened for input.

## DEFTYPE


Queue definition type. This can be:

- PREDEFINED (Predefined)

The queue was created with a **DEFINE** command, either by an operator or by a suitably authorized application sending a command message to the service queue.


- PERMDYN (Permanent dynamic)

Either the queue was created by an application issuing MQOPEN with the name of a model queue specified in the object descriptor (MQOD), or (if this is a model queue) this determines the type of dynamic queue that can be created from it.

 On z/OS the queue was created with QSGDISP (QMGR).

- TEMPDYN (Temporary dynamic)

Either the queue was created by an application issuing MQOPEN with the name of a model queue specified in the object descriptor (MQOD), or (if this is a model queue) this determines the type of dynamic queue that can be created from it.

 On z/OS the queue was created with QSGDISP (QMGR).

- SHAREDYN

A permanent dynamic queue was created when an application issued an MQOPEN API call with the name of this model queue specified in the object descriptor (MQOD).

On z/OS, in a queue sharing group environment, the queue was created with QSGDISP (SHARED).

## DESCR

Descriptive comment.

## **DISTL**

Whether distribution lists are supported by the partner queue manager. Supported only on Multiplatforms.

## GET

Whether the queue is enabled for gets.

## HARDENBO

Whether the back out count is hardened to ensure that the count of the number of times that a message has been backed out is accurate.

**Note:** This parameter affects only IBM MQ for z/OS. It can be set and displayed on other platforms but has no effect.

### **IMGRCOVQ**

Whether a local or permanent dynamic queue object is recoverable from a media image if linear logging is being used.

**Note:**  This parameter is not valid on IBM MQ for z/OS.

### **INDXTYPE**


Index type (supported only on z/OS).

### **INITQ**

Initiation queue name.

### **IPPROCS**

Number of applications that are currently connected to the queue to get messages from the queue.

 On z/OS, **IPPROCS** is returned as zero for queues defined with a disposition of GROUP. With a disposition of SHARED, only the handles for the queue manager sending back the information are returned, not the information for the whole group.

### **MAXDEPTH**

Maximum depth of queue.

### **MAXFSIZE**

The size, in megabytes, of the queue file displayed.

The default value for this attribute is DEFQFS, which stands for *default queue file size* and equates to a hard-coded value of 2,088,960 MB.

### **MAXMSGL**

Maximum message length.

### **MONQ**

Online monitoring data collection.

### **MSGDLVSQ**


Message delivery sequence.

### **NPMCLASS**

Level of reliability assigned to non-persistent messages that are put to the queue.

### **OPPROCS**

Number of applications that are currently connected to the queue to put messages on the queue.

 On z/OS, **OPPROCS** is returned as zero for queues defined with a disposition of GROUP. With a disposition of SHARED, only the handles for the queue manager sending back the information are returned, not the information for the whole group.

### **PROCESS**

Process name.

### **PROPCTL**

Property control attribute.

This parameter is applicable to Local, Alias and Model queues.

This parameter is optional.

Specifies how message properties are handled when messages are retrieved from queues using the MQGET call with the MQGMO\_PROPERTIES\_AS\_Q\_DEF option.

Permissible values are:

**ALL**

To contain all the properties of the message, except those contained in the message descriptor (or extension), select **ALL**. The **ALL** value enables applications that cannot be changed to access all the message properties from **MQRFH2** headers.

**COMPAT**

If the message contains a property with a prefix of **mcd.**, **jms.**, **usr.**, or **mqext.**, all message properties are delivered to the application in an **MQRFH2** header. Otherwise all properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

This is the default value; it allows applications which expect JMS related properties to be in an **MQRFH2** header in the message data to continue to work unmodified.

**FORCE**

Properties are always returned in the message data in an **MQRFH2** header regardless of whether the application specifies a message handle.

A valid message handle supplied in the **MsgHandle** field of the **MQGMO** structure on the **MQGET** call is ignored. Properties of the message are not accessible via the message handle.

**NONE**

All properties of the message, except those in the message descriptor (or extension), are removed from the message before the message is delivered to the application.

**PUT**

Whether the queue is enabled for puts.

**QDEPTHHI**

Queue Depth High event generation threshold.

**QDEPTHLO**

Queue Depth Low event generation threshold.

**QDPHIEV**

Whether Queue Depth High events are generated.

You cannot use **QDPHIEV** as a filter keyword.

**QDPLOEV**

Whether Queue Depth Low events are generated.

You cannot use **QDPLOEV** as a filter keyword.

**QDPMAXEV**

Whether Queue Full events are generated.

You cannot use **QDPMAXEV** as a filter keyword.

**QMID**

The internally generated unique name of the queue manager that hosts the queue.

**QSVCIEV**

Whether service interval events are generated.

You cannot use **QSVCIEV** as a filter keyword.

**QSVCIINT**

Service interval event generation threshold.

**QTYPE**

Queue type.

The queue type is always displayed.



On [Multiplatforms](#), **TYPE**(*type*) can be used as a synonym for this parameter.

**RETINTVL**

Retention interval.

**RNAME**

Name of the local queue, as known by the remote queue manager.

**RQMNAME**

Remote queue manager name.

**SCOPE**

Scope of queue definition (not supported on z/OS).

**SHARE**

Whether the queue can be shared.

**STATQ**

Whether statistics data information is to be collected.

**STGCLASS**

Storage class.

 **STREAMQ**

The name of a secondary queue where a copy of each message is put.

 **STRMQOS**

The quality of service to use when delivering messages to the duplicate queue.

**TARGET**


This parameter requests that the base object name of an aliased queue is displayed.

**TARGETTYPE**

This parameter requests that the target (base) type of an aliased queue is displayed.

**TPIPE**

The **TPIPE** names used for communication with OTMA using the IBM MQ - IMS bridge if the bridge is active. This parameter is supported only on z/OS.

 For more information about TPIPEs, see [Controlling the IMS bridge](#).

**TRIGDATA**

Trigger data.

**TRIGDPH**

Trigger depth.

**TRIGGER**

Whether triggers are active.

**TRIGMPRI**

Threshold message priority for triggers.

**TRIGTYPE**

Trigger type.

**USAGE**

Whether the queue is a transmission queue.

**XMITQ**

Transmission queue name.

For more details of these parameters, see [“DEFINE queues” on page 547](#).

**Related concepts**

[Working with model queues](#)

**Related tasks**

[Displaying default object attributes](#)



## DISPLAY SBSTATUS (display subscription status)

Use the MQSC command **DISPLAY SBSTATUS** to display the status of a subscription.

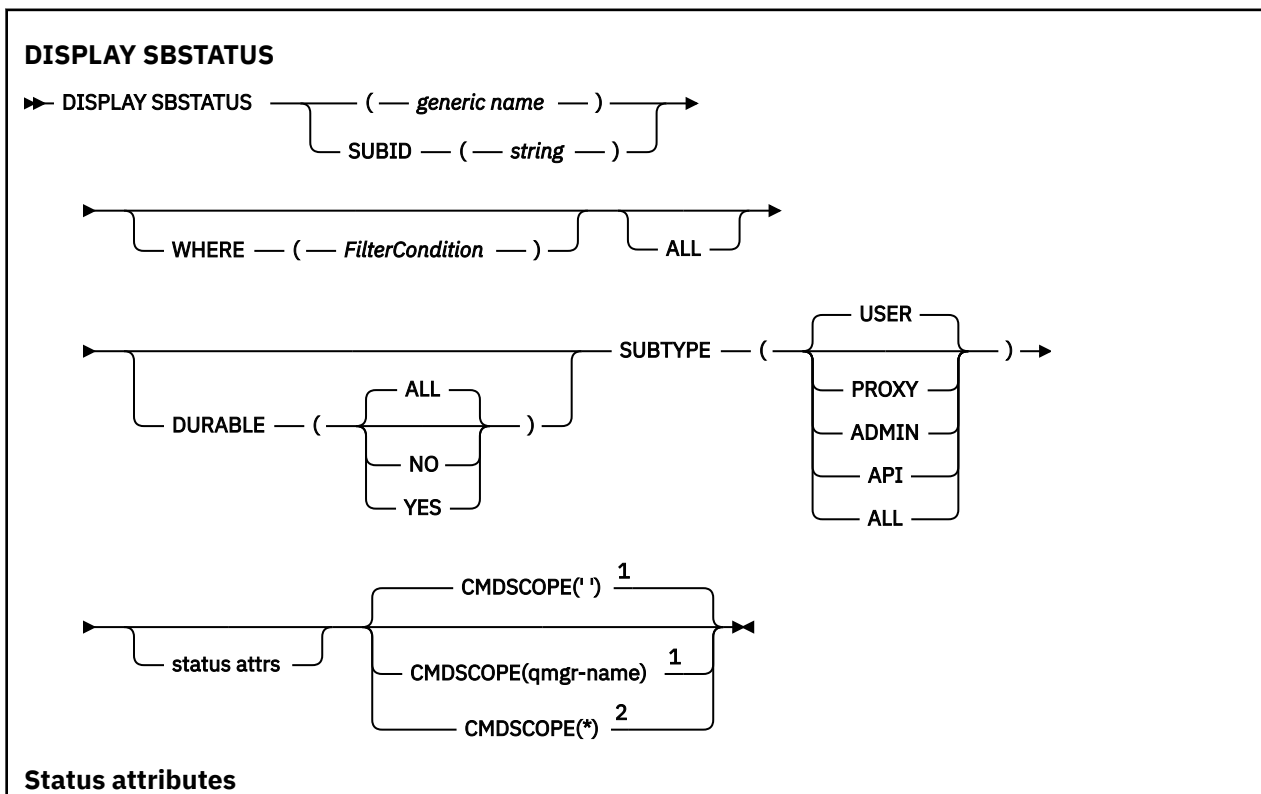
### Using MQSC commands

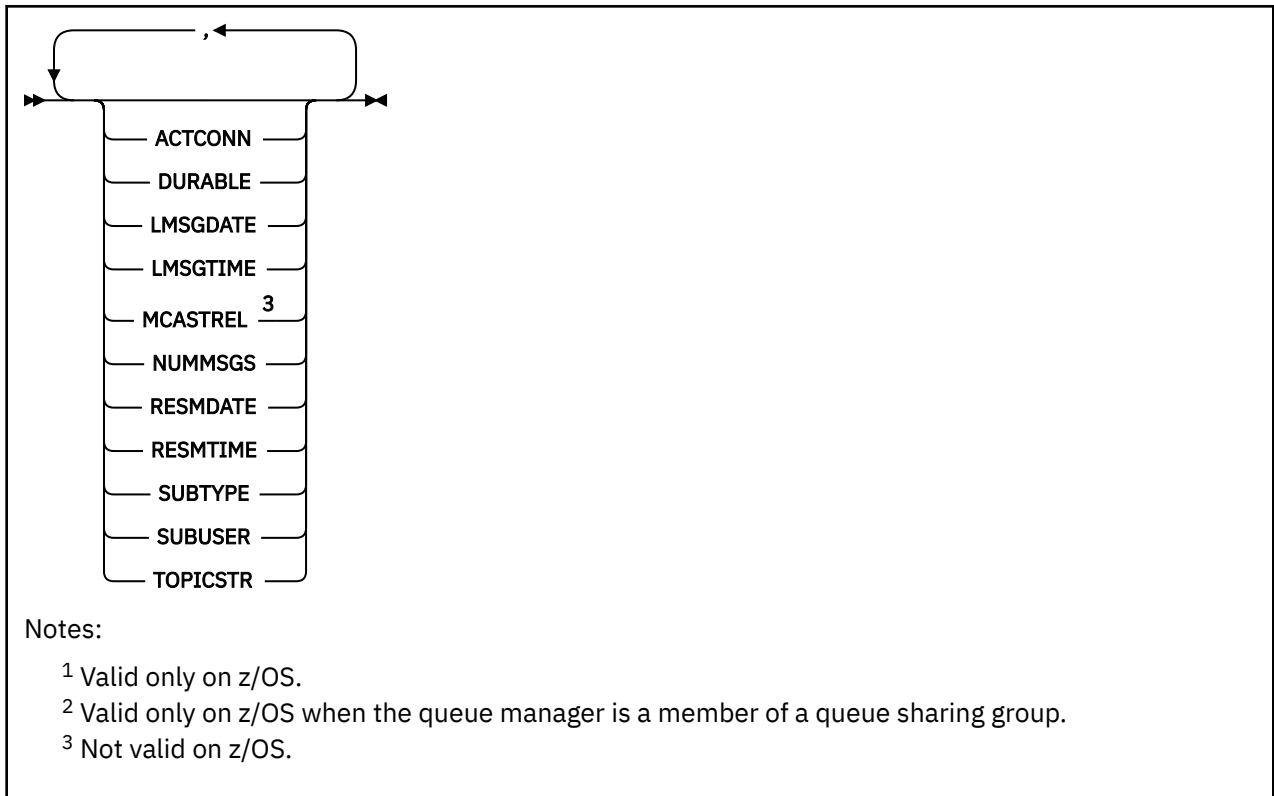
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY SBSTATUS” on page 818](#)
- [“Requested parameters” on page 820](#)

**Synonym: DIS SBSTATUS**





## Parameter descriptions for DISPLAY SBSTATUS

You must specify the name of the subscription definition for which you want to display status information. This can be a specific subscription name or a generic subscription name. By using a generic subscription name, you can display either:

- All subscription definitions
- One or more subscriptions that match the specified name

### **(generic-name)**

The local name of the subscription definition to be displayed. A trailing asterisk (\*) matches all subscriptions with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all subscriptions.

### **WHERE**

Specify a filter condition to display only those subscriptions that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Almost any parameter that can be used to display attributes for this **DISPLAY** command.

**z/OS** However, you cannot use the **CMDSCOPE** parameter as a filter keyword.

Subscriptions of a type for which the filter keyword is not a valid attribute are not displayed.

#### **operator**

This is used to determine whether a subscription satisfies the filter value on the given filter keyword. The operators are:

##### **LT**

Less than

##### **GT**

Greater than

##### **EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value USER on the **SUBTYPE** parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the **SUBUSER** parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**ALL**

Display all the status information for each specified subscription definition. This is the default if you do not specify a generic name, and do not request any specific parameters.

**z/OS** On z/OS this is also the default if you specify a filter condition using the **WHERE** parameter, but on other platforms only, requested attributes are displayed.

**z/OS CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

**CMDSCOPE** must be blank, or the local queue manager, if **QSGDISP** is set to GROUP.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use **CMDSCOPE** as a filter keyword.

**DURABLE**

Specify this attribute to restrict the type of subscriptions which are displayed.

**ALL**

Display all subscriptions.

**NO**

Only information about nondurable subscriptions is displayed.

**YES**

Only information about durable subscriptions is displayed.

**SUBTYPE**

Specify this attribute to restrict the type of subscriptions which are displayed.

**USER**

Displays only **API** and **ADMIN** subscriptions.

**PROXY**

Only system created subscriptions relating to inter-queue manager subscriptions are selected.

**ADMIN**

Only subscriptions that have been created by an administration interface or modified by an administration interface are selected.

**API**

Only subscriptions created by applications using an IBM MQ API call are selected.

**ALL**

All subscription types are displayed (no restriction).

**Requested parameters**

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

**ACTCONN**

Returns the *ConnId* of the *HConn* that currently has this subscription open.

**DURABLE**

A durable subscription is not deleted when the creating application closes its subscription handle.

**NO**

The subscription is removed when the application that created it is closed or disconnected from the queue manager.

**YES**

The subscription persists even when the creating application is no longer running or has been disconnected. The subscription is reinstated when the queue manager restarts.

**LMSGDATE**

The date on which a message was last published to the destination specified by this subscription.

**LMSGTIME**

The time on which a message was last published to the destination specified by this subscription.

**MCASTREL**

Indicator of the reliability of the multicast messages.

The values are expressed as a percentage. A value of 100 indicates that all messages are being delivered without problems. A value less than 100 indicates that some of the messages are experiencing network issues. To determine the nature of these issues you can enable event message generation, using the **COMMEV** parameter of the **COMMINFO** objects, and examine the generated event messages.

The following two values are returned:

- The first value is based on recent activity over a short period.
- The second value is based on activity over a longer period.

If no measurement is available the values are shown as blanks.

**NUMMSGs**

The number of messages put to the destination specified by this subscription since it was created, or since the queue manager was restarted, whichever is more recent. This number might not reflect the total number of messages that are, or have been, available to the consuming application. This is because it might also include publications that were partially processed but then undone by the queue manager due to a publication failure, or publications that were made within syncpoint that were rolled-back by the publishing application.

**RESMDATE**

The date of the most recent **MQSUB** API call that connected to the subscription.

**RESMTIME**

The time of the most recent **MQSUB** API call that connected to the subscription.

**SUBID( *string* )**

The internal, unique key identifying a subscription.

**SUBUSER( *string* )**

The owning user ID of the subscription.

**SUBTYPE**

Indicates how the subscription was created.

**PROXY**

An internally created subscription used for routing publications through a queue manager.

**ADMIN**

Created using the **DEF SUB** MQSC or PCF command. This **SUBTYPE** also indicates that a subscription has been modified using an administrative command.

**API**

Created using an **MQSUB** API call.

**TOPICSTR**

Returns the fully resolved topic string of the subscription.

For more details of these parameters, see [“DEFINE SUB \(create a durable subscription\)” on page 588](#).

**Related tasks**

[Checking messages on a subscription](#)

 **DISPLAY SECURITY (display security settings) on z/OS**

Use the MQSC command **DISPLAY SECURITY** to display the current settings for the security parameters.

**Using MQSC commands on z/OS**

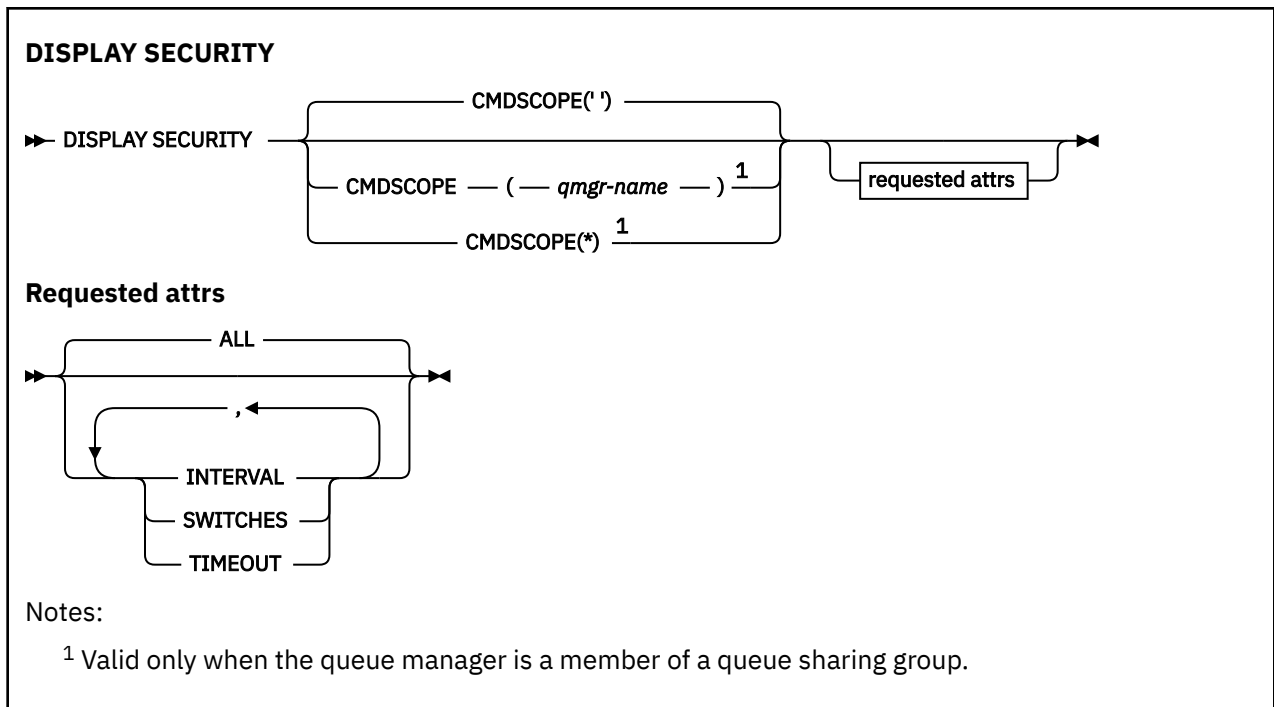
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY SECURITY” on page 822](#)

**Note:** From IBM WebSphere MQ 7.0 onwards, this command is no longer allowed to be issued from CSQINP1 or CSQINP2 on z/OS.

**Synonym:** DIS SEC



## Parameter descriptions for DISPLAY SECURITY

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered. This is the default value.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### ALL

Display the TIMEOUT, INTERVAL, and SWITCHES parameters. This is the default if no requested parameters are specified.

The command also outputs an additional message, either [CSQH037I](#) or [CSQH038I](#), stating whether security is currently using upper or mixed case security classes.

The command also outputs messages [CSQH040I](#) through [CSQH042I](#) showing the connection authentication settings currently in use.

### INTERVAL

Time interval between checks.

### SWITCHES

Display the current setting of the switch profiles.

If the subsystem security switch is off, no other switch profile settings are displayed.

## TIMEOUT

Timeout value.

See “ALTER SECURITY (alter security options) on z/OS” on page 413 for details of the TIMEOUT and INTERVAL parameters.

## Related tasks

[Displaying security status](#)

## **Multi** DISPLAY SERVICE (display service information) on Multiplatforms

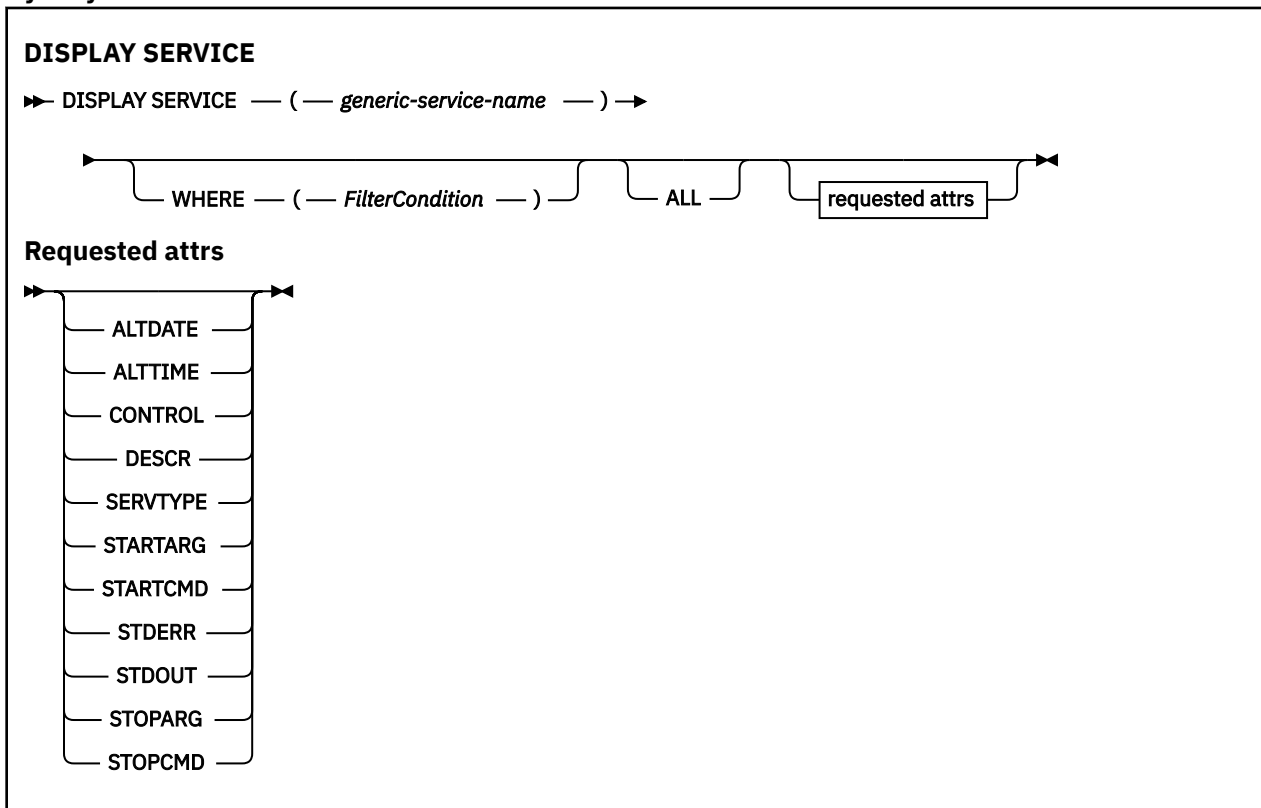
Use the MQSC command DISPLAY SERVICE to display information about a service.

## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Keyword and parameter descriptions for DISPLAY SERVICE” on page 823](#)
- [“Requested parameters” on page 825](#)

### Synonym:



## Keyword and parameter descriptions for DISPLAY SERVICE

You must specify a service for which you want to display information. You can specify a service by using either a specific service name or a generic service name. By using a generic service name, you can display either:

- Information about all service definitions, by using a single asterisk (\*), or
- Information about one or more service that match the specified name.

**( *generic-service-name* )**

The name of the service definition for which information is to be displayed. A single asterisk (\*) specifies that information for all service identifiers is to be displayed. A character string with an asterisk at the end matches all services with the string followed by zero or more characters.

**WHERE**

Specify a filter condition to display information for those listeners that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

**filter-keyword**

Any parameter that can be used to display attributes for this DISPLAY command.

**operator**

This is used to determine whether a listener satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value MANUAL on the CONTROL parameter), you can only use EQ or NE.

.

- A generic value. This is a character string. with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**ALL**

Specify this to display all the service information for each specified service. If this parameter is specified, any parameters that are requested specifically have no effect; all parameters are still displayed.

This is the default if you do not specify a generic identifier, and do not request any specific parameters.



On z/OS this is also the default if you specify a filter condition using the WHERE parameter, but on other platforms only requested attributes are displayed.

## Requested parameters

Specify one or more attributes that define the data to be displayed. The attributes can be specified in any order. Do not specify the same attribute more than once.

### **ALTDATE**

The date on which the definition was last altered, in the form yyyy-mm-dd.

### **ALTTIME**

The time at which the definition was last altered, in the form hh.mm.ss.

### **CONTROL**

How the service is to be started and stopped:

#### **MANUAL**

The service is not to be started automatically or stopped automatically. It is to be controlled by use of the START SERVICE and STOP SERVICE commands.

#### **QMGR**

The service is to be started and stopped at the same time as the queue manager is started and stopped.

#### **STARTONLY**

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

### **DESCR**

Descriptive comment.

### **SERVTYPE**

Specifies the mode in which the service is to run:

#### **COMMAND**

A command service object. Multiple instances of a command service object can be executed concurrently. You cannot monitor the status of command service objects.

#### **SERVER**

A server service object. Only one instance of a server service object can be executed at a time. The status of server service objects can be monitored using the DISPLAY SVSTATUS command.

### **STARTARG**

Specifies the arguments to be passed to the user program at queue manager startup.

### **STARTCMD**

Specifies the name of the program which is to run.

### **STDERR**

Specifies the path to the file to which the standard error (stderr) of the service program is to be redirected.

### **STDOUT**

Specifies the path to the file to which the standard output (stdout) of the service program is to be redirected.

### **STOPARG**

Specifies the arguments to be passed to the stop program when instructed to stop the service.

### **STOPCMD**

Specifies the name of the executable program to run when the service is requested to stop.

For more details of these parameters, see [“DEFINE SERVICE \(create a new service definition\) on Multiplatforms”](#) on page 581.

## z/OS **DISPLAY SMDS (display shared message data sets information) on z/OS**

Use the MQSC command DISPLAY SMDS to display the parameters of existing IBM MQ shared message data sets associated with a specified application structure.

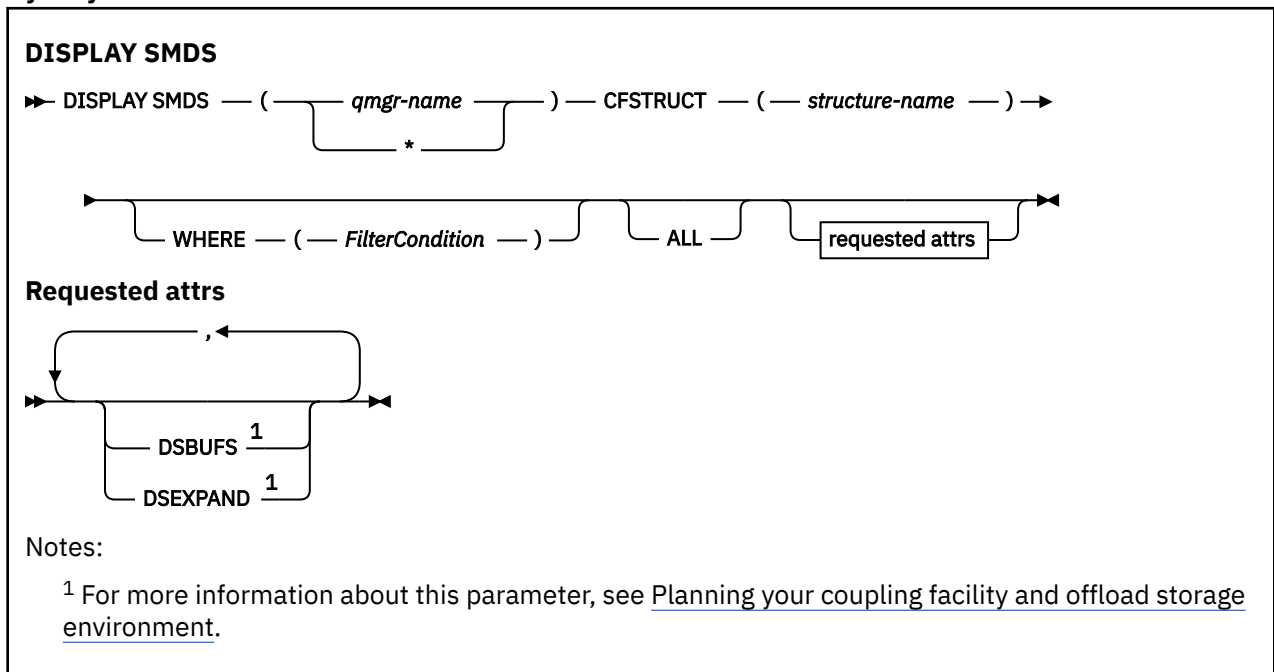
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY SMDS” on page 826](#)
- [“Usage notes for DISPLAY SMDSCONN” on page 829](#)

### Synonym:



### Parameter descriptions for DISPLAY SMDS

The parameter descriptions for the DISPLAY SMDS command.

#### **SMDS(*qmgr-name*|\*)**

Specifies the queue manager for which the shared message data set properties are to be displayed, or an asterisk to display the properties for all shared message data sets associated with the specified CFSTRUCT.

#### **CFSTRUCT( *structure-name* )**

Specify the coupling facility application structure for which the properties of one or more shared message data sets are to be displayed.

#### **WHERE**

Specify a filter condition to display only the SMDS information that satisfies the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### **filter-keyword**

Any parameter that can be used to display attributes for this DISPLAY command.

**operator**

This is used to determine whether a CF application structure satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use any of the operators except LK and NL. However, if the value is one from a possible set of values returnable on a parameter (for example, the value YES on the RECOVER parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

You can only use operators LK or NL for generic values on the DISPLAY SMDS command.

**ALL**

Specify this keyword to display all attributes. If this keyword is specified, any attributes that are requested specifically have no effect; all attributes are still displayed.

This is the default behavior if you do not specify a generic name and do not request any specific attributes.

**Requested parameters for DISPLAY SMDS**

The following information is returned for each selected data set:

**SMDS**

The queue manager name which owns the shared message data set for which properties are being displayed.

**CFSTRUCT**

The coupling facility application structure name.

**DSBUFS**

Displays the override value for the number of buffers to be used by the owning queue manager for accessing shared message data sets for this structure, or DEFAULT if the group value from the CFSTRUCT definition is being used.

## DSEXPAND

Displays the override value (YES or NO) for the data set expansion option, or DEFAULT if the group value from the CFSTRUCT definition is being used.

## z/OS **DISPLAY SMDSCONN (display shared message data sets connection information) on z/OS**

Use the MQSC command DISPLAY SMDSCONN to display status and availability information about the connection between the queue manager and the shared message data sets for the specified CFSTRUCT.

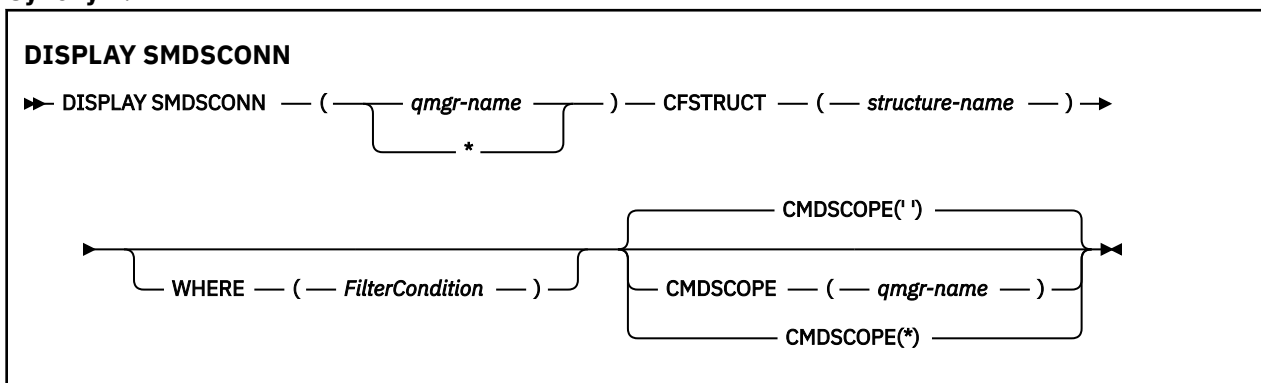
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY SMDSCONN” on page 828](#)
- [“Usage notes for DISPLAY SMDSCONN” on page 829](#)

### Synonym:



## Parameter descriptions for DISPLAY SMDSCONN

The parameter descriptions for the DISPLAY SMDS command.

### SMDSCONN(*qmgr-name*\*)

Specify the queue manager which owns the SMDS for which the connection information is to be displayed, or an asterisk to display the connection information for all shared message data sets associated with the specified CFSTRUCT.

### CFSTRUCT( *structure-name* )

Specify the structure name for which the shared message data set connection information is required.

### WHERE

Specify a filter condition to display only the SMDS connection information that satisfies the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### filter-keyword

Any parameter that can be used to display attributes for this DISPLAY command.

#### operator

This is used to determine whether a CF application structure satisfies the filter value on the given filter keyword. The operators are:

#### LT

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use any of the operators except LK and NL. However, if the value is one from a possible set of values returnable on a parameter (for example, the value YES on the RECOVER parameter), you can only use EQ or NE.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. The characters must be valid for the attribute you are testing. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

You can only use operators LK or NL for generic values on the DISPLAY SMDSCONN command.

**CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered.

This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group. You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

**Usage notes for DISPLAY SMDSCONN**

This command is only supported when the CFSTRUCT definition is currently using the option OFFLOAD(SMDS).

This information indicates whether the queue manager is currently able to allocate and open the data set.

The following results are returned for each selected connection:

**SMDSCONN**

The name of the queue manager which owns the shared message data set for this connection.

**CFSTRUCT**

The name of the coupling facility application structure.

**OPENMODE**

The mode in which the data set is currently open by this queue manager. This is one of the following:

**NONE**

The data set is not currently open.

**READONLY**

The data set is owned by another queue manager and is open for read-only access.

**UPDATE**

The data set is owned by this queue manager and is open for update access.

**RECOVERY**

The data set is open for recovery processing.

**STATUS**

The connection status as seen by this queue manager. This is one of the following:

**CLOSED**

This data set is not currently open.

**OPENING**

This queue manager is currently in the process of opening and validating this data set (including space map restart processing when necessary).

**OPEN**

This queue manager has successfully opened this data set and it is available for normal use.

**CLOSING**

This queue manager is currently in the process of closing this data set, including quiescing normal I/O activity and storing the saved space map if necessary.

**NOTENABLED**

The SMDS definition is not in the ACCESS(ENABLED) state so the data set is not currently available for normal use. This status is only set when the SMDSCONN status does not already indicate some other form of failure.

**ALLOCFAIL**

This queue manager was unable to locate or allocate this data set.

**OPENFAIL**

This queue manager was able to allocate the data set but was unable to open it, so it has now been deallocated.

**STGFAIL**

The data set could not be used because the queue manager was unable to allocate associated storage areas for control blocks, or for space map or header record processing.

**DATAFAIL**

The data set was successfully opened but the data was found to be invalid or inconsistent, or a permanent I/O error occurred, so it has now been closed and deallocated.

This may result in the shared message data set itself being marked as STATUS(FAILED).

**AVAIL**

The availability of this data set connection as seen by this queue manager. This is one of the following:

**NORMAL**

The connection can be used and no error has been detected.

**ERROR**

The connection is unavailable because of an error.

The queue manager may try to enable access again automatically if the error may no longer be present, for example when recovery completes or the status is manually set to RECOVERED.

Otherwise, it can be enabled again using the START SMDSCONN command in order to retry the action which originally failed.

#### **STOPPED**

The connection cannot be used because it has been explicitly stopped using the STOP SMDSCONN command. It can only be made available again by using a START SMDSCONN command to enable it.

#### **EXPANDST**

The data set automatic expansion status. This is one of the following:

##### **NORMAL**

No problem has been noted which would affect automatic expansion.

##### **FAILED**

A recent expansion attempt failed, causing the DSEXPAND option to be set to NO for this specific data set. This status is cleared when ALTER SMDS is used to set the DSEXPAND option back to YES or DEFAULT

##### **MAXIMUM**

The maximum number of extents has been reached, so future expansion is not possible (except by taking the data set out of service and copying it to larger extents).

Note, that the command works only if the structure is currently connected, that is, some shared queues allocated to that structure have been opened.

#### **Related reference**

[“START SMDSCONN \(restart a shared message data set connection\) on z/OS” on page 956](#)

Use the MQSC command START SMDSCONN to enable a previously stopped connection from this queue manager to the specified shared message data sets, allowing them to be allocated and opened again.

[“STOP SMDSCONN \(stop shared message data sets connection\) on z/OS” on page 976](#)

Use the MQSC command STOP SMDSCONN to terminate the connection from this queue manager to one or more specified shared message data sets (causing them to be closed and deallocated) and to mark the connection as STOPPED.

## **DISPLAY STGCLASS (display storage class information) on z/OS**

Use the MQSC command DISPLAY STGCLASS to display information about storage classes.

### **Using MQSC commands on z/OS**

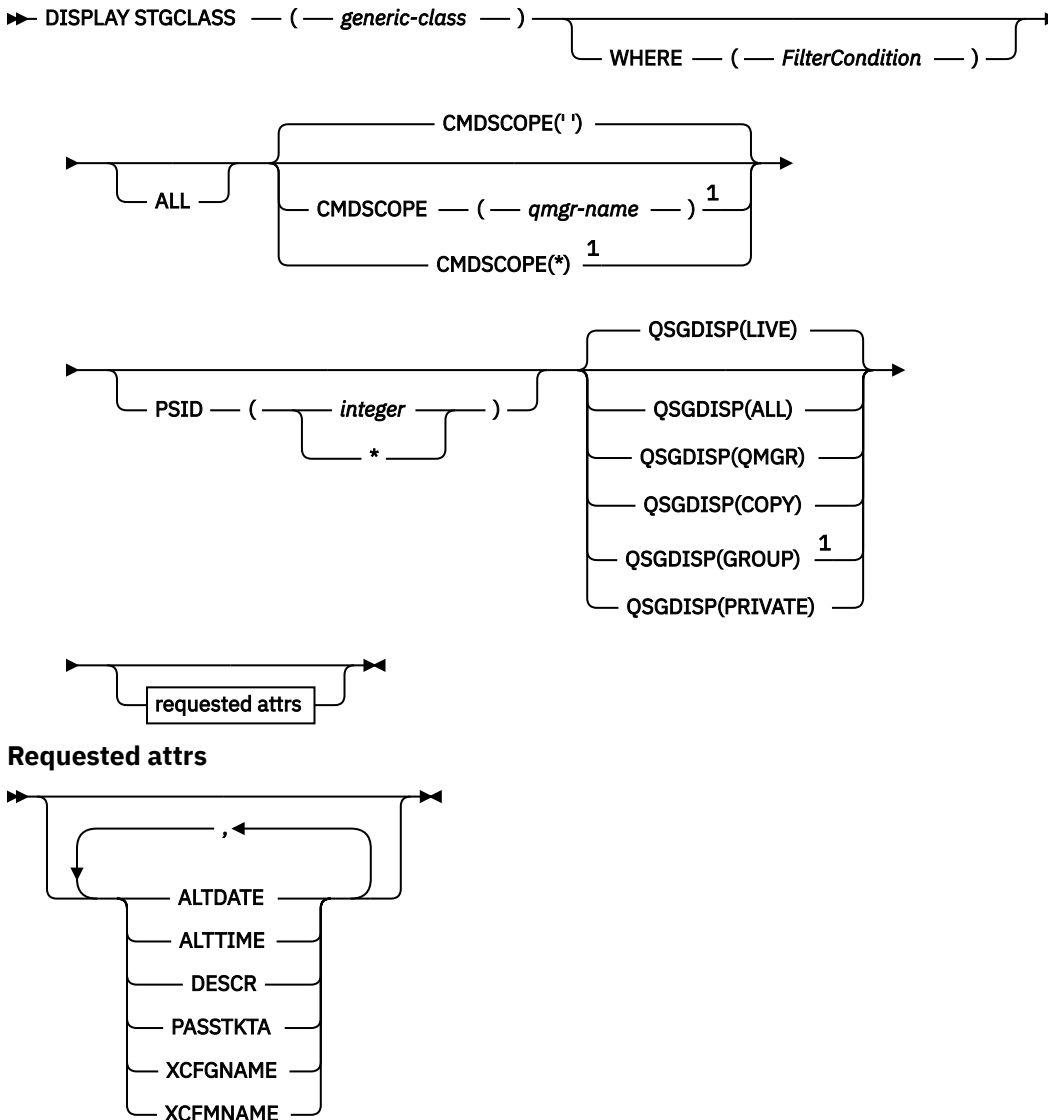
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY STGCLASS” on page 832](#)
- [“Requested parameters” on page 835](#)

**Synonym:** DIS STC

## DISPLAY STGCLASS



### Notes:

<sup>1</sup> Valid only on IBM MQ for z/OS when the queue manager is a member of a queue sharing group.

## Parameter descriptions for DISPLAY STGCLASS

You use DISPLAY STGCLASS to show the page set identifiers that are associated with each storage class.

### (*generic-class*)

Name of the storage class. This is required.

This is 1 through 8 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 through 9.

A trailing asterisk (\*) matches all storage classes with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all storage classes.

### WHERE

Specify a filter condition to display only those storage classes that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:



**filter-keyword**

Almost any parameter that can be used to display attributes for this DISPLAY command. However, you cannot use the CMDSCOPE or QSGDISP parameters as filter keywords. You cannot use PSID as a filter keyword if you also use it to select storage classes.

**operator**

This is used to determine whether a connection satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.  
You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter, you can only use EQ or NE.
- A generic value. This is a character string (such as the character string in the DESCR parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string ABC are listed. Only a single trailing wildcard character (asterisk) is permitted.  
You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**ALL**

Specify this to display all the parameters. If this parameter is specified, any parameters that are also requested specifically have no effect; all parameters are still displayed.

This is the default if you do not specify a generic name, and do not request any specific parameters.

On z/OS this is also the default if you specify a filter condition using the WHERE parameter, but on other platforms only requested attributes are displayed.

**CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

If QSGDISP is set to GROUP, CMDSCOPE must be blank or the local queue manager.

''

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

**\***

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

**PSID( integer )**

The page set identifier that a storage class maps to. This is optional.

The string consists of two numeric characters, in the range 00 through 99. An asterisk (\*) on its own specifies all page set identifiers. See [“DEFINE PSID \(define page set and buffer pool\) on z/OS” on page 545.](#)

**QSGDISP**

Specifies the disposition of the objects for which information is to be displayed. Values are:

**LIVE**

This is the default value and displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

**ALL**

Displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP).

If QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

In a shared queue manager environment, use

```
DISPLAY STGCLASS(generic-class) CMDSCOPE(*) QSGDISP(ALL)
```

to list ALL objects matching

```
name
```

in the queue sharing group without duplicating those in the shared repository.

**COPY**

Display information only for objects defined with QSGDISP(COPY).

**GROUP**

Display information only for objects defined with QSGDISP(GROUP). This is allowed only if there is a shared queue manager environment.

**PRIVATE**

Display information only for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

**QMGR**

Display information only for objects defined with QSGDISP(QMGR).

QSGDISP displays one of the following values:

**QMGR**

The object was defined with QSGDISP(QMGR).

**GROUP**

The object was defined with QSGDISP(GROUP).

**COPY**

The object was defined with QSGDISP(COPY).

You cannot use QSGDISP as a filter keyword.

**Requested parameters**

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

The default, if no parameters are specified (and the ALL parameter is not specified) is the storage class names, their page set identifiers and queue sharing group dispositions are displayed.

**ALTDATE**

The date on which the definition was last altered, in the form yyyy-mm-dd.

**ALLTIME**

The time at which the definition was last altered, in the form hh.mm.ss.

**DESCR**

Descriptive comment.

**PASSTKTA**

The application name used to authenticate IMS bridge passtickets. A blank value indicates that the default batch job profile name is to be used.

**XCFGNAME**

The name of the XCF group that IBM MQ is a member of.

**XCFMNAME**

The XCF member name of the IMS system within the XCF group specified in XCFGNAME.


For more details of these parameters, see [“DEFINE STGCLASS \(define storage class to page set mapping\) on z/OS”](#) on page 585.

**DISPLAY SUB (display subscription information)**

Use the MQSC command **DISPLAY SUB** to display the attributes associated with a subscription.

**Using MQSC commands**

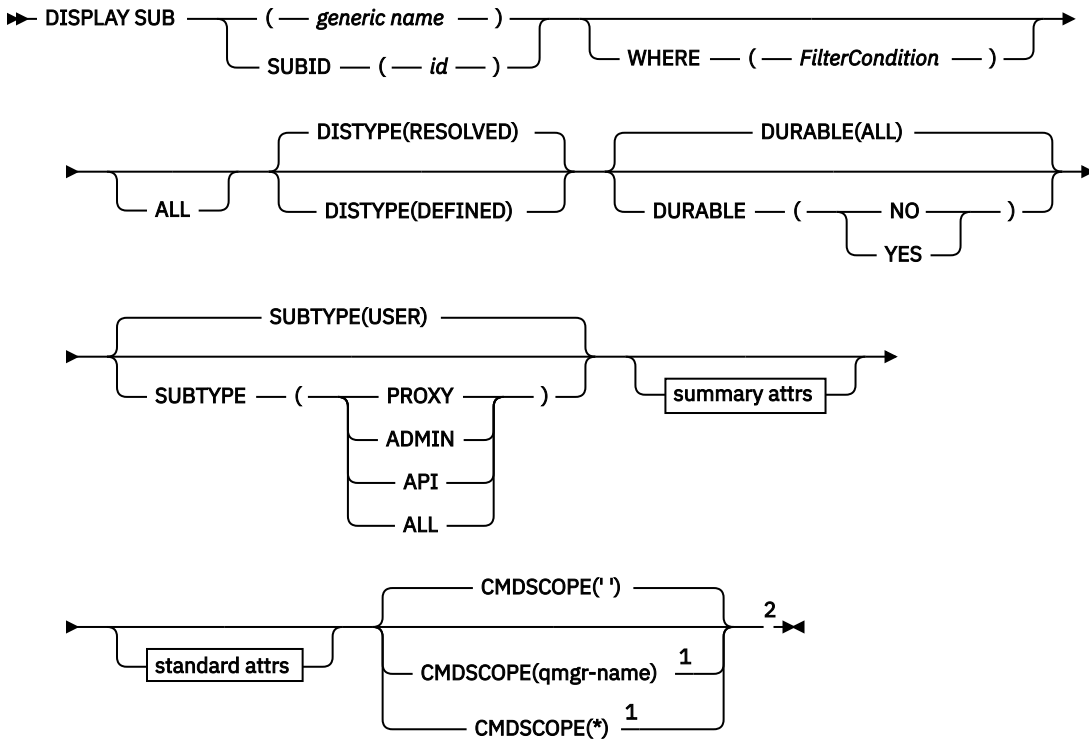
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

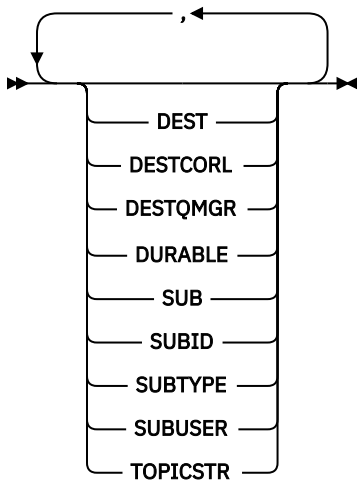
- [Syntax diagram](#)
- [“Usage notes for DISPLAY SUB” on page 837](#)
- [“Parameter descriptions for DISPLAY SUB” on page 837](#)

**Synonym: DIS SUB**

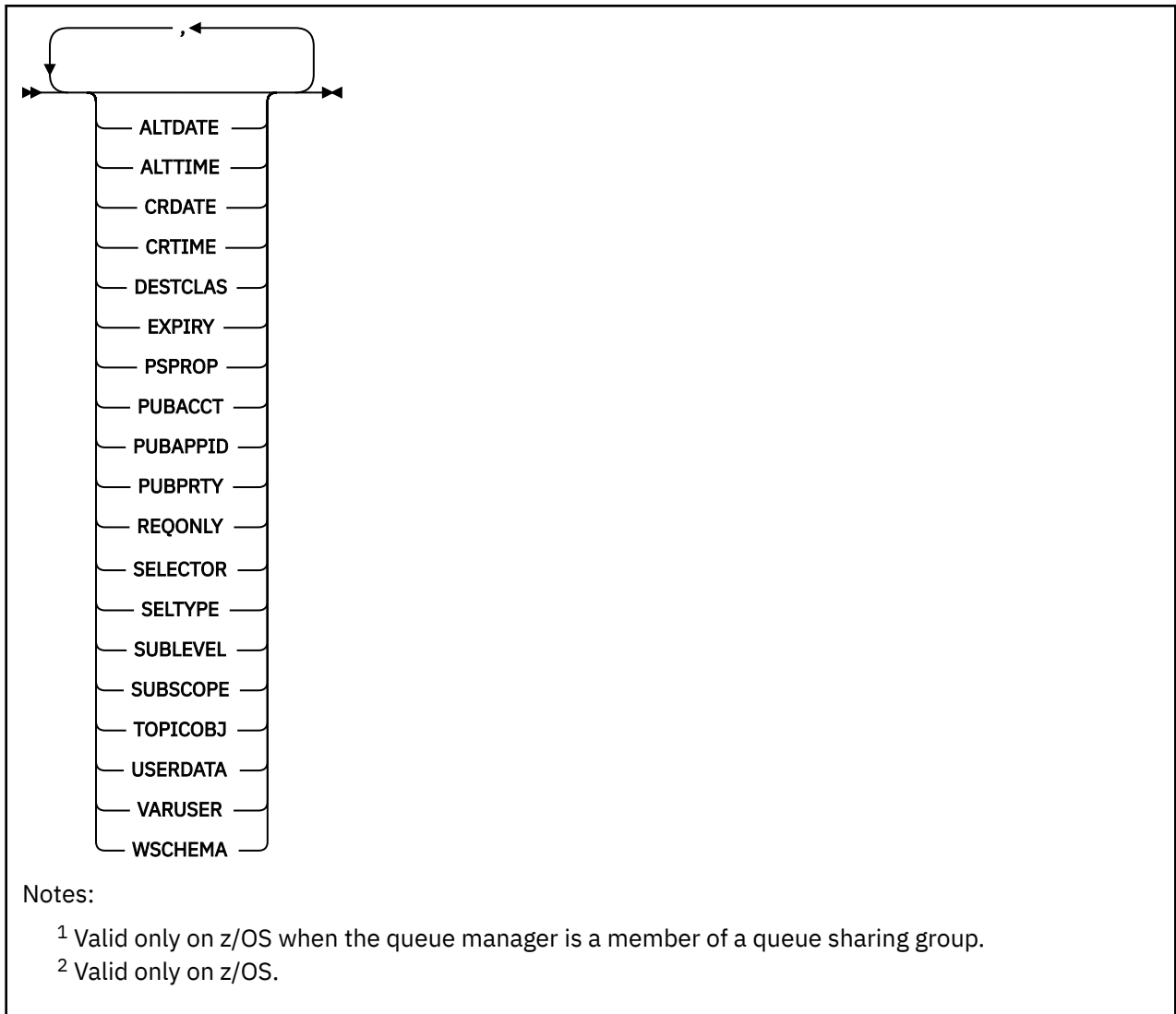
## DISPLAY SUB



### summary attributes



### standard attributes



### Usage notes for DISPLAY SUB

The **TOPICSTR** parameter might contain characters that cannot be translated into printable characters when the command output is displayed.

**z/OS** On z/OS, these non-printable characters are displayed as blanks.

**Multi** On Multiplatforms using runmqsc, these non-printable characters are displayed as dots.

### Parameter descriptions for DISPLAY SUB

You must specify either the name or the identifier of subscription you want to display. This can be a specific subscription name, or SUBID, or a generic subscription name. By using a generic subscription name, you can display either:

- All subscription definitions
- One or more subscriptions that match the specified name

The following forms are valid:

```
DIS SUB(xyz)
DIS SUB SUBID(123)
DIS SUB(xyz*)
```

**(generic-name)**

The local name of the subscription definition to be displayed. A trailing asterisk (\*) matches all subscriptions with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all subscriptions.

**WHERE**

Specify a filter condition to display only those subscriptions that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

**filter-keyword**

Almost any parameter that can be used to display attributes for this DISPLAY command. However, you cannot use the CMDSCOPE parameter as a filter keyword. Subscriptions of a type for which the filter keyword is not a valid attribute are not displayed.

**operator**

This is used to determine whether a subscription satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *filter-value*

**NL**

Does not match a generic string that you provide as a *filter-value*

**filter-value**


The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value QALIAS on the CLUSQT parameter), you can only use EQ or NE. For the parameters HARDENBO, SHARE, and TRIGGER, use either EQ YES or EQ NO.

- A generic value. This is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**Note:**  On z/OS there is a 256 character limit for the filter-value of the MQSC WHERE clause. This limit is not in place for other platforms.

**SUMMARY**

Specify this to display the set of summary attributes that you want displayed.

**ALL**

Specify this to display all the attributes.

If this parameter is specified, any attributes that are also requested specifically have no effect; all attributes are still displayed.

This is the default if you do not specify a generic name and do not request any specific attributes.

**ALTDATE( *string* )**

The date of the most recent **MQSUB** or **ALTER SUB** command that modified the properties of the subscription.

**ALTTIME( *string* )**

The time of the most recent **MQSUB** or **ALTER SUB** command that modified the properties of the subscription.

 **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of setting this value is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

**CRDATE( *string* )**

The date of the first **MQSUB** or **DEF SUB** command that created this subscription.

**CRTIME( *string* )**

The time of the first **MQSUB** or **DEF SUB** command that created this subscription.

**DEST(*string*)**

The destination for messages published to this subscription; this parameter is the name of a queue.

**DESTCLAS**

System managed destination.

**PROVIDED**

The destination is a queue.

**MANAGED**

The destination is managed.

**DESTCORL(*string*)**

The **CorrelId** used for messages published to this subscription.

A blank value (default) results in a system generated correlation identifier being used.

If set to ' 00 ' (48 zeros) the **CorrelId** set by the publishing application will be maintained in the copy of the message delivered to the subscription, unless messages are propagated across a publish/subscribe hierarchy.

If this byte string is enclosed in quotation marks, characters in the range A-F must be specified in uppercase.

**Note:** It is not possible to set the DESTCORL property programmatically with JMS.

**DESTQMGR(string)**

The destination queue manager for messages published to this subscription.

**DISTYPE**

Controls the output returned in the **TOPICSTR** and **TOPICOBJ** attributes.

**RESOLVED**

Returns the resolved (full) topic string in the **TOPICSTR** attribute. The value of the **TOPICOBJ** attribute is also returned. This is the default value.

**DEFINED**

Returns the values of the **TOPICOBJ** and **TOPICSTR** attributes provided when the subscription was created. The **TOPICSTR** attribute will contain the application part of the topic string only. You can use the values returned with **TOPICOBJ** and **TOPICSTR** to fully re-create the subscription by using **DISTYPE (DEFINED)**.

**DURABLE**

A durable subscription is not deleted when the creating application closes its subscription handle.

**ALL**

Display all subscriptions.

**NO**

The subscription is removed when the application that created it, is closed or disconnected from the queue manager.

**YES**

The subscription persists even when the creating application is no longer running or has been disconnected. The subscription is reinstated when the queue manager restarts.

**EXPIRY**

The time to expiry of the subscription object from the creation date and time.

**(integer)**

The time to expiry, in tenths of a second, from the creation date and time.

**UNLIMITED**

There is no expiry time. This is the default option supplied with the product.

**PSPROP**

The manner in which publish subscribe related message properties are added to messages sent to this subscription.

**NONE**

Do not add publish subscribe properties to the message.

**COMPAT**

Publish/subscribe properties are added within an MQRFH version 1 header unless the message was published in PCF format.

**MSGPROP**

Publish/subscribe properties are added as message properties.

**RFH2**

Publish/subscribe properties are added within an MQRFH version 2 header.

**PUBACCT(string)**

Accounting token passed by the subscriber, for propagation into messages published to this subscription in the AccountingToken field of the MQMD.

If this byte string is enclosed in quotation marks, characters in the range A-F must be specified in uppercase.

**PUBAPPID(string)**

Identity data passed by the subscriber, for propagation into messages published to this subscription in the ApplIdentityData field of the MQMD.

**PUBPRTY**

The priority of the message sent to this subscription.



**ASPUB**

Priority of the message sent to this subscription is taken from the priority supplied in the published message.

**ASQDEF**

Priority of the message sent to this subscription is taken from the default priority of the queue defined as a destination.

**(integer)**

An integer providing an explicit priority for messages published to this subscription.

**REQONLY**

Indicates whether the subscriber polls for updates using the MQSUBRQ API call, or whether all publications are delivered to this subscription.

**NO**

All publications on the topic are delivered to this subscription. This is the default value.

**YES**

Publications are only delivered to this subscription in response to an MQSUBRQ API call.

This parameter is equivalent to the subscribe option MQSO\_PUBLICATIONS\_ON\_REQUEST.

**SELECTOR(string)**

A selector that is applied to messages published to the topic.

**SELTYPE**

The type of selector string that has been specified.

**NONE**

No selector has been specified.

**STANDARD**

The selector references only the properties of the message, not its content, using the standard IBM MQ selector syntax. Selectors of this type are to be handled internally by the queue manager.

**EXTENDED**

The selector uses extended selector syntax, typically referencing the content of the message.

Selectors of this type cannot be handled internally by the queue manager; extended selectors can be handled only by another program such as IBM Integration Bus.

**SUB(string)**

The application's unique identifier for a subscription.

**SUBID(string)**

The internal, unique key identifying a subscription.

**SUBLEVEL(integer)**

The level within the subscription hierarchy at which this subscription is made. The range is zero through 9.

**SUBSCOPE**

Determines whether this subscription is forwarded to other queue managers, so that the subscriber receives messages published at those other queue managers.

**ALL**

The subscription is forwarded to all queue managers directly connected through a publish/subscribe collective or hierarchy.

**QMGR**

The subscription forwards messages published on the topic only within this queue manager.

**Note:** Individual subscribers can only restrict **SUBSCOPE**. If the parameter is set to ALL at topic level, then an individual subscriber can restrict it to QMGR for this subscription. However, if the parameter is set to QMGR at topic level, then setting an individual subscriber to ALL has no effect.

**SUBTYPE**

Indicates how the subscription was created.

**USER**

Displays only **API** and **ADMIN** subscriptions.

**PROXY**

An internally created subscription used for routing publications through a queue manager.

Subscriptions of type PROXY are not modified to ADMIN when alterations are attempted.

**ADMIN**

Created using **DEF SUB** MQSC or PCF command. This **SUBTYPE** also indicates that a subscription has been modified using an administrative command.

**API**

Created using an **MQSUB** API request.

**ALL**

All.

**SUBUSER(string)**

Specifies the user ID that is used for security checks that are performed to ensure that publications can be put to the destination queue associated with the subscription. This ID is either the user ID associated with the creator of the subscription or, if subscription takeover is permitted, the user ID that last took over the subscription. The length of this parameter must not exceed 12 characters.

**TOPICOBJ(string)**

The name of a topic object used by this subscription.

**TOPICSTR(string)**

Returns a topic string, that can contain wildcard characters to match a set of topic strings, for the subscription. The topic string is either the application provided portion only, or fully qualified, depending on the value of **DISTYPE**.

**USERDATA(string)**

Specifies the user data associated with the subscription. The string is a variable length value that can be retrieved by the application on an MQSUB API call and passed in a message sent to this subscription as a message property. The **USERDATA** is stored in the RFH2 header in the mqps folder with the key Sud.

An IBM MQ classes for JMS application can retrieve the subscription user data from the message by using the constant JMS\_IBM\_SUBSCRIPTION\_USER\_DATA. For more information, see [Retrieval of user subscription data](#).

**VARUSER**

Specifies whether a user other than the subscription creator can connect to and take over ownership of the subscription.

**ANY**

Any user can connect to and takeover ownership of the subscription.

**FIXED**

Takeover by another USERID is not permitted.

**WSHEMA**

The schema to be used when interpreting any wildcard characters in the topic string.

**CHAR**

Wildcard characters represent portions of strings.

**TOPIC**

Wildcard characters represent portions of the topic hierarchy.

**Related tasks**

[Displaying attributes of subscriptions](#)

## DISPLAY SVSTATUS (display services status) on Multiplatforms

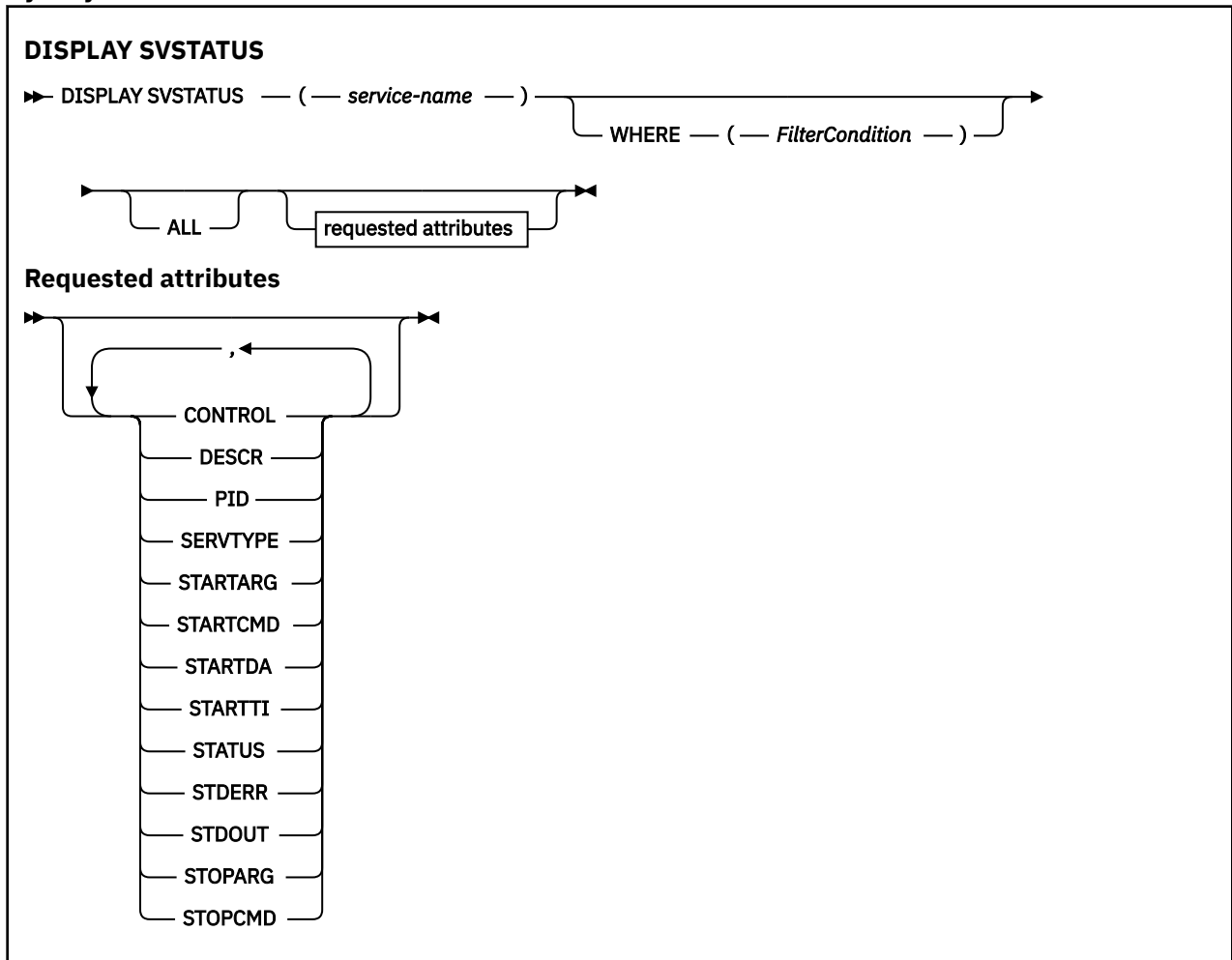
Use the MQSC command **DISPLAY SVSTATUS** to display status information for one or more services. Only services with a **SERVTYPE** of SERVER are displayed.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Keyword and parameter descriptions for DISPLAY SVSTATUS” on page 843](#)
- [“Requested parameters” on page 844](#)

#### Synonym:



### Keyword and parameter descriptions for DISPLAY SVSTATUS

You must specify a service for which you want to display status information. You can specify a service by using either a specific service name or a generic service name. By using a generic service name, you can display either:

- Status information for all service definitions, by using a single asterisk (\*), or
- Status information for one or more services that match the specified name.

**(generic-service-name)**

The name of the service definition for which status information is to be displayed. A single asterisk (\*) specifies that information for all connection identifiers is to be displayed. A character string with an asterisk at the end matches all services with the string followed by zero or more characters.

**WHERE**

Specify a filter condition to display status information for those services that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

**filter-keyword**

Any parameter that can be used to display attributes for this **DISPLAY** command.

**operator**

This is used to determine whether a service satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this can be:

- An explicit value, that is a valid value for the attribute being tested.  
You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter (for example, the value MANUAL on the **CONTROL** parameter), you can only use EQ or NE.
- A generic value. This is a character string with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**ALL**

Display all the status information for each specified service. This is the default if you do not specify a generic name, and do not request any specific parameters.

**Requested parameters**

Specify one or more attributes that define the data to be displayed. The attributes can be specified in any order. Do not specify the same attribute more than once.

**CONTROL**

How the service is to be started and stopped:

**MANUAL**

The service is not to be started automatically or stopped automatically. It is to be controlled by use of the **START SERVICE** and **STOP SERVICE** commands.

**QMGR**

The service is to be started and stopped at the same time as the queue manager is started and stopped.

**STARTONLY**

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**DESCR**

Descriptive comment.

**PID**

The operating system process identifier associated with the service.

**SERVTYPE**

The mode in which the service runs. A service can have a **SERVTYPE** of SERVER or COMMAND, but only services with **SERVTYPE (SERVER)** are displayed by this command.

**STARTARG**

The arguments passed to the user program at startup.

**STARTCMD**

The name of the program being run.

**STARTDA**

The date on which the service was started.

**STARTTI**

The time at which the service was started.

**STATUS**

The status of the process:

**RUNNING**

The service is running.

**STARTING**

The service is in the process of initializing.

**STOPPING**

The service is stopping.

**STDERR**

Destination of the standard error (stderr) of the service program.

**STDOUT**

Destination of the standard output (stdout) of the service program.

**STOPARG**

The arguments to be passed to the stop program when instructed to stop the service.

**STOPCMD**

The name of the executable program to run when the service is requested to stop.

For more information about these parameters, see [“DEFINE SERVICE \(create a new service definition\) on Multiplatforms” on page 581](#).

**Related concepts**

[Working with services](#)

**Related tasks**

[Using a server service object](#)

[Using a command service object](#)

## **z/OS** **DISPLAY SYSTEM (display system information) on z/OS**

Use the MQSC command DISPLAY SYSTEM to display general system parameters and information.

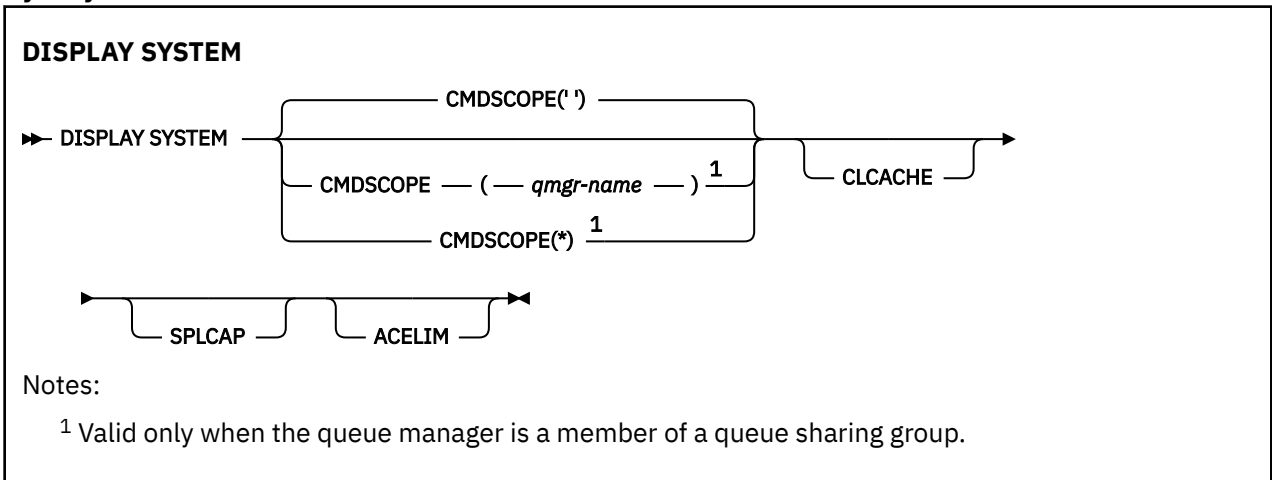
### **Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DISPLAY SYSTEM” on page 846](#)
- [“Parameter descriptions for DISPLAY SYSTEM” on page 847](#)

**Synonym:** DIS SYSTEM



### **Usage notes for DISPLAY SYSTEM**

1. DISPLAY SYSTEM returns a report that shows the initial values of the system parameters and the current values as changed by the SET SYSTEM command:
  - Default user ID for command security checks (CMDUSER).
  - Time in seconds for which queue manager exits can execute during each invocation (EXITLIM).
  - How many started server tasks to use to run queue manager exits (EXITTCB).
  - Number of log records written by IBM MQ between the start of one checkpoint and the next (LOGLOAD).
  - The Measured Usage Pricing property for this queue manager (MULCCAPT). This property is only displayed if the MULCCAPT property is set to REFINED.
  - The OTMA connection parameters (OTMACON).
  - Whether queue manager restart waits until all indexes are built, or completes before all indexes are built (QINDEXBLD).
  - Coded character set identifier for the queue manager (QMCCSID).
  - The queue sharing group parameters (QSGDATA).
  - The RESLEVEL auditing parameter (RESAUDIT).
  - The message routing code assigned to messages not solicited from a specific console (ROUTCODE).
  - Whether SMF accounting data is collected when IBM MQ is started (SMFACCT).
  - Whether SMF statistics are collected when IBM MQ is started (SMFSTAT).

- **V 9.3.0** From IBM MQ for z/OS 9.3.0, the time, in minutes and seconds, between consecutive gatherings of statistics data (STATIME). This value is also used for accounting data if ACCTIME is set to -1.
  - **V 9.3.0** Time, in minutes and seconds, between each gathering of accounting data (ACCTIME).
  - Whether tracing is started automatically (TRACSTR).
  - Size of trace table, in 4 KB blocks, to be used by the global trace facility (TRACTBL).
  - Time between scanning the queue index for WLM-managed queues (WLMTIME).
  - WLMTIMU indicates whether WLMTIME is given in seconds or minutes.
  - A list of messages excluded from being written to any log (EXCLMSG).
  - It might also return a report about system status.
2. This command is issued internally by IBM MQ at the end of queue manager startup.

## Parameter descriptions for DISPLAY SYSTEM

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

''

The command runs on the queue manager on which it was entered. This is the default value.

#### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect is the same as entering the command on every queue manager in the queue sharing group.

### ACELIM

The maximum size of the ACE storage pool in kilobytes .

### CLCACHE

The type of the cluster cache .

### SPLCAP

Whether the AMS component is installed .

## DISPLAY TCLUSTER (display cluster topic attributes)

Use the MQSC command DISPLAY TCLUSTER to display the attributes of the IBM MQ cluster topic object.

## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

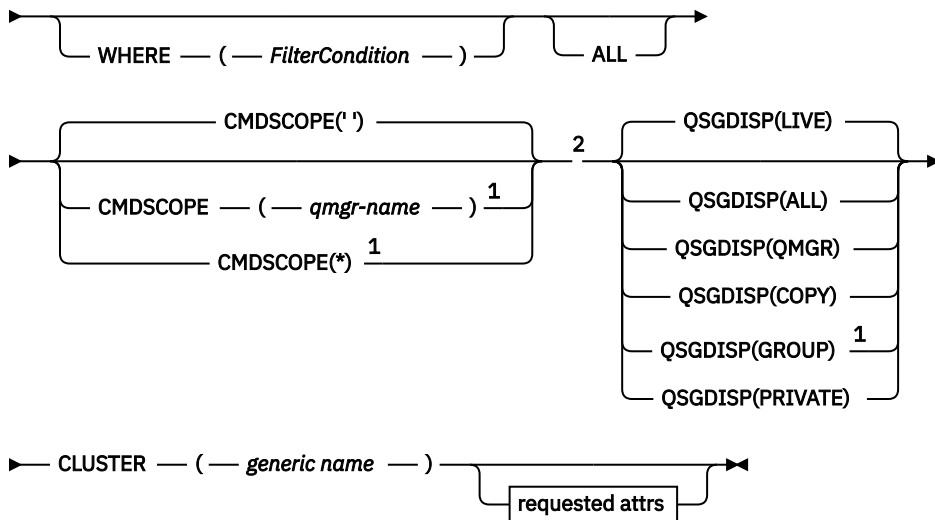
**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

**Note:** The **DISPLAY TCLUSTER** command produces the same output as the **DISPLAY TOPIC TYPE (CLUSTER)** command. See [“DISPLAY TOPIC \(display topic information\)” on page 854](#) for further information on all the attributes displayed.

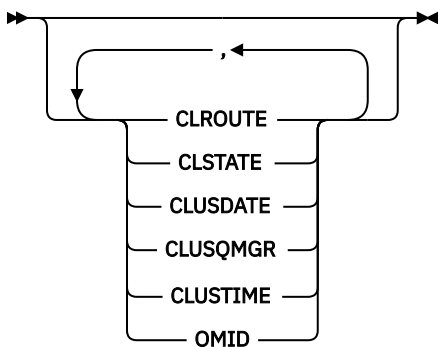
**Synonym:** DIS TCLUSTER

## DISPLAY TCLUSTER

►► DISPLAY TCLUSTER — ( — *generic-topic-name* — ) ►►



### Requested attrs



### Notes:

- <sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>2</sup> Valid only on z/OS.

## Parameter descriptions for DISPLAY TCLUSTER

You must specify the name of the cluster topic definition you want to display. This name can be a specific cluster topic name or a generic cluster topic name. By using a generic topic name, you can display either:

### (*generic-topic-name*)

The name of the administrative cluster topic definition to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk (\*) matches all administrative topic objects with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all administrative topic objects.

### WHERE

Specify a filter condition to display only those administrative topic object definitions that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

#### filter-keyword

Almost any parameter that can be used to display attributes for this DISPLAY command.



► **z/OS** However, you cannot use the CMDSCOPE, or QSGDISP parameters as filter keywords.

### operator

This part is used to determine whether a topic object satisfies the filter value on the given filter keyword. The operators are:

#### LT

Less than

#### GT

Greater than

#### EQ

Equal to

#### NE

Not equal to

#### LE

Less than or equal to

#### GE

Greater than or equal to

#### LK

Matches a generic string that you provide as a *filter-value*

#### NL

Does not match a generic string that you provide as a *filter-value*

### filter-value

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this value can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter, you can use only EQ or NE.

- A generic value. This value is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**Note:** ► **z/OS** On z/OS there is a 256 character limit for the filter-value of the MQSC **WHERE** clause. This limit is not in place for other platforms.

### ALL

Specify this parameter to display all the attributes. If this parameter is specified, any attributes that are requested specifically have no effect; all attributes are still displayed.

This is the default if you do not specify a generic name, and do not request any specific attributes.

### ► **z/OS** **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered. This value is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

### **\***

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this process is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

## **z/OS QSGDISP**

Specifies the disposition of the objects for which information is to be displayed. Values are:

### **LIVE**

LIVE is the default value and displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

### **ALL**

Display information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP).

If QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

In a shared queue manager environment, use

```
DISPLAY TOPIC(name) CMDSCOPE(*) QSGDISP(ALL)
```

to list ALL objects matching name in the queue sharing group without duplicating those objects in the shared repository.

### **COPY**

Display information only for objects defined with QSGDISP(COPY).

### **GROUP**

Display information only for objects defined with QSGDISP(GROUP). This is allowed only if there is a shared queue manager environment.

### **PRIVATE**

Display information only for objects defined with QSGDISP(QMGR) or QSGDISP(COPY). QSGDISP(PRIVATE) displays the same information as QSGDISP(LIVE).

### **QMGR**

Display information only for objects defined with QSGDISP(QMGR).

### **QSGDISP**

QSGDISP displays one of the following values:

#### **QMGR**

The object was defined with QSGDISP(QMGR).

#### **GROUP**

The object was defined with QSGDISP(GROUP).

#### **COPY**

The object was defined with QSGDISP(COPY).

You cannot use QSGDISP as a filter keyword.

## CLUSTER

Displays topics with the specified cluster name. The value can be a generic name.

## Requested attributes

### CLROUTE

The routing behavior to use for topics in the cluster defined by the **CLUSTER** parameter.

### CLSTATE

The current state of this topic in the cluster defined by the **CLUSTER** parameter. The values can be as follows:

#### **ACTIVE**

The cluster topic is correctly configured and being adhered to by this queue manager.

#### **PENDING**

Only seen by a hosting queue manager, this state is reported when the topic has been created but the full repository has not yet propagated it to the cluster. This might be because the host queue manager is not connected to a full repository, or because the full repository has deemed the topic to be invalid.

#### **INVALID**

This clustered topic definition conflicts with an earlier definition in the cluster and is therefore not currently active.

#### **ERROR**

An error has occurred with respect to this topic object.

This parameter is typically used to aid diagnosis when multiple definitions of the same clustered topic are defined on different queue managers, and the definitions are not identical. See [Routing for publish/subscribe clusters: Notes on behavior](#).

### CLUSDATE

The date on which the information became available to the local queue manager, in the form yyyy-mm-dd.

### CLUSQMGR

The name of the queue manager that hosts the topic.

### CLUSTIME


The time at which the information became available to the local queue manager, in the form hh.mm.ss.

### QMID

The internally generated unique name of the queue manager that hosts the topic.

## Usage notes for DISPLAY TCLUSTER

1. On z/OS, the channel initiator must be running before you can display information about cluster topics.
2. The TOPICSTR parameter might contain characters that cannot be translated into printable characters when the command output is displayed.

 On z/OS, these non-printable characters are displayed as blanks.

 On [Multiplatforms](#) using the **runmqsc** command, these non-printable characters are displayed as dots.

## Related reference

“DISPLAY TPSTATUS (display topic status)” on page 862

Use the MQSC command **DISPLAY TPSTATUS** to display the status of one or more topics in a topic tree.

“DISPLAY TOPIC (display topic information)” on page 854

Use the MQSC command **DISPLAY TOPIC** to display the attributes of one or more IBM MQ topic objects of any type.

## z/OS **DISPLAY THREAD (display thread information) on z/OS**

Use the MQSC command DISPLAY THREAD to display information about active and in-doubt threads.

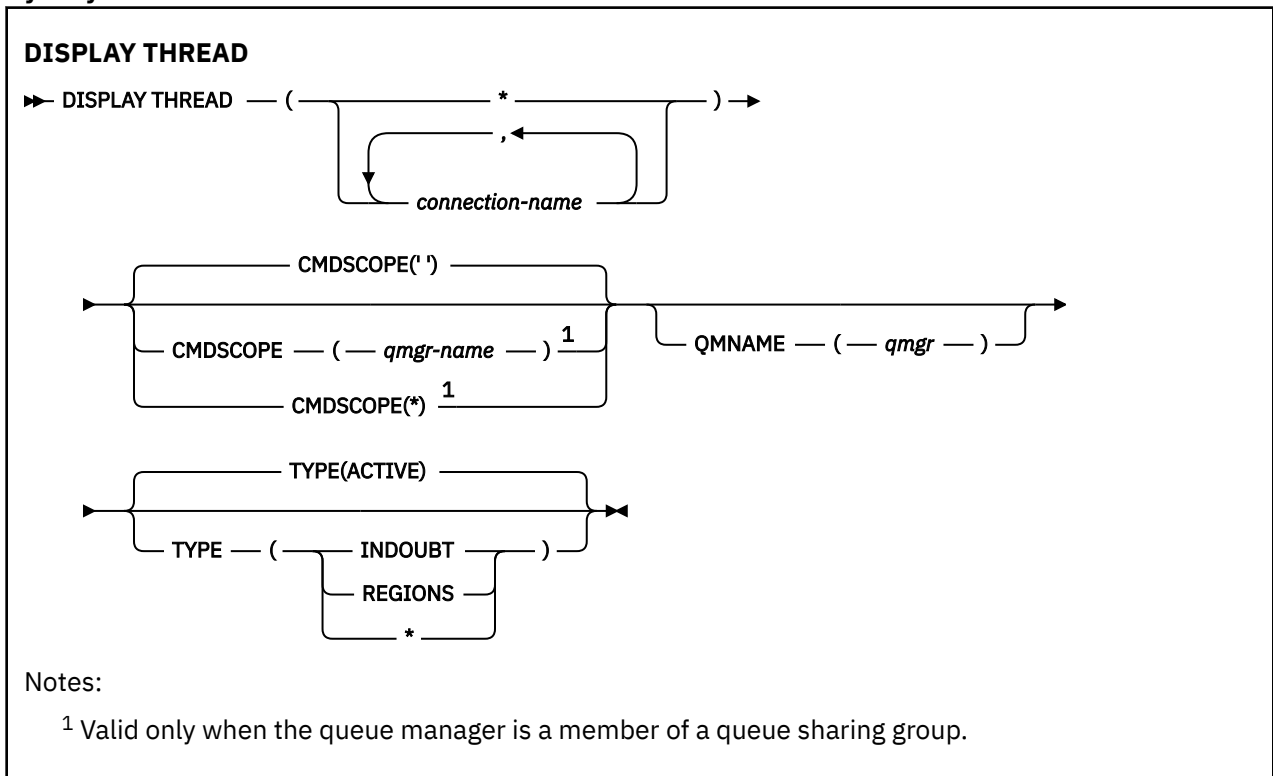
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 852](#)
- [“Parameter descriptions for DISPLAY THREAD” on page 852](#)

**Synonym:** DIS THD



### Usage notes

Threads shown as in doubt on one invocation of this command will probably be resolved for subsequent invocations.

This command is retained for compatibility with earlier release of IBM MQ. It has been superseded by the DISPLAY CONN command which is preferable to use.

### Parameter descriptions for DISPLAY THREAD

#### (*connection-name*)

List of one or more *connection-name* s (of 1 through 8 characters each).

- For batch connections, this name is the batch job name

- For CICS connections, this name is the CICS applid
  - For IMS connections, this name is the IMS job name
  - For TSO connections, this name is the TSO user ID
  - For RRS connections, this is RRSBATCH for all RRSBATCH-type connections, or the batch job name
- Threads are selected from the address spaces associated with these connections only.

(\*)

Displays threads associated with all connections to IBM MQ.

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. This is the default value.

#### ***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### **TYPE**

The type of thread to display. This parameter is optional.

#### **ACTIVE**

Display only active threads.

An active thread is one for which a unit of recovery has started but not completed. Resources are held in IBM MQ on its behalf.

This is the default if TYPE is omitted.

#### **INDOUBT**

Display only in-doubt threads.

An in-doubt thread is one that is in the second phase of the two-phase commit operation.

Resources are held in IBM MQ on its behalf. External intervention is needed to resolve the status of in-doubt threads. You might only have to start the recovery coordinator ( CICS, IMS, or RRS), or you might need to do more. They might have been in doubt at the last restart, or they might have become in doubt since the last restart.

#### **REGIONS**

Display a summary of active threads for each active connection.

**Note:** Threads used internally by IBM MQ are excluded.

\*

Display both active and in-doubt threads, but not regions.

If, during command processing, an active thread becomes in doubt, it might appear twice: once as active and once as in doubt.

### **QMNAME**

Specifies that IBM MQ should check whether the designated queue manager is INACTIVE, and if so, report any shared units of work that were in progress on the designated and inactive queue manager.

This option is valid only for TYPE(INDOUBT).

**z/OS** For more information about the DISPLAY THREAD command and in-doubt recovery, see [Recovering units of recovery on another queue manager in the queue sharing group](#). Also, see messages CSQV401I through CSQV406I, and CSQV432I, in [Agent services messages \(CSQV...\)](#).

## DISPLAY TOPIC (display topic information)

Use the MQSC command **DISPLAY TOPIC** to display the attributes of one or more IBM MQ topic objects of any type.

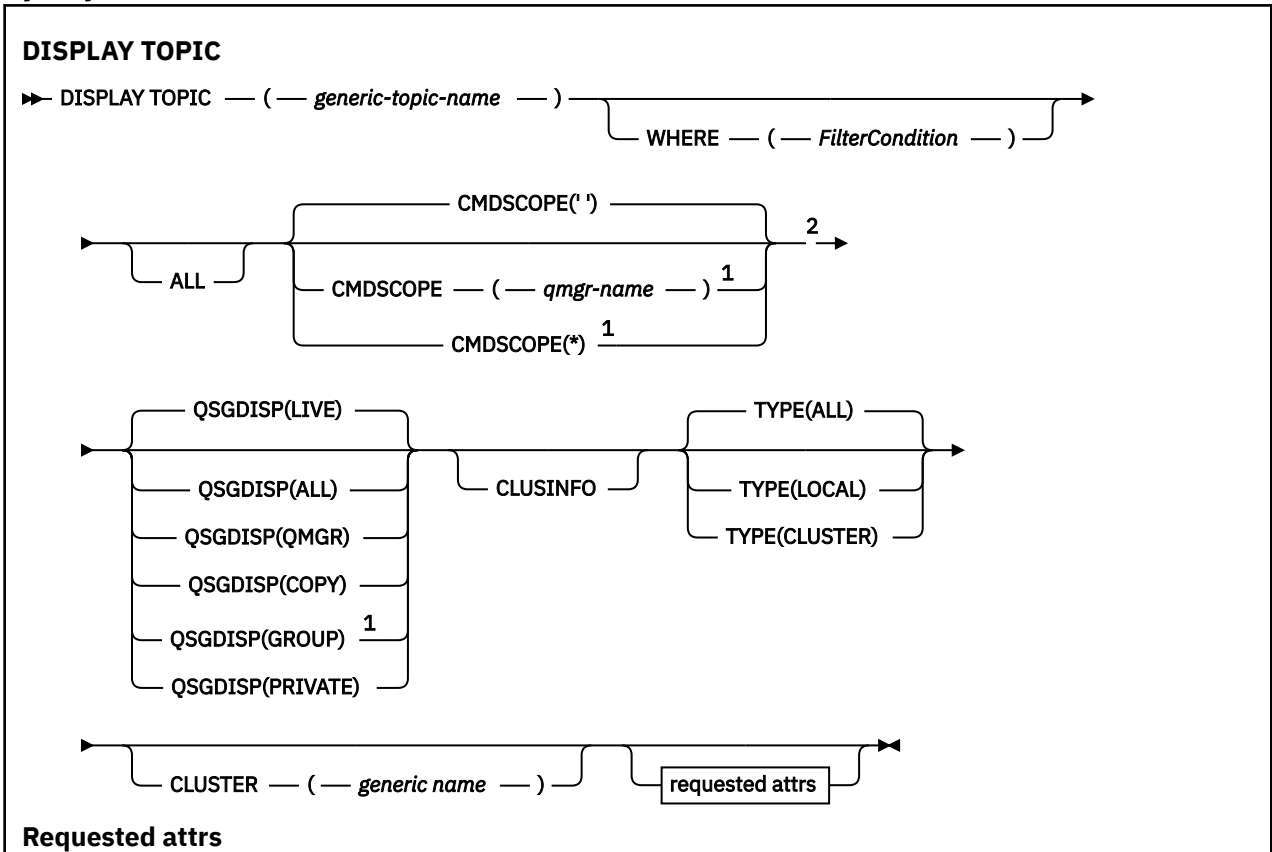
### Using MQSC commands

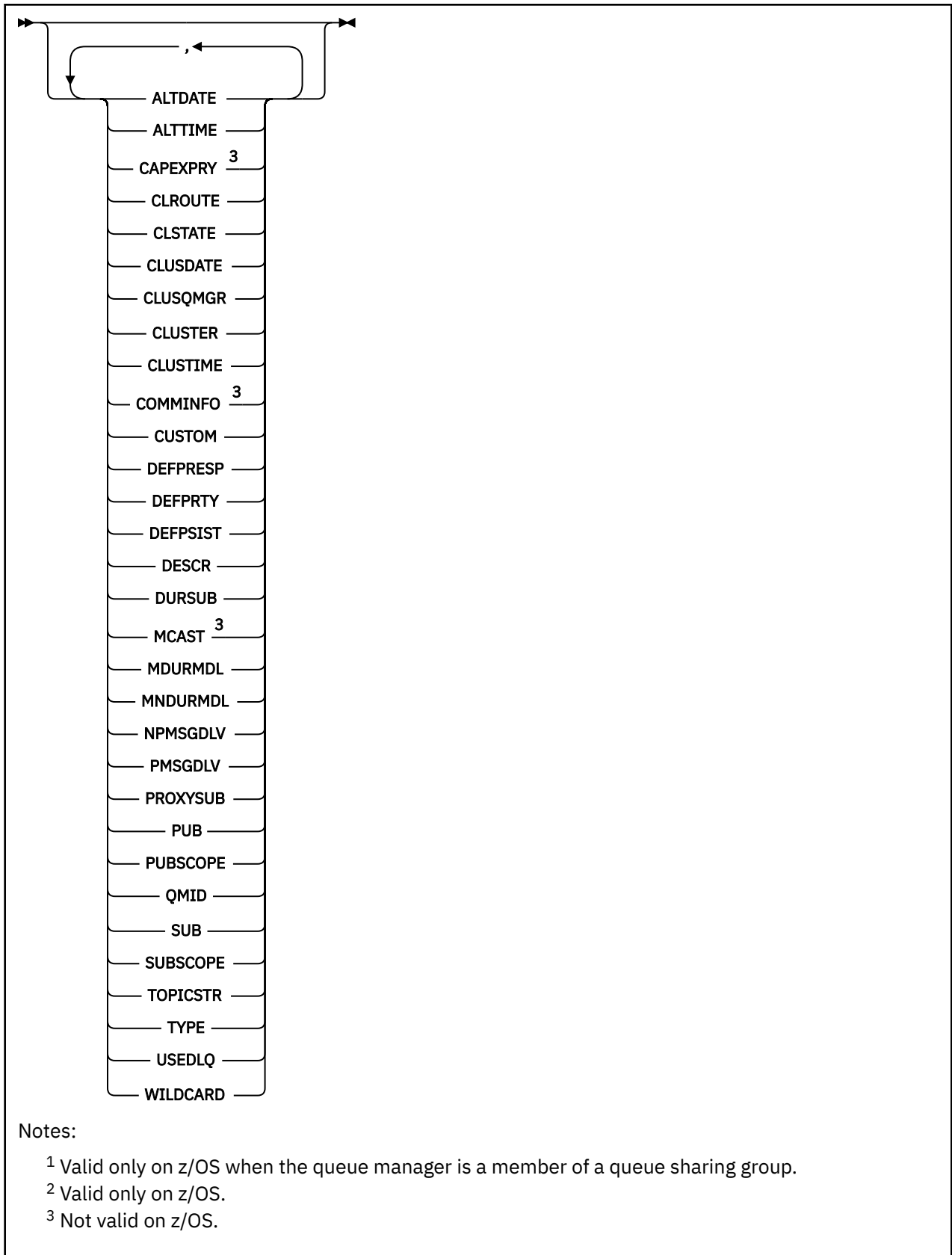
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).


- [Syntax diagram](#)
- [“Usage notes for DISPLAY TOPIC” on page 855](#)
- [“Parameter descriptions for DISPLAY TOPIC” on page 856](#)
- [“Requested parameters” on page 859](#)

**Synonym:** DIS TOPIC





### Usage notes for DISPLAY TOPIC

1.  On z/OS, the channel initiator must be running before you can display information about cluster topics, using **TYPE (CLUSTER)** or the **CLUSINFO** parameter.

2. The **TOPICSTR** parameter might contain characters that cannot be translated into printable characters when the command output is displayed.

**z/OS** On z/OS, these non-printable characters are displayed as blanks.

**Multi** On [Multiplatforms](#) using the `runmqsc` command, these non-printable characters are displayed as dots

3. You can use the following command (or synonym) as an alternative way to display these attributes.

```
DISPLAY TCLUSTER
```

This command produces the same output as the following command:

```
DISPLAY TOPIC TYPE(CLUSTER)
```

If you enter the command in this way, do not use the **TYPE** parameter.

## Parameter descriptions for **DISPLAY TOPIC**

You must specify the name of the topic definition you want to display. This name can be a specific topic name or a generic topic name. By using a generic topic name, you can display either:

- All topic definitions
- One or more topic definitions that match the specified name

### **(generic-topic-name)**

The name of the administrative topic definition to be displayed (see [Rules for naming IBM MQ objects](#)). A trailing asterisk (\*) matches all administrative topic objects with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all administrative topic objects.

## **WHERE**

Specify a filter condition to display only those administrative topic object definitions that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

### **filter-keyword**

Almost any parameter that can be used to display attributes for this **DISPLAY** command. However, you cannot use the **CMDSCOPE**, or **QSGDISP** parameters as filter keywords.

### **operator**

This part is used to determine whether a topic object satisfies the filter value on the given filter keyword. The operators are:

#### **LT**

Less than

#### **GT**

Greater than

#### **EQ**

Equal to

#### **NE**

Not equal to

#### **LE**

Less than or equal to

#### **GE**

Greater than or equal to

#### **LK**

Matches a generic string that you provide as a *filter-value*

#### **NL**

Does not match a generic string that you provide as a *filter-value*



### filter-value


The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this value can be:

- An explicit value, that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter, you can use only EQ or NE.

- A generic value. This value is a character string (such as the character string you supply for the DESCR parameter) with an asterisk at the end, for example ABC\*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed. Only a single trailing wildcard character (asterisk) is permitted.

You cannot use a generic filter-value for parameters with numeric values or with one of a set of values.

**Note:**  On z/OS there is a 256 character limit for the filter-value of the MQSC **WHERE** clause. This limit is not in place for other platforms.

### ALL

Specify this parameter to display all the attributes. If this parameter is specified, any attributes that are requested specifically have no effect; all attributes are still displayed.

This is the default if you do not specify a generic name, and do not request any specific attributes.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered. This value is the default value.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this process is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE as a filter keyword.

### **QSGDISP**

Specifies the disposition of the objects for which information is to be displayed. Values are:

#### LIVE

LIVE is the default value and displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

#### ALL

Display information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

If there is a shared queue manager environment, and the command is being processed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP).

If QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

In a shared queue manager environment, use

```
DISPLAY TOPIC(name) CMDSCOPE(*) QSGDISP(ALL)
```

to list ALL objects matching name in the queue sharing group without duplicating those objects in the shared repository.

**COPY**

Display information only for objects defined with QSGDISP(COPY).

**GROUP**

Display information only for objects defined with QSGDISP(GROUP). This is allowed only if there is a shared queue manager environment.

**PRIVATE**

Display information only for objects defined with QSGDISP(QMGR) or QSGDISP(COPY). QSGDISP(PRIVATE) displays the same information as QSGDISP(LIVE).

**QMGR**

Display information only for objects defined with QSGDISP(QMGR).

**QSGDISP**

QSGDISP displays one of the following values:

**QMGR**

The object was defined with QSGDISP(QMGR).

**GROUP**

The object was defined with QSGDISP(GROUP).


**COPY**

The object was defined with QSGDISP(COPY).

You cannot use QSGDISP as a filter keyword.

**CLUSINFO**


Requests that, in addition to information about attributes of topics defined on this queue manager, information about these and other topics in the cluster, that match the selection criteria, is displayed. In this case, there might be multiple topics with the same topic string displayed. The cluster information is obtained from the repository on this queue manager.

 On z/OS, the channel initiator must be running before you can use the CLUSINFO parameter to display information about cluster topics.

**CLUSTER**

Limits the information displayed to topics with the specified cluster name if entered with a value in brackets. The value can be a generic name.

If you do not enter a value to qualify this parameter, it is treated as a requested parameter, and cluster name information is returned about all the topics displayed.

 On z/OS, the channel initiator must be running before you can use the CLUSINFO parameter to display information about cluster topics.

**TYPE**

Specifies the type of topics that you want to be displayed. Values are:

**ALL**

Display all topic types, including cluster topics if you also specify CLUSINFO.

**LOCAL**

Display locally defined topics.

**CLUSTER**

Display topics that are defined in publish/subscribe clusters. Cluster attributes include:

**CLUSDATE**

The date on which the definition became available to the local queue manager, in the form yyyy-mm-dd.

**CLUSQMGR**

The name of the queue manager hosting the topic.

**CLUSTIME**

The time at which the definition became available to the local queue manager, in the form hh.mm.ss.

**QMID**

The internally generated, unique name of the queue manager hosting the topic.

**Requested parameters**

Specify one or more parameters that define the data to be displayed. The parameters can be specified in any order, but do not specify the same parameter more than once.

Most of the parameters are relevant for both types of topics, but parameters that are not relevant for a particular type of topic cause no output, nor is an error raised.

The following table shows the parameters that are relevant for each type of topic. There is a brief description of each parameter after the table, but for more information, see [“DEFINE TOPIC \(define a new administrative topic\)”](#) on page 595.

<i>Table 174. Parameters that can be returned by the DISPLAY TOPIC command</i>		
	<b>Local topic</b>	<b>Cluster topic</b>
<u>ALTDATE</u>	✓	✓
<u>ALTTIME</u>	✓	✓
<u>CLROUTE</u>	✓	✓
<u>CLSTATE</u>		✓
<u>CLUSDATE</u>		✓
<u>CLUSQMGR</u>		✓
<u>CLUSTER</u>	✓	✓
<u>CLUSTIME</u>		✓
<u>COMMINFO</u>	✓	
<u>CUSTOM</u>	✓	✓
<u>DEFPRTY</u>	✓	✓
<u>DEFPSIST</u>	✓	✓
<u>DEFPRESP</u>	✓	✓
<u>DESCR</u>	✓	✓
<u>DURSUB</u>	✓	✓
<u>MCAST</u>	✓	

Table 174. Parameters that can be returned by the DISPLAY TOPIC command (continued)

	Local topic	Cluster topic
<u>MDURMDL</u>	✓	✓
<u>MNDURMDL</u>	✓	✓
<u>NPMSGDLV</u>	✓	✓
<u>PMSGDLV</u>	✓	✓
<u>PROXYSUB</u>	✓	✓
<u>PUB</u>	✓	✓
<u>PUBSCOPE</u>	✓	✓
<u>QMID</u>		✓
<u>SUB</u>	✓	✓
<u>SUBSCOPE</u>	✓	✓
<u>TOPICSTR</u>	✓	✓
<u>TYPE</u>	✓	✓
<u>USEDLQ</u>	✓	
<u>WILDCARD</u>	✓	✓

**ALTDATE**

The date on which the definition or information was last altered, in the form yyyy-mm-dd.

**ALTTIME**

The time at which the definition or information was last altered, in the form hh.mm.ss.

**V 9.3.1 V 9.3.1 Multi CAPEXPY**

The maximum time, expressed in tenths of a second, until a message put on an object handle, opened using this object on the resolution path, remains in the system until it becomes eligible for expiry processing.

**CLROUTE**

The routing behavior to use for topics in the cluster defined by the **CLUSTER** parameter.

**CLSTATE**

The current state of this topic in the cluster defined by the **CLUSTER** parameter. The values can be as follows:

**ACTIVE**

The cluster topic is correctly configured and being adhered to by this queue manager.

**PENDING**

Only seen by a hosting queue manager, this state is reported when the topic has been created but the full repository has not yet propagated it to the cluster. This might be because the host queue manager is not connected to a full repository, or because the full repository has deemed the topic to be invalid.

**INVALID**

This clustered topic definition conflicts with an earlier definition in the cluster and is therefore not currently active.

**ERROR**

An error has occurred with respect to this topic object.

This parameter is typically used to aid diagnosis when multiple definitions of the same clustered topic are defined on different queue managers, and the definitions are not identical. See [Routing for publish/subscribe clusters: Notes on behavior](#).

**CLUSDATE**

The date on which the information became available to the local queue manager, in the form yyyy-mm-dd.

**CLUSQMGR**

The name of the queue manager that hosts the topic.

**CLUSTER**

The name of the cluster that the topic is in.

**CLUSTIME**

The time at which the information became available to the local queue manager, in the form hh.mm.ss.

**COMMINFO**

The communication information object name.

**CUSTOM**

This attribute is reserved for the configuration of new features before separate attributes have been introduced. It can contain the values of zero or more attributes as pairs of attribute name and value in the form NAME (VALUE).

**DEFPRTY**

Default priority of the messages published to this topic.

**DEFPSIST**

Default persistence of messages published to this topic.

**DEFPRESP**

Default put response for this topic. This attribute defines the behavior that must be used by applications when the put response type in the MQPMO options has been set to MQPMO\_RESPONSE\_AS\_TOPIC\_DEF.

**DESCR**

Description of this administrative topic object.

**DURSUB**

Determines whether the topic permits durable subscriptions to be made.

**MCAST**

Specifies whether the topic is enabled for multicast.

**MDURMDL**

The name of the model queue for durable managed subscriptions.

**MNDURMDL**

The name of the model queue for non-durable managed subscriptions.

**NPMSGDLV**

The delivery mechanism for non-persistent messages.

**PMSGDLV**

The delivery mechanism for persistent messages.

**PROXYSUB**

Determines whether a proxy subscription is forced for this subscription, even if no local subscriptions exist.

**PUB**

Determines whether the topic is enabled for publication.

**PUBSCOPE**

Determines whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster.

**QMID**

The internally generated unique name of the queue manager that hosts the topic.

**SUB**

Determines whether the topic is enabled for subscription.

**SUBSCOPE**

Determines whether this queue manager propagates subscriptions to queue managers as part of a hierarchy or as part of a publish/subscribe cluster.

**TOPICSTR**

The topic string.

**TYPE**

Specifies whether this object is a local topic or cluster topic.

**USEDLQ**

Determines whether the dead-letter queue is used when publication messages cannot be delivered to their correct subscriber queue.

**WILDCARD**

The behavior of wildcard subscriptions with respect to this topic.

For more details of these parameters, except the **CLSTATE** parameter, see [“DEFINE TOPIC \(define a new administrative topic\)”](#) on page 595.

**Related tasks**

[Displaying administrative topic object attributes](#)

[Changing administrative topic attributes](#)

**Related reference**

[“DISPLAY TPSTATUS \(display topic status\)”](#) on page 862


Use the MQSC command **DISPLAY TPSTATUS** to display the status of one or more topics in a topic tree.

**DISPLAY TPSTATUS (display topic status)**

Use the MQSC command **DISPLAY TPSTATUS** to display the status of one or more topics in a topic tree.

**Using MQSC commands**

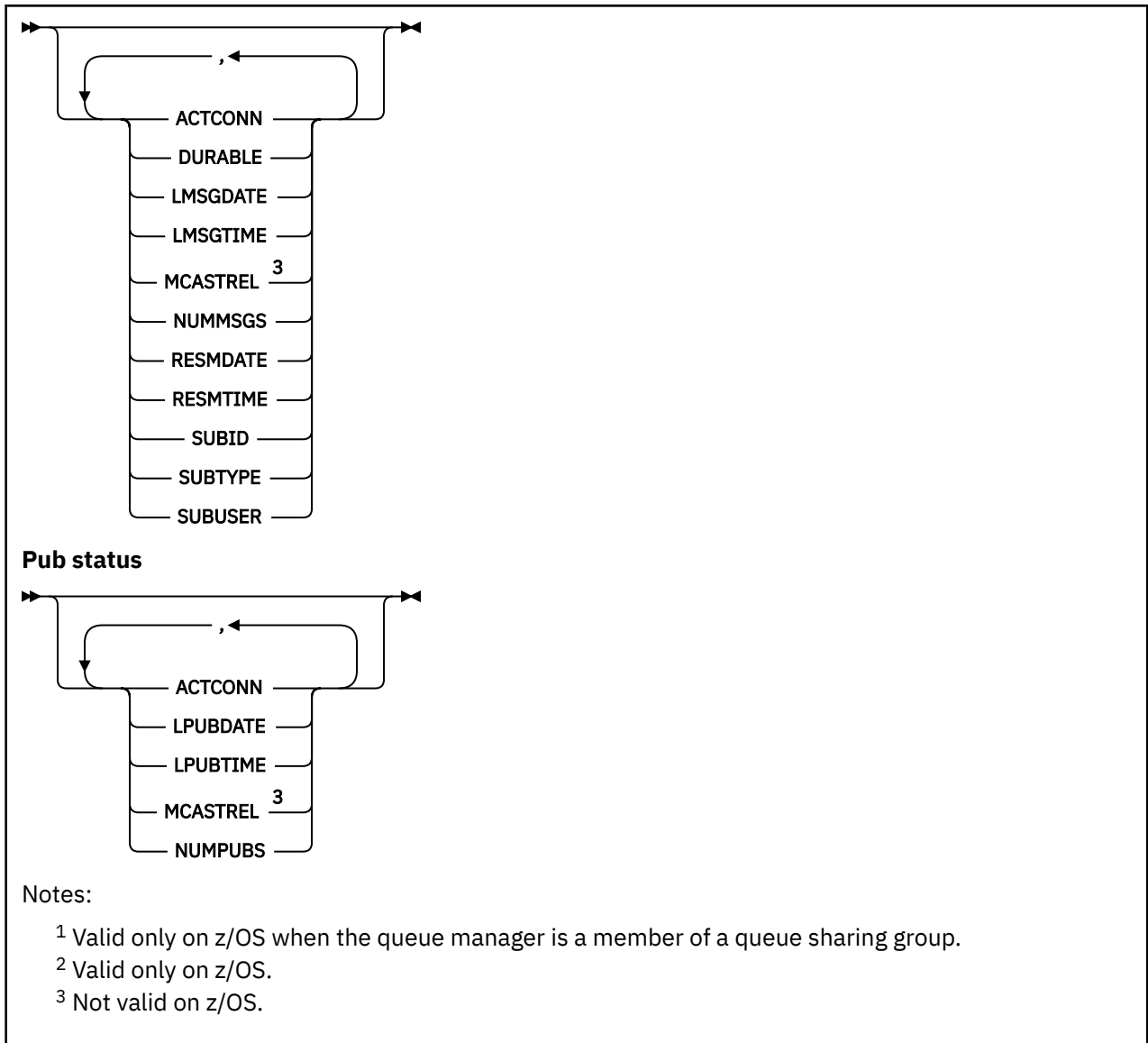
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for DISPLAY TPSTATUS”](#) on page 864
- [“Parameter descriptions for DISPLAY TPSTATUS”](#) on page 864
- [“Topic status parameters”](#) on page 866
- [“Sub status parameters”](#) on page 868
- [“Pub status parameters”](#) on page 869

**Synonym:** DIS TPS





## Usage notes for DISPLAY TPSTATUS

1. The TOPICSTR parameter might contain characters that cannot be translated into printable characters when the command output is displayed.
  - ▶ **Multi** On Multiplatforms using the `runmqsc` command, these non-printable characters are displayed as dots.
  - ▶ **z/OS** On z/OS, these non-printable characters are displayed as blanks.
2. The topic-string input parameter on this command must match the topic you want to act on. Keep the character strings in your topic strings as characters that can be used from the location issuing the command. If you issue commands using MQSC, you have fewer characters available to you than if you are using an application that submits PCF messages, such as the IBM MQ Explorer.

## Parameter descriptions for DISPLAY TPSTATUS

The **DISPLAY TPSTATUS** command requires a topic string value to determine which topic nodes the command returns.



**topicstr)**

The value of the topic string for which you want to display status information. You cannot specify the name of an IBM MQ topic object.

The topic string can have one of the following values:

- A specific topic string value. For example, `DIS TPS('Sports/Football')` returns just the 'Sports/Football' node.
- A topic string containing a "+" wildcard character. For example, `DIS TPS('Sports/Football/+')` returns all direct child nodes of the 'Sports/Football' node.
- A topic string containing a "#" wildcard character. For example, `DIS TPS('Sports/Football/#')` returns the 'Sports/Football' node and all its descendant nodes.
- A topic string containing more than one wildcard. For example, `DIS TPS('Sports+/Teams/#')` returns any direct child node of 'Sports' that also has a 'teams' child, with all descendants of the latter nodes.

The **DISPLAY TPSTATUS** command does not support the '\*' wildcard. For more information about using wildcards, see the related topic.

- To return a list of all root-level topics, use `DIS TPS(' + ')`
- To return a list of all topics in the topic tree, use `DIS TPS(' # ')`, but note that this command might return a large amount of data.
- To filter the list of topics returned, use the **WHERE** parameter. For example, `DIS TPS('Sports/Football/+') WHERE(TOPICSTR LK 'Sports/Football/L*')` returns all direct child nodes of the 'Sports/Football' node, that begin with the letter "L".

**WHERE**

Specifies a filter condition to display only those administrative topic definitions that satisfy the selection criterion of the filter condition. The filter condition is in three parts: *filter-keyword*, *operator*, and *filter-value*:

**filter-keyword**

Except for the **CMDSCOPE** parameter, any parameter that you can use with this **DISPLAY** command.

**operator**

Determines whether a topic string satisfies the filter value on the given filter keyword. The operators are:

**LT**

Less than

**GT**

Greater than

**EQ**

Equal to

**NE**

Not equal to

**LE**

Less than or equal to

**GE**

Greater than or equal to

**LK**

Matches a generic string that you provide as a *topicstr*

**NL**

Does not match a generic string that you provide as a *topicstr*

**filter-value**

The value that the attribute value must be tested against using the operator. Depending on the filter-keyword, this value can be:

- An explicit value that is a valid value for the attribute being tested.

You can use operators LT, GT, EQ, NE, LE, or GE only. However, if the attribute value is one from a possible set of values on a parameter, you can use only EQ or NE.

- A generic value. This value is a character string with an asterisk at the end, for example ABC\*. If the operator is LK, the command lists all topic nodes that begin with the string (ABC in the example). If the operator is NL, the command lists all topic nodes that do not begin with the string.

You cannot use a generic *filter-value* for parameters with numeric values or with one of a set of values.

**ALL**

Use this parameter to display all attributes.

If this parameter is specified, any attributes that you request specifically have no effect; the command displays all attributes.

This parameter is the default parameter if you do not specify a generic name, and do not request any specific attributes.

**z/OS CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. This value is the default value.

**qmgr-name**

The command runs on the named queue manager, if the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which you enter the command, but only if you are using a queue sharing group environment and the command server is enabled.

\*

The command runs on the local queue manager and on every active queue manager in the queue sharing group. The effect of this option is equivalent to entering the command on every queue manager in the queue sharing group.

**TYPE****TOPIC**

The command displays status information relating to each topic node, which is the default if you do not provide a **TYPE** parameter.

**PUB**

The command displays status information relating to applications that have topic nodes open for publish.

**SUB**

The command displays status information relating to applications that subscribe to the topic node or nodes. The subscribers that the command returns are not necessarily the subscribers that would receive a message published to this topic node. The value of **SelectionString** or **SubLevel** determines which subscribers receive such messages.

**Topic status parameters**

Topic status parameters define the data that the command displays. You can specify these parameters in any order but must not specify the same parameter more than once.

Topic objects can be defined with attributes that have a value of *ASPARENT*. Topic status shows the resolved values that result in finding the setting of the closest parent administrative topic object in the topic tree, and so will never display a value of *ASPARENT*.

#### **ADMIN**

If the topic node is an admin-node, the command displays the associated topic object name containing the node configuration. If the field is not an admin-node the command displays a blank.

V 9.3.1

Multi

#### **CAPEXPY(*integer*)**

The maximum time, expressed in tenths of a second, until a message published to a topic which inherits properties from this object, remains in the system until it becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

#### ***integer***

The value must be in the range one through to 999 999 999.

#### **NOLIMIT**

There is no limit on the expiry time of messages put to this topic.

#### **ASPARENT**

The maximum message expiry time is based on the setting of the closest parent administrative topic object in the topic tree. This is the default value.

#### **CLROUTE**

The routing behavior to use for topics in the cluster defined by the **CLUSTER** parameter. The values can be as follows:

#### ***DIRECT***

A publication on this topic string, originating from this queue manager, is sent direct to any queue manager in the cluster with a matching subscription.

#### ***TOPICHOST***

A publication on this topic string, originating from this queue manager, is sent to one of the queue managers in the cluster that hosts a definition of the corresponding clustered topic object, and from there to any queue manager in the cluster with a matching subscription.

#### ***NONE***

This topic node is not clustered.

#### **CLUSTER**

The name of the cluster to which this topic belongs.

..

This topic does not belong to a cluster. Publications and subscriptions for this topic are not propagated to publish/subscribe cluster-connected queue managers.

#### **COMMINFO**

Displays the resolved value of the name of the communication information object to be used for this topic node.

#### **DEFPRESP**

Displays the resolved default put response of messages published to the topic. The value can be *SYNC* or *ASYNC*

#### **DEFPRTY**

Displays the resolved default priority of messages published to the topic.

#### **DEFPSIST**

Displays the resolved default persistence for this topic string. The value can be *YES* or *NO*.

#### **DURSUB**

Displays the resolved value that shows whether applications can make durable subscriptions. The value can be *YES* or *NO*.

**MCAST**

Displays the resolved value that shows whether the topic could be transmittable via multicast or not. The value can be *ENABLED*, *DISABLED*, or *ONLY*.

**MDURMDL**

Displays the resolved value of the name of the model queue to be used for durable subscriptions.

**MNDURMDL**

Displays the resolved value of the name of the model queue used for non-durable subscriptions.

**NPMSGDLV**

Displays the resolved value for the delivery mechanism for non-persistent messages published to this topic. The value can be *ALL*, *ALLDUR*, or *ALLAVAIL*.

**PMSGDLV**

Displays the resolved value for the delivery mechanism for persistent messages published to this topic. The value can be *ALL*, *ALLDUR*, or *ALLAVAIL*.

**PUB**

Displays the resolved value that shows whether publications are allowed for this topic. The values can be *ENABLED* or *DISABLED*.

**PUBCOUNT**

Displays the number of handles that are open for publish on this topic node.

**PUBSCOPE**

Determines whether this queue manager propagates publications, for this topic node, to other queue managers as part of a hierarchy or a cluster, or whether it restricts them to only subscriptions defined on the local queue manager. The value can be *QMGR* or *ALL*.

**RETAINED**

Displays whether there is a retained publication associated with this topic. The value can be *YES* or *NO*.

**SUB**

Displays the resolved value that shows whether subscriptions are allowed for this topic. The values can be *ENABLED* or *DISABLED*.

**SUBCOUNT**

Displays the number of subscribers to this topic node, including durable subscribers that are not currently connected.

**SUBSCOPE**

Determines whether this queue manager propagates subscriptions, for this topic node, to other queue managers as part of a cluster or hierarchy, or whether it restricts the subscriptions to only the local queue manager. The value can be *QMGR* or *ALL*.

**USEDLQ**

Determines whether the dead-letter queue is used when publication messages cannot be delivered to their correct subscriber queue. The value can be *YES* or *NO*.

**Sub status parameters**

Sub status parameters define the data that the command displays. You can specify these parameters in any order but must not specify the same parameter more than once.

**ACTCONN**

Detects local publications, returning the currently active ConnectionId (CONNID) that opened this subscription.

**DURABLE**

Indicates whether a durable subscription is not deleted when the creating application closes its subscription handle, and persists over queue manager restart. The value can be *YES* or *NO*.

**LMSGDATE**

The date on which an MQPUT call last sent a message to this subscription. The MQPUT call updates the date field only when the call successfully puts a message to the destination specified by this subscription. An MQSUBRQ call causes an update to this value.

**LMSGTIME**

The time at which an MQPUT call last sent a message to this subscription. The MQPUT call updates the time field only when the call successfully puts a message to the destination specified by this subscription. An MQSUBRQ call causes an update to this value.

**MCASTREL**

Indicator of the reliability of the multicast messages.

The values are expressed as a percentage. A value of 100 indicates that all messages are being delivered without problems. A value less than 100 indicates that some of the messages are experiencing network issues. To determine the nature of these issues you can enable event message generation, use the **COMMEV** parameter of the COMMINFO objects, and examine the generated event messages.

The following two values are returned:

- The first value is based on recent activity over a short period.
- The second value is based on activity over a longer period.

If no measurement is available the values are shown as blanks.

**NUMMSGS**

Number of messages put to the destination specified by this subscription. An MQSUBRQ call causes an update to this value.

**RESMDATE**

Date of the most recent MQSUB call that connected to this subscription.

**RESMTIME**

Time of the most recent MQSUB call that connected to this subscription.

**SUBID**

An all time unique identifier for this subscription, assigned by the queue manager. The format of **SUBID** matches that of a CorrelId. For durable subscriptions, the command returns the **SUBID** even if the subscriber is not currently connected to the queue manager.

**SUBTYPE**

The type of subscription, indicating how it was created. The value can be *ADMIN*, *API*, or *PROXY*.

**SUBUSER**

The user ID that owns this subscription, which can be either the user ID associated with the creator of the subscription or, if subscription takeover is permitted, the user ID that last took over the subscription.

**Pub status parameters**

Pub status parameters define the data that the command displays. You can specify these parameters in any order but must not specify the same parameter more than once.

**ACTCONN**

The currently active ConnectionId (CONNID) associated with the handle that has this topic node open for publish.

**LPUBDATE**

The date on which this publisher last sent a message.

**LPUBTIME**

The time at which this publisher last sent a message.

**MCASTREL**

Indicator of the reliability of the multicast messages.

The values are expressed as a percentage. A value of 100 indicates that all messages are being delivered without problems. A value less than 100 indicates that some of the messages are experiencing network issues. To determine the nature of these issues you can enable event message generation, using the **COMMEV** parameter of the COMMINFO objects, and examine the generated event messages.

The following two values are returned:

- The first value is based on recent activity over a short period.
- The second value is based on activity over a longer period.

If no measurement is available the values are shown as blanks.

### NUMPUBS

Number of publishes by this publisher. This value records the actual number of publishes, not the total number of messages published to all subscribers.

### Related tasks

[Displaying administrative topic object attributes](#)

### Related reference

[“DISPLAY TOPIC \(display topic information\)” on page 854](#)

Use the MQSC command **DISPLAY TOPIC** to display the attributes of one or more IBM MQ topic objects of any type.

## **DISPLAY TRACE (display active traces list) on z/OS**

Use the MQSC command DISPLAY TRACE to display a list of active traces.

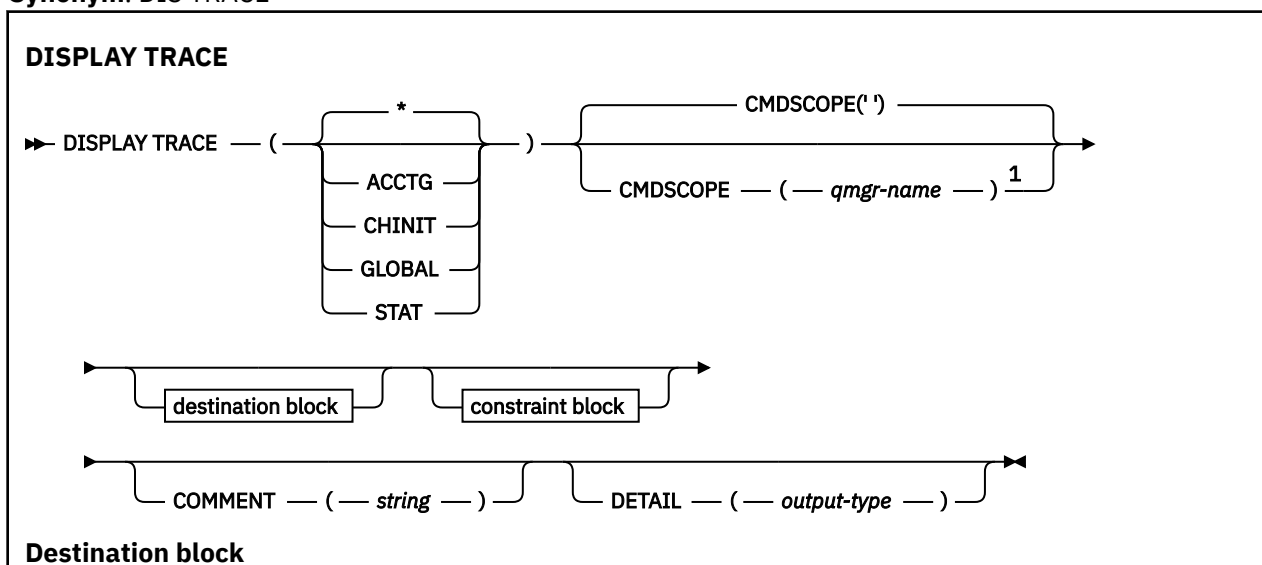
### Using MQSC commands on z/OS

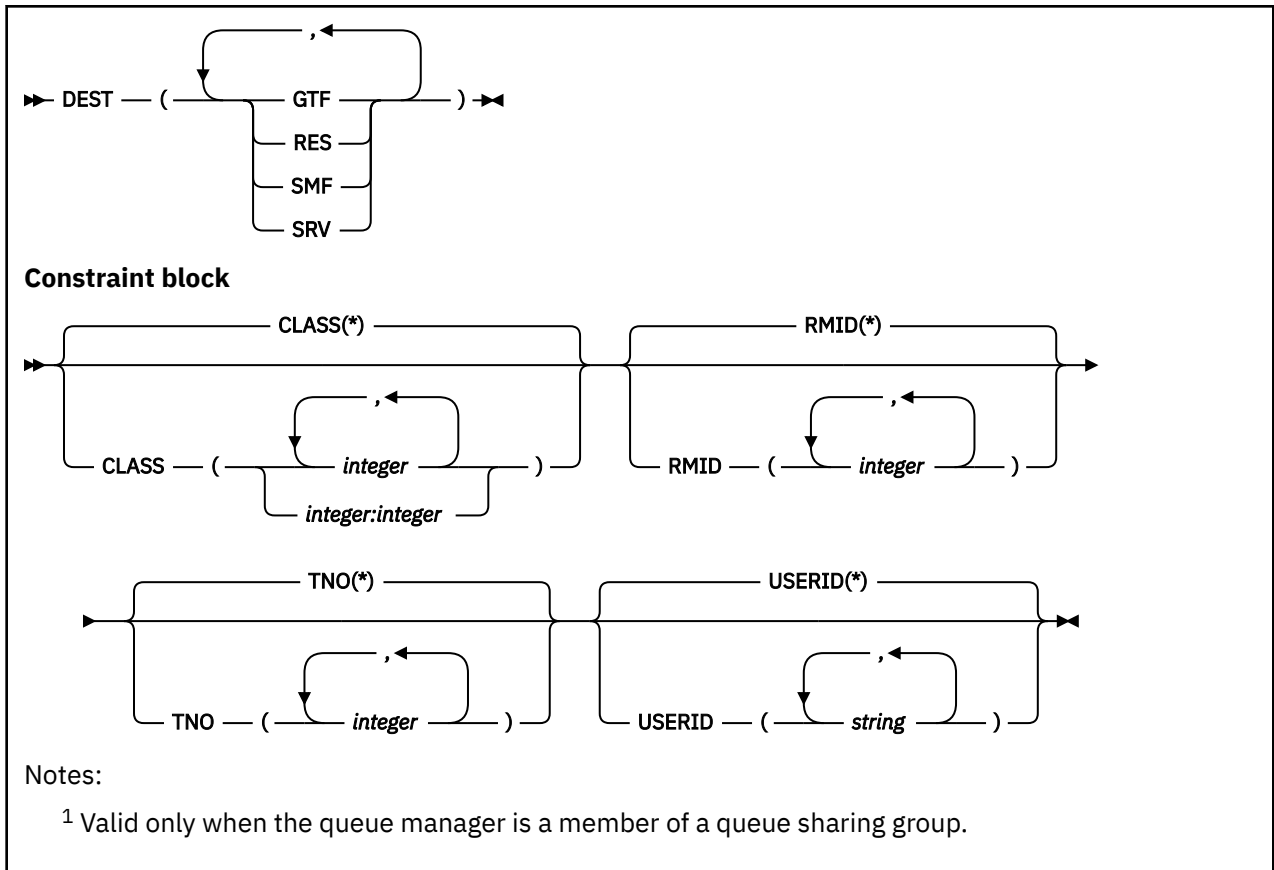
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY TRACE” on page 871](#)
- [“Destination block” on page 872](#)
- [“Constraint block” on page 872](#)

**Synonym:** DIS TRACE





## Parameter descriptions for DISPLAY TRACE

All parameters are optional. Each option that is used limits the effect of the command to active traces that were started using the same option, either explicitly or by default, with exactly the same parameter values.

\*

Does not limit the list of traces. This is the default. The CLASS option cannot be used with DISPLAY TRACE(\*).

Each remaining parameter in this section limits the list to traces of the corresponding type:

### ACCTG

Accounting data (the synonym is A)

### CHINIT

Service data from the channel initiator. The synonym is CHI or DQM.

### GLOBAL

Service data from the entire queue manager except the channel initiator. The synonym is G.

### STAT

Statistical data (the synonym is S)

### COMMENT(*string*)

Specifies a comment. This does not appear in the display, but it might be recorded in trace output.

### DETAIL(*output-type*)

This parameter is ignored; it is retained only for compatibility with earlier releases.

Possible values for *output-type* are \*, 1, or 2.

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

## Destination block

### DEST

Limits the list to traces started for particular destinations. More than one value can be specified, but do not use the same value twice. If no value is specified, the list is not limited.

Possible values and their meanings are:

**GTF**

The Generalized Trace Facility

**RES**

A wraparound table residing in the ECSA (extended common service area)

**SMF**

The System Management Facility

**SRV**

A serviceability routine designed for IBM for problem diagnosis

## Constraint block

### CLASS( *integer* )

Limits the list to traces started for particular classes. See [“START TRACE \(start trace\) on z/OS” on page 957](#) for a list of allowed classes.

The default is CLASS(\*), which does not limit the list.

### RMID( *integer* )

Limits the list to traces started for particular resource managers. See [“START TRACE \(start trace\) on z/OS” on page 957](#) for a list of allowed resource manager identifiers. Do not use this option with the STAT or CHINIT trace type.

The default is RMID(\*), which does not limit the list.

### TNO( *integer* )

Limits the list to particular traces, identified by their trace number (0 to 32). Up to 8 trace numbers can be used. If more than one number is used, only one value for USERID can be used. The default is TNO(\*), which does not limit the list.

0 is the trace that the channel initiator can start automatically. Traces 1 to 32 are those for queue manager or the channel initiator that can be started automatically by the queue manager, or manually, using the START TRACE command.

### USERID( *string* )

Limits the list to traces started for particular user IDs. Up to 8 user IDs can be used. If more than one user ID is used, only one value can be used for TNO. Do not use this option with STAT. The default is USERID(\*), which does not limit the list.



## z/OS DISPLAY USAGE (display usage information) on z/OS

Use the MQSC command DISPLAY USAGE to display information about the current state of a page set, to display information about the log data sets, or to display information about the shared message data sets.

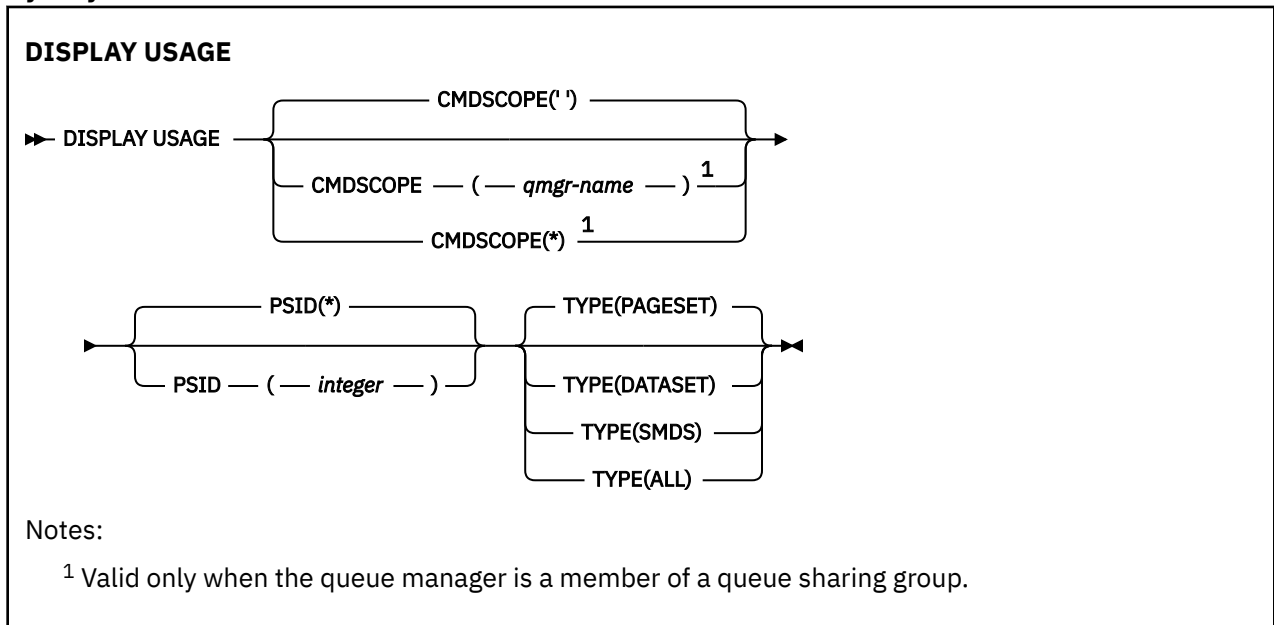
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for DISPLAY USAGE” on page 873](#)

**Synonym:** DIS USAGE



### Parameter descriptions for DISPLAY USAGE

#### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

#### PSID(*integer*)

The page-set identifier. This is optional.

\*

An asterisk (\*) on its own specifies all page set identifiers. This is the default value.

### **integer**

This is a number, in the range 00 through 99.

The command fails if PSID has been specified together with TYPE(DATASET), or TYPE(SMDS).

If the command is running at the same time as an ALTER BUFFPOOL command the buffer pool attributes might not be entirely consistent. For example, the value of the location parameter might be BELOW, but the number of available buffers value might be more than can fit below the bar. If this occurs, run the display command again when the ALTER BUFFPOOL command has completed.

### **TYPE**

Defines the type of information to be displayed. Values are:

#### **PAGESET**

Display page set and buffer pool information. This is the default.

#### **DATASET**

Display data set information for log data sets. This returns messages containing 44-character data set names for the following:

- The log data set containing the BEGIN\_UR record for the oldest incomplete unit of work for this queue manager, or if there are no incomplete units of work, the log data set containing the current highest written RBA.
- The log data set containing the oldest restart\_RBA of any page set owned by this queue manager.
- The log data set with a timestamp range that includes the timestamp of the last successful backup of any application structure known within the queue sharing group.

#### **SMDS**

Display data set space usage information and buffer pool information for shared message data sets owned by this queue manager. Space usage information is only available when the data set is open. Buffer pool information is only available when the queue manager is connected to the structure. For more information about the displayed information, see the descriptions of messages CSQE280I and CSQE285I.

#### **ALL**

Display page set, data set, and SMDS information.

**Note:** This command is issued internally by IBM MQ:

- During queue manager shutdown so that the restart RBA is recorded on the z/OS console log.
- At queue manager startup so that page set information can be recorded.
- When DEFINE PSID is used to dynamically define the first page set in the queue manager that uses the buffer pool specified on the DEFINE PSID command.

### **Related reference**

[“ALTER PSID \(change page set expansion method\) on z/OS” on page 346](#)

Use the MQSC command **ALTER PSID** to change the expansion method for a page set.

## **MOVE QLOCAL (move messages between local queues) on z/OS**

Use the MQSC command MOVE QLOCAL to move all the messages from one local queue to another.

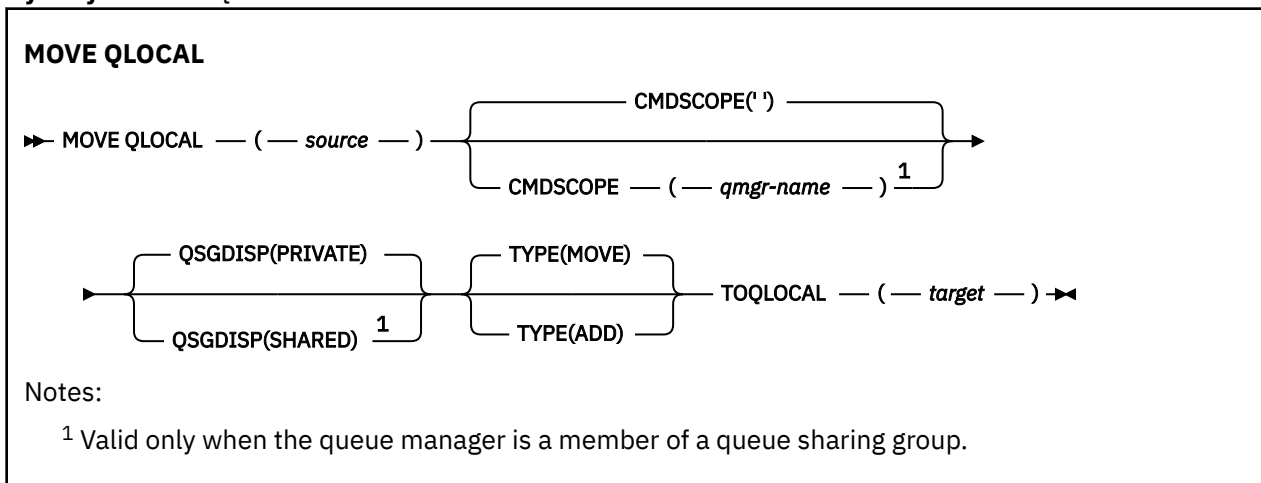
### **Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- Syntax diagram
- “Usage notes for MOVE QLOCAL” on page 875
- “Parameter descriptions for MOVE QLOCAL” on page 876

**Synonym:** MOVE QL



## Usage notes for MOVE QLOCAL

1. A typical use of the MOVE QLOCAL command is to move messages from a private queue to a shared queue when you are setting up a queue sharing group environment.
  2. The MOVE QLOCAL command **moves** messages; it does not copy them.
  3. The MOVE QLOCAL command moves messages in a similar way to an application performing successive MQGET and MQPUT calls. However, the MOVE QLOCAL command does not physically delete logically-expired messages and, therefore, no expiration reports are generated.
  4. The priority, context, and persistence of each message are not changed.
  5. The command performs no data conversion and calls no exits.
  6. Confirm-on-delivery (COD) report messages are not generated but confirm-on-arrival (COA) report messages are. This means that more than one COA report message can be generated for a message.
  7. The MOVE QLOCAL command transfers the messages in batches. At COMMIT time, if the trigger conditions are met, trigger messages are produced. This might be at the end of the move operation.
- Note:** Before the transfer of messages begins, this command verifies that the number of messages on the source queue, when added to the number of messages on the target queue, does not exceed MAXDEPTH on the target queue.
- If the MAXDEPTH of the target queue were to be exceeded, no messages are moved.
8. The MOVE QLOCAL command can change the sequence in which messages can be retrieved. The sequence remains unchanged only if:
    - You specify TYPE (MOVE) and
    - The MSGDLVSQ parameter of the source and target queues is the same.
  9. Messages are moved within one or more syncpoints. The number of messages in each syncpoint is determined by the queue manager.
  10. If anything prevents the moving of one or more messages, the command stops processing. This can mean that some messages have already been moved, while others remain on the source queue. Some of the reasons that prevent a message being moved are:
    - The target queue is full.
    - The message is too long for the target queue.
    - The message is persistent, but the target queue cannot store persistent messages.

- The page set is full.
11. Treatment of message properties depends on the source queue PROPCTL value. Message properties are handled as if an MQGET was performed with MQGMO\_PROPERTIES\_AS\_Q\_DEF.

**Note:** Message properties are always moved when MOVE QLOCAL is used to or from certain SYSTEM queues that hold messages with properties required by IBM MQ.

## Parameter descriptions for MOVE QLOCAL

You must specify the names of two local queues: the one you want to move messages from (the source queue) and the one you want to move the messages to (the target queue).

### **source**

The name of the local queue from which messages are moved. The name must be defined to the local queue manager.

The command fails if the queue contains uncommitted messages.

If an application has this queue open, or has open a queue that eventually resolves to this queue, the command fails. For example, the command fails if this queue is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open.

An application can open this queue while the command is in progress but the application waits until the command has completed.

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

### **QSGDISP**

Specifies the disposition of the source queue.

#### **PRIVATE**

The queue is defined with QSGDISP(QMGR) or QSGDISP(COPY). This is the default value.

#### **SHARED**

The queue is defined with QSGDISP(SHARED). This is valid only in a queue sharing group environment.

### **TYPE**

Specifies how the messages are moved.

#### **MOVE**

Move the messages from the source queue to the empty target queue.

The command fails if the target queue already contains one or more messages. The messages are deleted from the source queue. This is the default value.

#### **ADD**

Move the messages from the source queue and add them to any messages already on the target queue.

The messages are deleted from the source queue.

## target

The name of the local queue to which messages are moved. The name must be defined to the local queue manager.

The name of the target queue can be the same as that of the source queue only if the queue exists as both a shared and a private queue. In this case, the command moves messages to the queue that has the opposite disposition (shared or private) from that specified for the source queue on the QSGDISP parameter.

If an application has this queue open, or has open a queue that eventually resolves to this queue, the command fails. The command also fails if this queue is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open.

No application can open this queue while the command is in progress.

If you specify TYPE (MOVE), the command fails if the target queue already contains one or more messages.


The DEFTYPE, HARDENBO, and USAGE parameters of the target queue must be the same as those of the source queue.

## PING CHANNEL (test channel response)

Use the MQSC command **PING CHANNEL** to test a channel by sending data as a special message to the remote queue manager, and checking that the data is returned. The data is generated by the local queue manager.

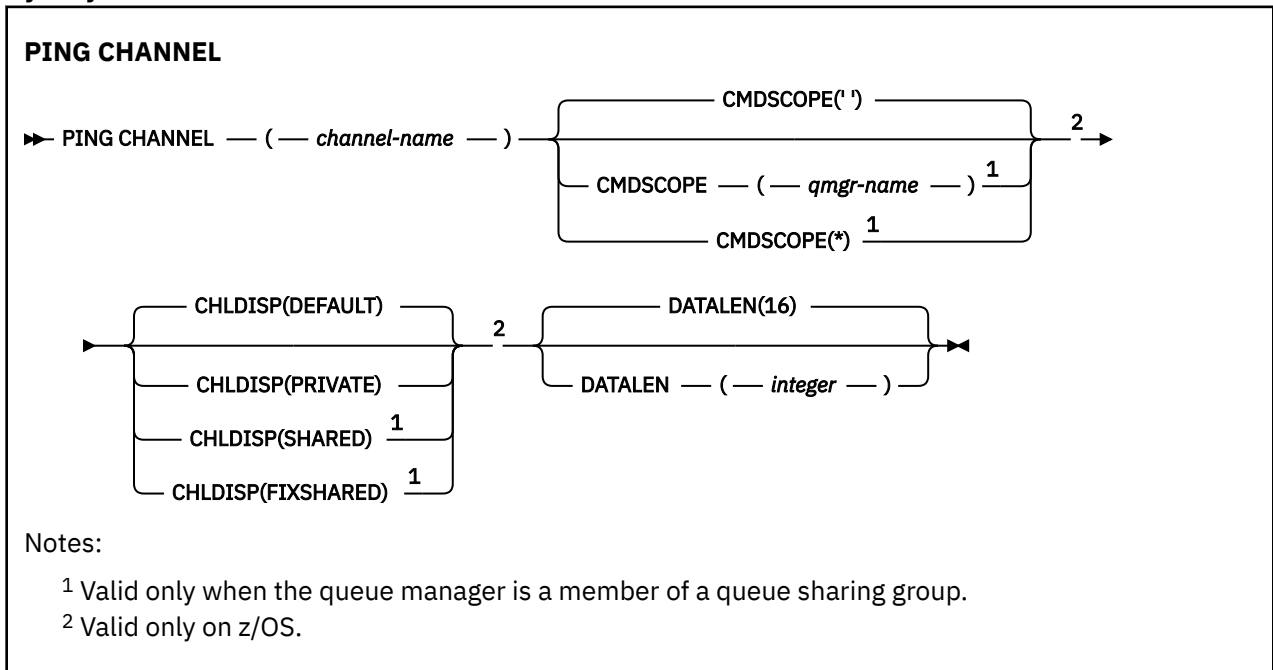
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).


 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 878](#)
- [“Parameter descriptions for PING CHANNEL” on page 878](#)

**Synonym:** PING CHL



## Usage notes

1.  On z/OS, the command server and the channel initiator must be running.
2. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager's repository.
3. This command can be used only for sender (SDR), server (SVR), and cluster-sender (CLUSDR) channels (including those that have been defined automatically). It is not valid if the channel is running; however, it is valid if the channel is stopped or in retry mode.

## Parameter descriptions for PING CHANNEL

### *(channel-name)*

The name of the channel to be tested. This is required.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

If **CHLDISP** is set to SHARED, **CMDSCOPE** must be blank or the local queue manager.

''

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

**Note:** The '\*' option is not permitted if **CHLDISP** is FIXSHARED.

### **CHLDISP**

This parameter applies to z/OS only and can take the values of:

- DEFAULT
- PRIVATE
- SHARED
- FIXSHARED

If this parameter is omitted, then the DEFAULT value applies. This is the value of the default channel disposition attribute, **DEFCDISP**, of the channel object.

In conjunction with the various values of the **CMDSCOPE** parameter, this parameter controls two types of channel:

#### **SHARED**

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of SHARED.

#### **PRIVATE**

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than SHARED.

**Note:** This disposition is **not** related to the disposition set by the disposition of the queue sharing group of the channel definition.

The combination of the **CHLDISP** and **CMDSCOPE** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of **CHLDISP** and **CMDSCOPE** are summarized in the following table.

<i>Table 175. CHLDISP and CMDSCOPE for PING CHANNEL</i>			
<b>CHLDISP</b>	<b>CMDSCOPE( ) or CMDSCOPE (local-qmgr)</b>	<b>CMDSCOPE (qmgr-name)</b>	<b>CMDSCOPE(*)</b>
PRIVATE	Ping private channel on the local queue manager	Ping private channel on the named queue manager	Ping private channel on all active queue managers
SHARED	<p>Ping a shared channel on the most suitable queue manager in the group</p> <p>This might automatically generate a command using <b>CMDSCOPE</b> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted	Not permitted
FIXSHARED	Ping a shared channel on the local queue manager	Ping a shared channel on the named queue manager	Not permitted

#### **DATALEN( integer )**

The length of the data, in the range 16 through 32 768. This is optional.

#### **Related concepts**

[Checking links using Ping](#)

#### **Related tasks**

[Using Ping to test communications](#)

## Multi PING QMGR (test queue manager response) on Multiplatforms

Use the MQSC command PING QMGR to test whether the queue manager is responsive to commands.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Usage notes” on page 880](#)

**Synonym:** PING QMGR

#### PING QMGR

► PING QMGR ◄

### Usage notes

If commands are issued to the queue manager by sending messages to the command server queue, this command causes a special message to be sent to it, consisting of a command header only, and checking that a positive reply is returned.

## ALW PURGE CHANNEL (stop and purge a channel) on AIX, Linux, and Windows

Use the MQSC command PURGE CHANNEL to stop and purge a telemetry or AMQP channel. Purging a telemetry or AMQP channel disconnects all the MQTT or AMQP clients connected to it, cleans up the state of the MQTT or AMQP clients, and stops the telemetry or AMQP channel. Cleaning the state of a client deletes all the pending publications, including any last will and testament message required by the client, and removes all the subscriptions from the client.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for PURGE CHANNEL” on page 880](#)

**Synonym:** None

#### PURGE CHANNEL

► PURGE CHANNEL — ( — *channel-name* — ) — CHLTYPE(MQTT) —  
CLIENTID — ( — *clientid* — ) ◄

### Parameter descriptions for PURGE CHANNEL

#### (*channel name*)

The name of the telemetry or AMQP channel to be stopped and purged. This parameter is required.

#### CHLTYPE (*string*)

Channel type. This parameter is required. It must follow immediately after the (*channel-name*) parameter.

The value must be either MQTT or AMQP.



### CLIENTID (*string*)

Client identifier. The client identifier is a 23 byte string that identifies an MQ Telemetry Transport or AMQP client. When the PURGE CHANNEL command specifies a CLIENTID, only the connection for the specified client identifier is purged. If the CLIENTID is not specified, all the connections on the channel are purged.

## z/OS RECOVER BSDS (recover bootstrap data set) on z/OS

Use the MQSC command RECOVER BSDS to reestablish a dual bootstrap data set (BSDS), after a data set error one has caused one to stop working.

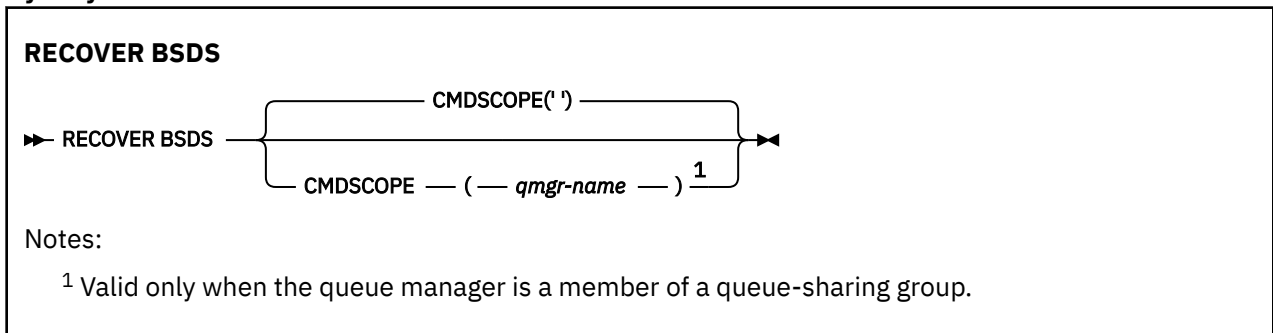
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for RECOVER BSDS” on page 881](#)
- [“Keyword and parameter descriptions for RECOVER BSDS” on page 881](#)

**Synonym:** REC BSDS



### Usage notes for RECOVER BSDS

**Note:** Command processing consists of allocating a data set with the same name as the one that encountered the error and copying onto the new data set the contents of the BSDS that does not have an error.

### Keyword and parameter descriptions for RECOVER BSDS

#### CMDSCOPE

This parameter specifies how the command is executed when the queue manager is a member of a queue-sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command is run on the queue manager on which it was entered. This is the default value.

#### *qmgr-name*

The command is run on the queue manager you specify, providing the queue manager is active within the queue-sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue-sharing group environment and if the command server is enabled.

## z/OS RECOVER CFSTRUCT (recover CF application structure) on z/OS

Use the MQSC command RECOVER CFSTRUCT to initiate recovery of CF application structures and associated shared message data sets. This command is valid only when the queue manager is a member of a queue sharing group.

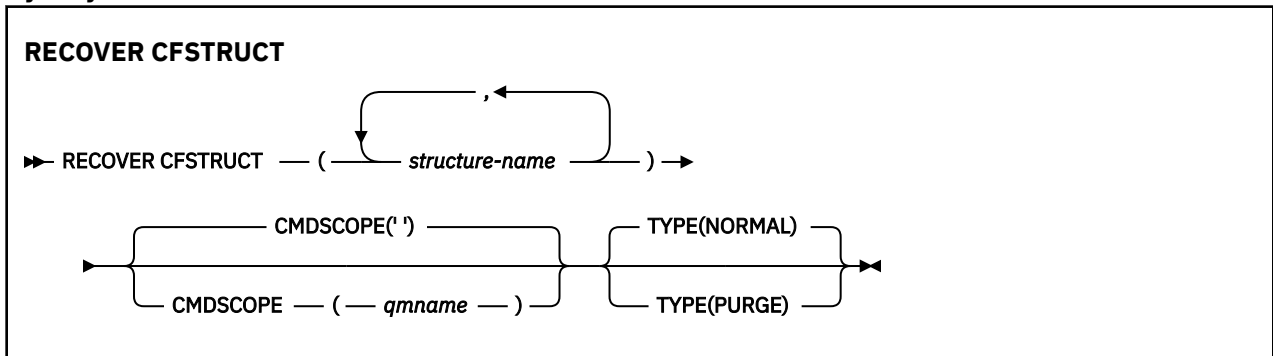
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for RECOVER CFSTRUCT” on page 882](#)
- [“Keyword and parameter descriptions for RECOVER CFSTRUCT” on page 883](#)

**Synonym:** REC CFSTRUCT



### Usage notes for RECOVER CFSTRUCT

- The command fails if neither the specified application structure nor its associated shared message data sets are flagged as being in a FAILED state.
- If a data set is marked as FAILED but the corresponding structure is not, then the **RECOVER CFSTRUCT** command changes the structure state to FAILED, deleting the contents to perform recovery. This action deletes all nonpersistent messages stored in the structure and makes the structure unavailable until recovery is complete.
- For a structure with associated shared message data sets, the **RECOVER CFSTRUCT** command recovers the structure plus the offloaded message data for any data sets which are either already marked as FAILED or found to be empty or invalid when opened by recovery processing. Any data sets which are marked as ACTIVE and have valid headers are assumed not to require recovery.
- When recovery processing completes normally, all associated shared message data sets for the recovered structures (including data sets which did not need recovery) are marked as RECOVERED, indicating that the space map needs to be rebuilt.
- Following recovery, space map rebuild processing is performed for each affected data set, to map the space occupied by the recovered message data (ignoring any existing messages which were nonpersistent or backed out). When the space map has been rebuilt for each data set, it is marked as ACTIVE again.
- The command fails if any one of the specified structure names is not defined in the CFRM policy data set.

- The recovery process is both I/O and processor intensive, and can only run on a single z/OS image. It should therefore be run on the most powerful or least busy system in the queue sharing group.
- The most likely failure is the loss of a complete CF and hence the simultaneous loss of all the application structures therein. If backup date and times are similar for each failed application structure, it is more efficient to recover them in a single **RECOVER CFSTRUCT** command.
- This command fails if any of the specified CF structures is defined with either a CFLEVEL of less than 3, or with RECOVER set to NO.
- To use TYPE(NORMAL), you must have taken a backup of the CF structures, using the **BACKUP CFSTRUCT** command.
- If backups of the requested CF structures have not been taken recently, using TYPE(NORMAL) may take a considerable amount of time.
- If a backup of the CF structure, or a required archive log, is not available, you can recover to an empty CF structure using TYPE(PURGE).
- The command **RECOVER CFSTRUCT(CSQSYSAPPL) TYPE(PURGE)** is prohibited. This is to prevent the accidental loss of queue manager internal objects.

## Keyword and parameter descriptions for RECOVER CFSTRUCT

### **CFSTRUCT( *structure-names ...* )**

Specify list of names of up to 63 structure names for which the coupling facility application structures are to be recovered, along with any associated shared message data sets which also need recovery. If resources for more than one structure need to be recovered, it is more efficient to recover them at the same time.

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### ***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

### **TYPE**

Specifies which variant of the **RECOVER** command is to be issued. Values are:

#### **NORMAL**

Perform true recovery by restoring data from a backup taken using the **BACKUP CFSTRUCT** command and reapplying logged changes since that time. Any nonpersistent messages are discarded.

This is the default.

#### **PURGE**


Reset the structure and associated shared message data sets to an empty state. This can be used to restore a working state when no backup is available, but results in the loss of all affected messages.

## REFRESH CLUSTER (rebuild a cluster)

Use the MQSC command REFRESH CLUSTER to discard all locally held cluster information and force it to be rebuilt. The command also processes any autodefined channels that are in doubt. After the command completes processing, you can perform a "cold-start" on the cluster.

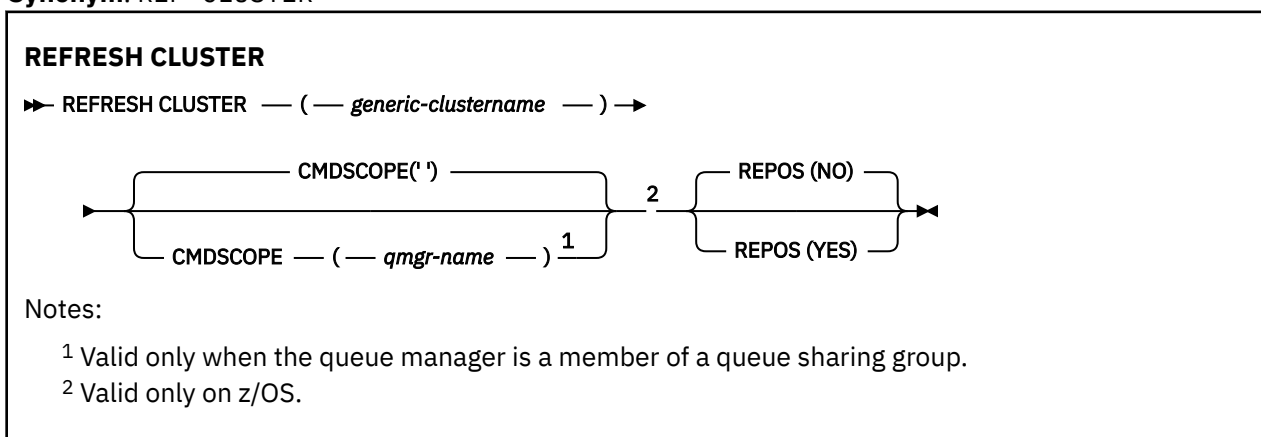
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for REFRESH CLUSTER” on page 884](#)
- [“Parameter descriptions for REFRESH CLUSTER” on page 886](#)

**Synonym:** REF CLUSTER



### Usage notes for REFRESH CLUSTER

1. Issuing **REFRESH CLUSTER** is disruptive to the cluster. It might make cluster objects invisible for a short time until the **REFRESH CLUSTER** processing completes. This can affect running applications, as described in [Application issues seen when running REFRESH CLUSTER](#). If an application is publishing or subscribing on a cluster topic, that topic might become temporarily unavailable. See [REFRESH CLUSTER considerations for publish/subscribe clusters](#). The unavailability results in a break in the publication stream until the **REFRESH CLUSTER** command completes. If the command is issued on a full repository queue manager, **REFRESH CLUSTER** might make a large volume of messages flow.
2. For large clusters, use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See [Refreshing in a large cluster can affect performance and availability of the cluster](#).
3. Quiesce all publish/subscribe applications before issuing the **REFRESH CLUSTER** command, because issuing this command in a publish/subscribe cluster disrupts delivery of publications to and from other queue managers in the cluster, and might result in proxy subscriptions from other queue managers being canceled. If this happens, resynchronize the proxy subscriptions after the cluster has refreshed, and keep all publish/subscribe applications quiesced until after the proxy subscriptions have been resynchronized. See [REFRESH CLUSTER considerations for publish/subscribe clusters](#).
4. When the command returns control to the user, it does not signify the command has completed. Activity on `SYSTEM.CLUSTER.COMMAND.QUEUE` indicates the command is still processing. See also the [REFRESH CLUSTER step in Checking that async commands for distributed networks have finished](#).



5. If cluster-sender channels are running at the time **REFRESH CLUSTER** is issued, the refresh might not complete until the channels stop and restart. To hasten completion, stop all cluster-sender channels for the cluster before you run the **REFRESH CLUSTER** command. During the processing of the **REFRESH CLUSTER** command, if the channel is not in doubt, the channel state might be re-created.
6. If you select REPOS(YES), check that all cluster-sender channels in the relevant cluster are inactive or stopped before you issue the **REFRESH CLUSTER** command.

If cluster-sender channels are running at the time you run the **REFRESH CLUSTER REPOS(YES)** command, those cluster-sender channels are ended during the operation and left in an INACTIVE state after the operation completes. Alternatively, you can force the channels to stop using the STOP CHANNEL command with MODE(FORCE).

Stopping the channels ensures that the refresh can remove the channel state, and that the channel runs with the refreshed version after the refresh completes. If the state of a channel cannot be deleted, its state is not renewed after the refresh. If a channel was stopped, it does not automatically restart. The channel state cannot be deleted if the channel is in doubt, or because it is also running as part of another cluster.

If you choose the option REPOS(YES) on full repository queue manager, you must alter it to be a partial repository. If it is the sole working repository in the cluster, the result is that there is no full repository left in the cluster. After the queue manager is refreshed, and restored to its status of a full repository, you must refresh the other partial repositories to restore a working cluster.

If it is not the sole remaining repository, you do not need to refresh the partial repositories manually. Another working full repository in the cluster informs the other members of the cluster that the full repository running the **REFRESH CLUSTER** command resumed its role as a full repository.

7. It is not normally necessary to issue a **REFRESH CLUSTER** command except in one of the following circumstances:
  - Messages were removed from either the SYSTEM . CLUSTER . COMMAND . QUEUE, or from another a cluster transmission queue, where the destination queue is SYSTEM . CLUSTER . COMMAND . QUEUE on the queue manager in question.
  - Issuing a **REFRESH CLUSTER** command is recommended by IBM Service.
  - The CLUSRCVR channels were removed from a cluster, or their CONNAME s were altered on two or more full repository queue managers while they could not communicate.
  - The same name was used for a CLUSRCVR channel on more than one queue manager in a cluster. As a result, messages destined for one of the queue managers were delivered to another. In this case, remove the duplicates, and run a **REFRESH CLUSTER** command on the single remaining queue manager with a CLUSRCVR definition.
  - RESET CLUSTER ACTION(FORCEREMOVE) was issued in error.
  - The queue manager was restarted from an earlier point in time than it finished last time it was used; for example, by restoring backed up data.
8. Issuing **REFRESH CLUSTER** does not correct mistakes in cluster definitions, nor is it necessary to issue the command after such mistakes are corrected.
9. During **REFRESH CLUSTER** processing, the queue manager generates the message AMQ9875 followed by the message AMQ9442 or AMQ9404. The queue manager might also generate the message AMQ9420. If the cluster functionality is not affected, the message AMQ9420 can be ignored.
10.  On z/OS, the command fails if the channel initiator is not started.
11.  On z/OS, any errors are reported to the console on the system where the channel initiator is running. They are not reported to the system that issued the command.

## Parameter descriptions for REFRESH CLUSTER

### ( *generic-clustername* )

The name of the cluster to be refreshed. Alternatively *generic-clustername* can be specified as "\*". If "\*" is specified, the queue manager is refreshed in all the clusters that it is a member of. If used with REPOS (YES), this forces the queue manager to restart its search for full repositories from the information in the local CLUSSDR definitions. It restarts its search, even if the CLUSSDR definitions connect the queue manager to several clusters.

The *generic-clustername* parameter is required.

### ▶ z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. '' is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered. If you do so, you must be using a queue sharing group environment and the command server must be enabled.

### REPOS

Specifies whether objects representing full repository cluster queue managers are also refreshed.

#### NO

The queue manager retains knowledge of all cluster queue manager and cluster queues marked as locally defined. It also retains knowledge of all cluster queue managers that are marked as full repositories. In addition, if the queue manager is a full repository for the cluster, it retains knowledge of the other cluster queue managers in the cluster. Everything else is removed from the local copy of the repository and rebuilt from the other full repositories in the cluster. Cluster channels are not stopped if REPOS (NO) is used. A full repository uses its CLUSSDR channels to inform the rest of the cluster that it completed its refresh.

NO is the default.

#### YES

Specifies that in addition to the REPOS (NO) behavior, objects representing full repository cluster queue managers are also refreshed. The REPOS (YES) option must not be used if the queue manager is itself a full repository. If it is a full repository, you must first alter it so that it is not a full repository for the cluster in question. The full repository location is recovered from the manually defined CLUSSDR definitions. After the refresh with REPOS (YES) is issued, the queue manager can be altered so that it is once again a full repository, if required.

▶ z/OS On z/OS, N and Y are accepted synonyms of NO and YES.

### Related concepts

[REFRESH CLUSTER considerations for publish/subscribe clusters](#)

### Related reference

[Application issues seen when running REFRESH CLUSTER](#)

### Related information


[Clustering: Using REFRESH CLUSTER best practices](#)

## REFRESH QMGR (refresh a queue manager)

Use the MQSC command REFRESH QMGR to perform special operations on queue managers.

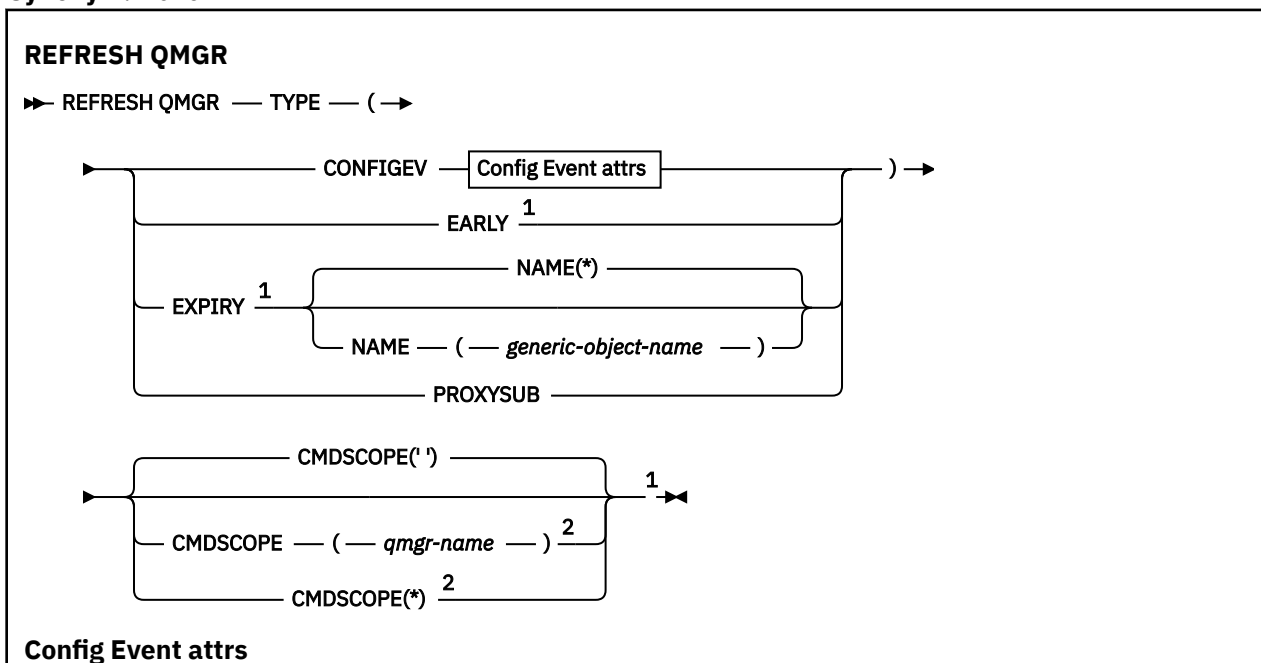
### Using MQSC commands

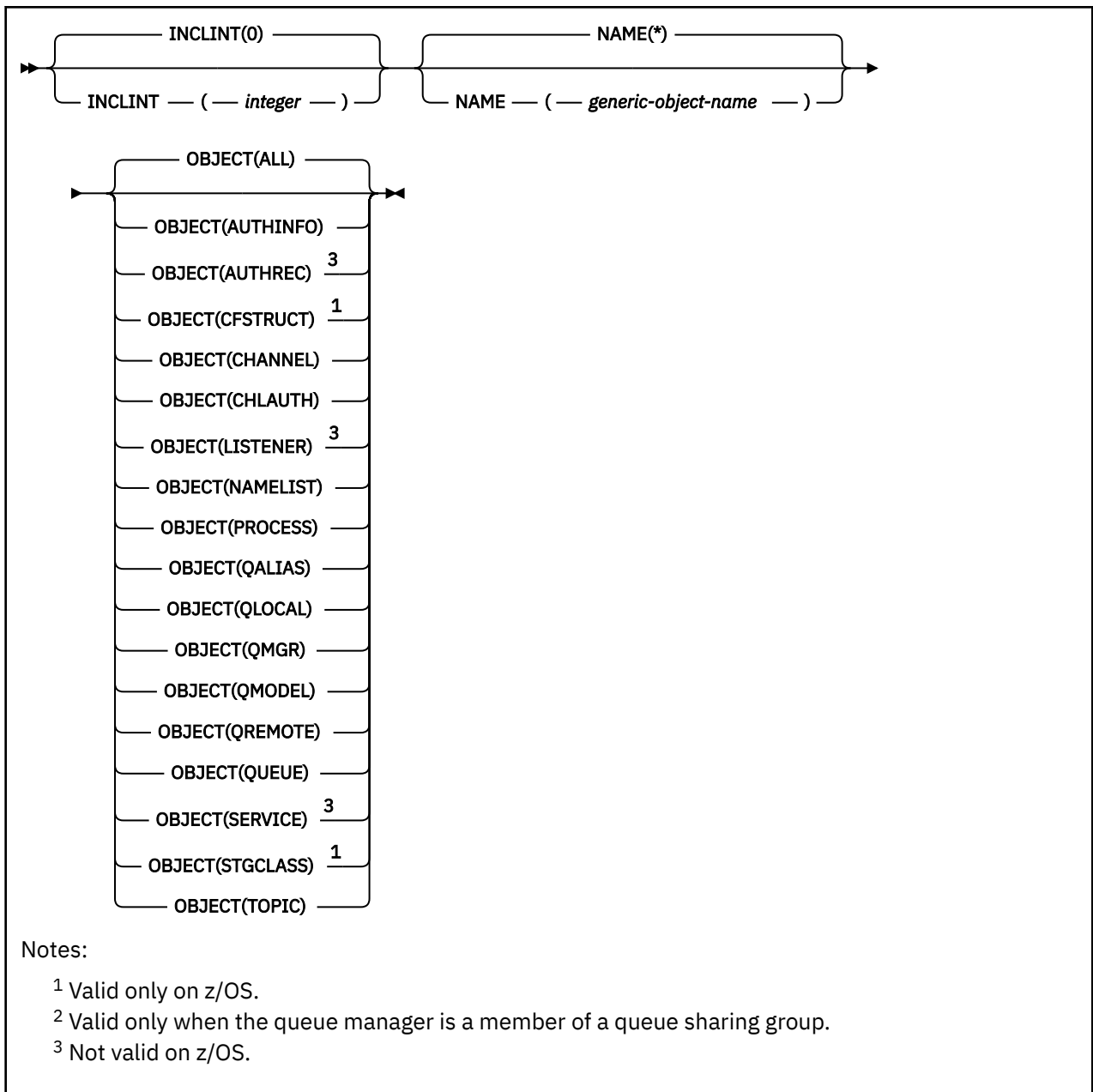
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
-  See [“Using REFRESH QMGR on z/OS” on page 888](#)
- [“Usage Notes for REFRESH QMGR” on page 889](#)
- [“Parameter descriptions for REFRESH QMGR” on page 889](#)

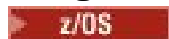
### Syntax diagram

**Synonym:** None





## Using REFRESH QMGR on z/OS



`REFRESH QMGR` can be used on z/OS. Depending on the parameters used on the command, it may be issued from various sources. For an explanation of the symbols in this table, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Table 176. `REFRESH QMGR` command and command sources



Command	Command Sources	Notes
<code>REFRESH QMGR TYPE(CONFIGEV)</code>	2CR	
<code>REFRESH QMGR TYPE(EARLY)</code>	C	Queue manager must not be active.
<code>REFRESH QMGR TYPE(EXPIRY)</code>	2CR	



Table 176. REFRESH QMGR command and command sources (continued)

Command	Command Sources	Notes
REFRESH QMGR TYPE(PROXYSUB)	2CR	CHINIT must be active to complete the command.

## Usage Notes for REFRESH QMGR

1. Issue this command with TYPE(CONFIGEV) after setting the CONFIGEV queue manager attribute to ENABLED, to bring the queue manager configuration up to date. To ensure that complete configuration information is generated, include all objects; if you have many objects, it might be preferable to use several commands, each with a different selection of objects, but such that all are included.
2. You can also use the command with TYPE(CONFIGEV) to recover from problems such as errors on the event queue. In such cases, use appropriate selection criteria, to avoid excessive processing time and event messages generation.
3. Issue the command with TYPE(EXPIRY) at any time when you believe that a queue could contain numbers of expired messages.
4.  If TYPE(EARLY) is specified, no other keywords are allowed and the command can be issued only from the z/OS console and only if the queue manager is not active.
5. You are unlikely to use **REFRESH QMGR TYPE(PROXYSUB)** other than in exceptional circumstances. See [Resynchronization of proxy subscriptions](#).
6. Successful completion of the **REFRESH QMGR TYPE(PROXYSUB)** command does not mean that the action completed. To check for true completion, see the [REFRESH QMGR TYPE\(PROXYSUB\)](#) step in [Checking that async commands for distributed networks have finished](#).
7.  If a **REFRESH QMGR TYPE(PROXYSUB)** command is issued on z/OS when the CHINIT is not running, the command is queued up and will be processed when the CHINIT starts.
8. Running the command REFRESH QMGR TYPE(CONFIGEV) OBJECT(ALL) includes authority records.  
You cannot specify the **INCLINT** and **NAME** parameters if you explicitly specify AUTHREC events. If you specify **OBJECT(ALL)** the **INCLINT** and **NAME** parameters are ignored.

## Parameter descriptions for REFRESH QMGR

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

This parameter is not valid with TYPE(EARLY).

**INCLINT (integer)**

Specifies a value in minutes defining a period immediately before the current time, and requests that only objects that have been created or changed within that period (as defined by the ALTDAT and ALTTIME attributes) are included. The value must be in the range zero through 999 999. A value of zero means there is no time limit (this is the default).

This parameter is valid only with TYPE(CONFIGEV).

**NAME (generic-object-name)**

Requests that only objects with names that match the one specified are included. A trailing asterisk (\*) matches all object names with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all objects (this is the default). NAME is ignored if OBJECT(QMGR) is specified.

This parameter is not valid with TYPE(EARLY).

**OBJECT (objtype)**

Requests that only objects of the specified type are included. (Synonyms for object types, such as QL, can also be specified.) The default is ALL, to include objects of every type.

This parameter is valid only with TYPE(CONFIGEV).

**TYPE**

This is required. Values are:

**CONFIGEV**

Requests that the queue manager generates a configuration event message for every object that matches the selection criteria specified by the OBJECT, NAME and INCLINT parameters. Matching objects defined with QSGDISP(QMGR) or QSGDISP(COPY) are always included. Matching objects defined with QSGDISP(GROUP) or QSGDISP(SHARED) are included only if the command is being executed on the queue manager where it is entered.

**EARLY**

Requests that the subsystem function routines (generally known as early code) for the queue manager replace themselves with the corresponding routines in the linkpack area (LPA).

You need to use this command only after you install new subsystem function routines (provided as corrective maintenance or with a new version or release of IBM MQ). This command instructs the queue manager to use the new routines.

 See [Update the z/OS link list and LPA](#) for more information about IBM MQ early code routines.

**EXPIRY**

Requests that the queue manager performs a scan to discard expired messages for every queue that matches the selection criteria specified by the NAME parameter. (The scan is performed regardless of the setting of the EXPRINT queue manager attribute.)

**PROXYSUB**

Requests that the queue manager resynchronizes the proxy subscriptions that are held with, and on behalf of, queue managers that are connected in a hierarchy or publish/subscribe cluster.

You should only resynchronize the proxy subscriptions in exceptional circumstances. See [Resynchronization of proxy subscriptions](#).

**REFRESH SECURITY (refresh security settings)**

Use the MQSC command REFRESH SECURITY to perform a security refresh.

**Using MQSC commands**

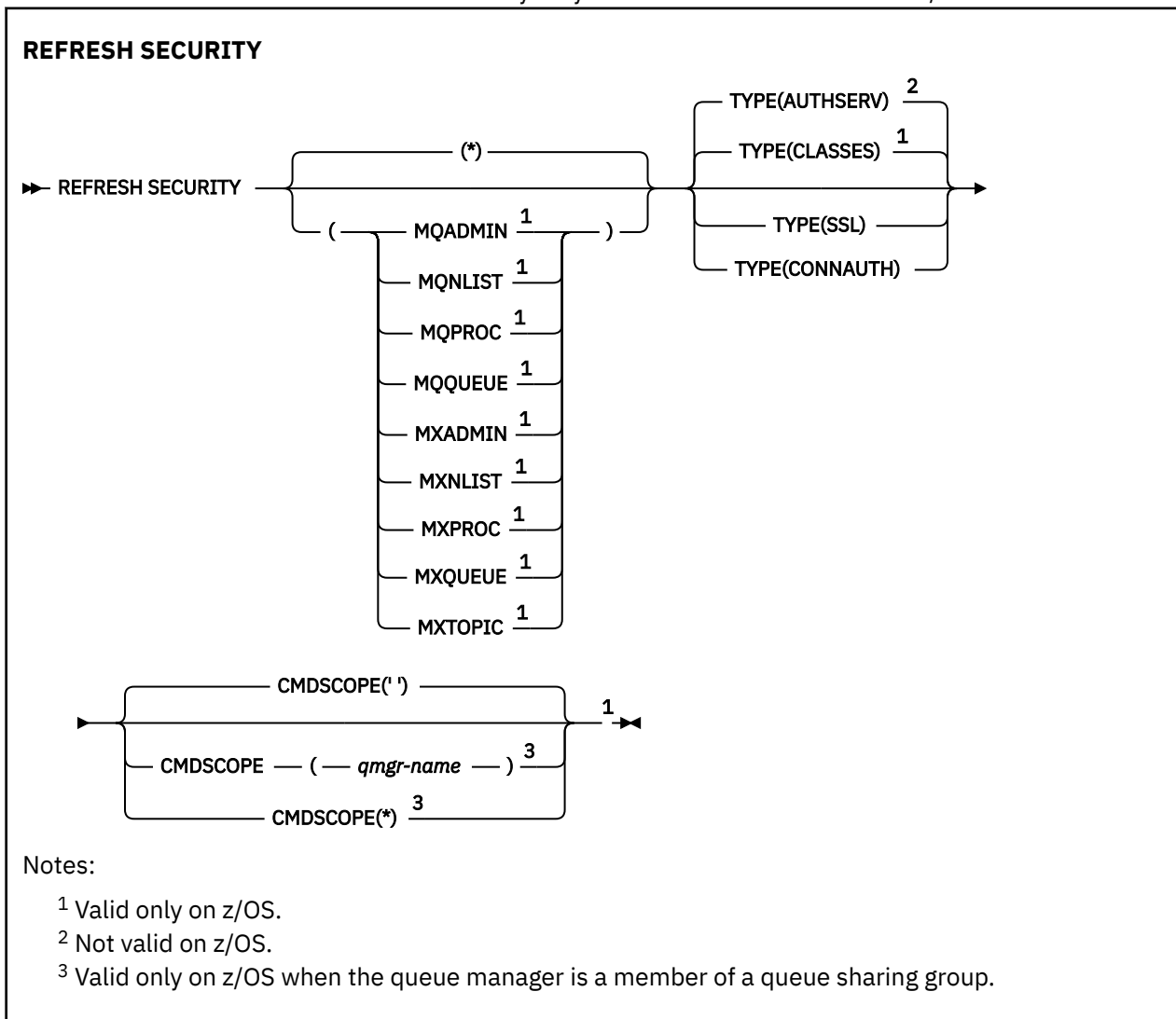
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
-  See [“Using REFRESH SECURITY on z/OS” on page 891](#)

- “Usage notes for REFRESH SECURITY” on page 892
- “Parameter descriptions for REFRESH SECURITY” on page 893

**Synonym:** REF SEC

**z/OS** REBUILD SECURITY is another synonym for REFRESH SECURITY on z/OS.



## Using REFRESH SECURITY on z/OS

**z/OS**

REFRESH SECURITY can be used on z/OS. Depending on the parameters used on the command, it may be issued from various sources. For an explanation of the symbols in this table, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Table 177. REFRESH SECURITY command and command sources

Command	Command Sources	Notes
REFRESH SECURITY TYPE(CLASSES)	CR	
REFRESH SECURITY TYPE(SSL)	CR	Not allowed from CSQINPT or CSQINP2. Channel initiator must be running.

## Usage notes for REFRESH SECURITY

When you issue the REFRESH SECURITY TYPE(SSL) MQSC command, all running TLS channels are stopped and restarted. Sometimes TLS channels can take a long time to shut down and this means that the refresh operation takes some time to complete. There is a time limit of 10 minutes for a TLS refresh to complete **z/OS** (or 1 minute on z/OS), so it can potentially take 10 minutes for the command to finish. This can give the appearance that the refresh operation has "frozen". The refresh operation will fail with an MQSC error message of AMQ9710 or PCF error MQRCCF\_COMMAND\_FAILED if the timeout is exceeded before all channels have stopped. This is likely to happen if the following conditions are true:

- The queue manager has many TLS channels running simultaneously when the refresh command is invoked
- The channels are handling large numbers of messages

If a refresh fails under these conditions, retry the command later when the queue manager is less busy. In the case where many channels are running, you can choose to stop some of the channels manually before invoking the REFRESH command.

When using TYPE(SSL):

1. **z/OS** On z/OS, the command server and channel initiator must be running.
2. **z/OS** On z/OS, IBM MQ determines whether a refresh is needed due to one, or more, of the following reasons:
  - The contents of the key repository have changed
  - The location of the LDAP server to be used for Certification Revocation Lists has changed
  - The location of the key repository has changedIf no refresh is needed, the command completes successfully and the channels are unaffected.
3. **Multi** On Multiplatforms, the command updates all TLS channels regardless of whether a security refresh is needed.
4. If a refresh is to be performed, the command updates all TLS channels currently running, as follows:
  - Sender, server and cluster-sender channels using TLS are allowed to complete the current batch. In general they then run the TLS handshake again with the refreshed view of the TLS key repository. However, you must manually restart a requester-server channel on which the server definition has no CONNAME parameter.
  - AMQP channels using TLS are restarted, with any currently connected clients being forcibly disconnected. The client receives an amqp:connection:forced AMQP error message.
  - All other channel types using TLS are stopped with a STOP CHANNEL MODE(FORCE) STATUS(INACTIVE) command. If the partner end of the stopped message channel has retry values defined, the channel retries and the new TLS handshake uses the refreshed view of the contents of the TLS key repository, the location of the LDAP server to be used for Certification Revocation Lists, and the location of the key repository. In the case of a server-connection channel, the client application loses its connection to the queue manager and has to reconnect in order to continue.

**z/OS** When using TYPE(CLASSES):

- Classes MQADMIN, MQNLIST, MQPROC, and MQQUEUE can only hold profiles defined in uppercase.
- Classes MXADMIN, MXNLIST, MXPROC, and MQXUEUE can hold profiles defined in mixed case.
- Class MXTOPIC can be refreshed whether using uppercase or mixed case classes. Although it is a mixed case class, it is the only mixed case class that can be active with either group of classes.
- The MQCMD and MQCONN classes cannot be specified, and are not included by REFRESH SECURITY CLASS(\*).

Security information from the MQCMD and MQCONN classes is not cached in the queue manager. See [Refreshing queue manager security on z/OS](#) for further information.

## Notes:

1. Performing a REFRESH SECURITY(\*) TYPE (CLASSES) operation is the only way to change the classes being used by your system from uppercase-only support to mixed case support.  
Do this by checking the queue manager attribute SCYCASE to see if it is set to UPPER or MIXED
2. It is your responsibility to ensure that you have copied, or defined, all the profiles you need in the appropriate classes before you carry out a REFRESH SECURITY(\*) TYPE (CLASSES) operation.
3. A refresh of an individual class is allowed only if the classes currently being used are of the same type. For example, if MQPROC is in use, you can issue a refresh for MQPROC but not MXPROC.

## Parameter descriptions for REFRESH SECURITY

The command qualifier allows you to indicate more precise behavior for a specific TYPE value. Select from:

\*

A full refresh of the type specified is performed. **z/OS** This is the default value on z/OS systems.

### **z/OS** MQADMIN

Valid only if TYPE is CLASSES. Specifies that Administration type resources are to be refreshed. Valid on z/OS only.

**Note:** If, when refreshing this class, it is determined that a security switch relating to one of the other classes has been changed, a refresh for that class also takes place.

### **z/OS** MQNLIST

Valid only if TYPE is CLASSES. Specifies that Namelist resources are to be refreshed. Valid on z/OS only.

### **z/OS** MQPROC

Valid only if TYPE is CLASSES. Specifies that Process resources are to be refreshed. Valid on z/OS only.

### **z/OS** MQQUEUE

Valid only if TYPE is CLASSES. Specifies that Queue resources are to be refreshed. Valid on z/OS only.

### **z/OS** MXADMIN

Valid only if TYPE is CLASSES. Specifies that administration type resources are to be refreshed. Valid on z/OS only.

**Note:** If, when refreshing this class, it is determined that a security switch relating to one of the other classes has been changed, a refresh for that class also takes place.

### **z/OS** MXNLIST

Valid only if TYPE is CLASSES. Specifies that namelist resources are to be refreshed. Valid on z/OS only.

### **z/OS** MXPROC

Valid only if TYPE is CLASSES. Specifies that process resources are to be refreshed. Valid on z/OS only.

### **z/OS** MXQUEUE

Valid only if TYPE is CLASSES. Specifies that queue resources are to be refreshed. Valid on z/OS only.


### **z/OS** MXTOPIC

Valid only if TYPE is CLASSES. Specifies that topic resources are to be refreshed. Valid on z/OS only.

### **z/OS** CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value  for non-z/OS systems.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## **TYPE**

Specifies the type of refresh that is to be performed.

### **AUTHSERV**

The list of authorizations held internally by the authorization services component is refreshed.

This is the default value.

### **CLASSES**

IBM MQ in-storage ESM (external security manager, for example RACF ) profiles are refreshed. The in-storage profiles for the resources being requested are deleted. New entries are created when security checks for them are performed, and are validated when the user next requests access.



You can select specific resource classes for which to perform the security refresh.

This is valid only on z/OS where it is the default.


## **CONNAUTH**

Refreshes the cached view of the configuration for connection authentication.

The connection authentication configuration consists of the following resources:

- The AUTHINFO object that is referenced by the queue manager's **CONNAUTH** attribute.
-  The authentication token configuration in the **AuthToken** stanza of the `qm.ini` file.
-  The keystore that contains certificates and symmetric keys that are used to validate authentication tokens. This keystore is referenced by the **KeyStore** attribute of the **AuthToken** stanza.

You must refresh the configuration before the queue manager recognizes changes to these resources.

 On [Multiplatforms](#), this is a synonym for AUTHSERV.

See [Connection authentication](#) for more information.

## **SSL**


Refreshes the cached view of the Secure Sockets Layer, or Transport Layer Security, key repository and allows updates to become effective on successful completion of the command. Also refreshed are the locations of:

- the LDAP servers to be used for Certified Revocation Lists
- the key repository

as well as any cryptographic hardware parameters specified through IBM MQ.

To refresh CHLAUTH use the [“REFRESH QMGR \(refresh a queue manager\)” on page 887](#) command.

### Related tasks

 [Refreshing queue manager security on z/OS](#)

### **RESET CFSTRUCT (reset a CF application structure) on z/OS**

Use the MQSC command RESET CFSTRUCT to modify the status of a specific application structure.

### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Notes:” on page 895](#)
- [“Parameter descriptions for RESET CFSTRUCT” on page 895](#)

**Synonym:** None.

#### **RESET CFSTRUCT**

➤ RESET CFSTRUCT (*structure-name*) ACTION(FAIL) ➤

### Notes:

1. Valid only when the queue manager is a member of a queue sharing group.
2. RESET CFSTRUCT requires the structure to be defined with CFLEVEL(5).

### Parameter descriptions for RESET CFSTRUCT

#### **CFSTRUCT( *structure-name* )**

Specify the name of the coupling facility application structure that you want to reset.

#### **ACTION(FAIL)**

Specify this keyword to simulate a structure failure and set the status of the application structure to FAILED.


**Note:** Failing a structure deletes all nonpersistent messages stored in the structure, and makes the structure unavailable until recovery is complete. Structure recovery can take a long time to complete. Therefore, this action should be used only in a situation where you can resolve a problem with the structure by forcing the structure to be reallocated and recovered.

### **RESET CHANNEL (reset message sequence number for a channel)**

Use the MQSC command **RESET CHANNEL** to reset the message sequence number for an IBM MQ channel with, optionally, a specified sequence number to be used the next time that the channel is started.

### Using MQSC commands

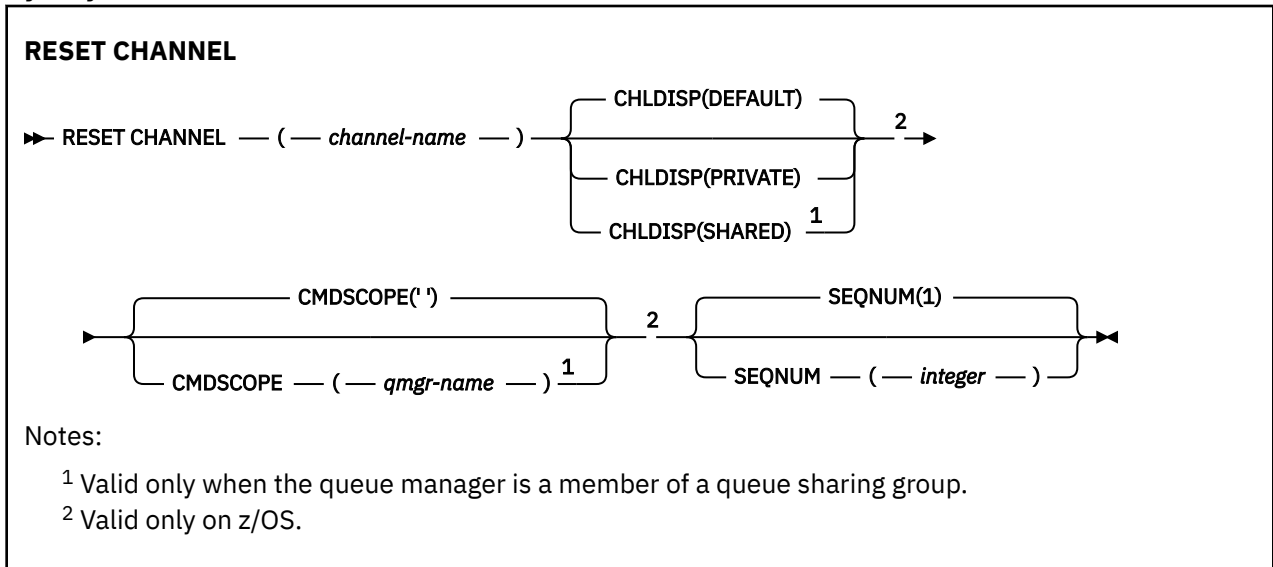
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)

- “Usage notes” on page 896
- “Parameter descriptions for RESET CHANNEL” on page 896

**Synonym:** RESET CHL



## Usage notes

1. **z/OS** On z/OS, the command server and channel initiator must be running.
2. This command can be issued to a channel of any type except SVRCONN and CLNTCONN channels, (including those that have been defined automatically). However, if it is issued to a sender or server channel, then in addition to resetting the value at the end at which the command is issued, the value at the other (receiver or requester) end is also reset to the same value the next time this channel is initiated (and resynchronized if necessary). Issuing this command on a cluster-sender channel might reset the message sequence number at either end of the channel. However, this is not significant because the sequence numbers are not checked on clustering channels.
3. If the command is issued to a receiver, requester, or cluster-receiver channel, the value at the other end is not reset as well; this must be done separately if necessary.
4. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager's repository.
5. If the message is non-persistent, and the **RESET CHANNEL** command is issued to the sender channel, reset data is sent and flows every time the channel starts.

## Parameter descriptions for RESET CHANNEL

### (channel-name)

The name of the channel to be reset. This is required.

### **z/OS** CHLDISP

This parameter applies to z/OS only and can take the values of:

- DEFAULT
- PRIVATE
- SHARED

If this parameter is omitted, then the DEFAULT value applies. This is taken from the default channel disposition attribute, **DEFCDISP**, of the channel object.



In conjunction with the various values of the **CMDSCOPE** parameter, this parameter controls two types of channel:

**SHARED**

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of SHARED.

**PRIVATE**

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than SHARED.

**Note:** This disposition is **not** related to the disposition set by the disposition of the queue sharing group of the channel definition.

The combination of the **CHLDISP** and **CMDSCOPE** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.

The various combinations of **CHLDISP** and **CMDSCOPE** are summarized in the following table:

<i>Table 178. CHLDISP and CMDSCOPE for RESET CHANNEL</i>		
<b>CHLDISP</b>	<b>CMDSCOPE( ) or CMDSCOPE (local-qmgr)</b>	<b>CMDSCOPE (qmgr-name)</b>
PRIVATE	Reset private channel on the local queue manager	Reset private channel on the named queue manager
SHARED	Reset a shared channel on all active queue managers.  This might automatically generate a command using <b>CMDSCOPE</b> and send it to the appropriate queue managers. If there is no definition for the channel on the queue managers to which the command is sent, or if the definition is unsuitable for the command, the action fails there.  The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.	Not permitted

**z/OS CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

If **CHLDISP** is set to SHARED, **CMDSCOPE** must be blank or the local queue manager.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name only if you are using a queue sharing group environment and if the command server is enabled.

### SEQNUM( *integer* )

The new message sequence number, which must be in the range 1 through 999 999 999. This is optional.

### Related tasks

[Troubleshooting a problem where a channel refuses to run](#)

## RESET CLUSTER (reset a cluster)

Use the MQSC command **RESET CLUSTER** to perform special operations on clusters.

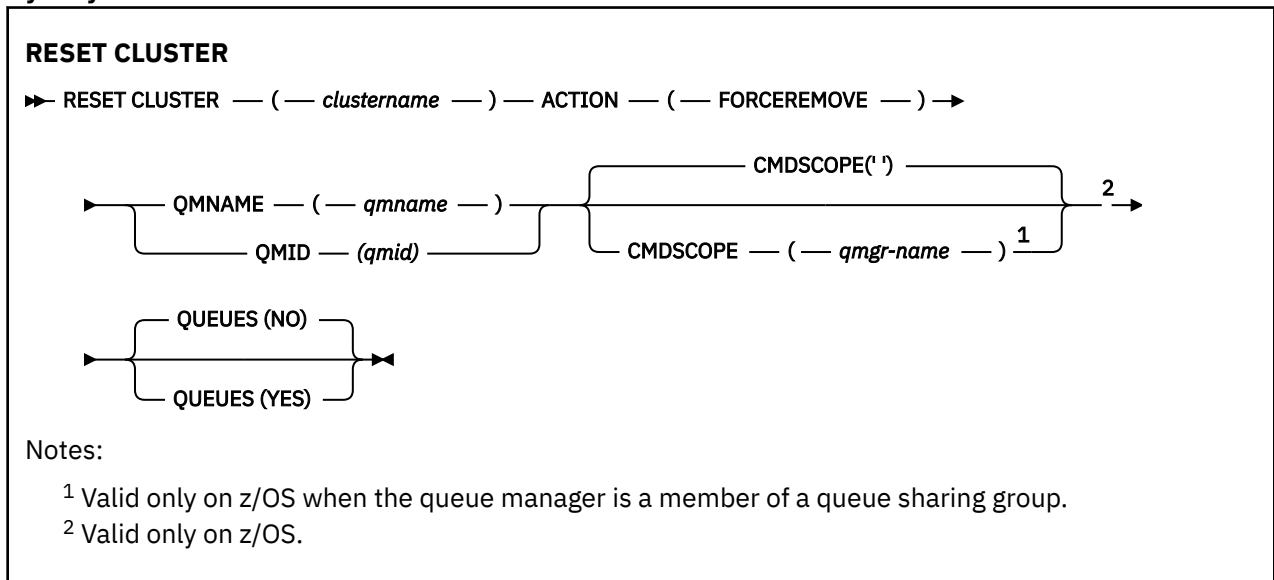
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for RESET CLUSTER” on page 898](#)
- [“Parameter descriptions for RESET CLUSTER” on page 899](#)

**Synonym:** None



### Usage notes for RESET CLUSTER

- **z/OS** On z/OS, the command fails if the channel initiator has not been started.
- **z/OS** On z/OS, any errors are reported to the console on the system where the channel initiator is running; they are not reported to the system that issued the command.
- To avoid any ambiguity, it is preferable to use QMID rather than QMNAME. The queue manager identifier can be found by commands such as DISPLAY QMGR and DISPLAY CLUSQMGR.

If QMNAME is used, and there is more than one queue manager in the cluster with that name, the command is not actioned.

- If you use characters other than those listed in [Rules for naming IBM MQ objects](#) in your object or variable names, for example in QMID, you must enclose the name in quotation marks.

- If you remove a queue manager from a cluster using this command, you can rejoin it to the cluster by issuing a **REFRESH CLUSTER** command. Wait at least 10 seconds before issuing a **REFRESH CLUSTER** command, because the repository ignores any attempt to rejoin the cluster within 10 seconds of a **RESET CLUSTER** command. If the queue manager is in a publish/subscribe cluster, you then need to reinstate any required proxy subscriptions. See [REFRESH CLUSTER considerations for publish/subscribe clusters](#).

**Note:** For large clusters, use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See [Refreshing in a large cluster can affect performance and availability of the cluster](#).

- Successful completion of the command does not mean that the action completed. To check for true completion, see the [RESET CLUSTER](#) step in [Checking that async commands for distributed networks have finished](#).

## Parameter descriptions for RESET CLUSTER

### **(*clustername*)**

The name of the cluster to be reset. This is required.

### **ACTION(FORCEREMOVE)**

Requests that the queue manager is forcibly removed from the cluster. This might be needed to ensure correct cleanup after a queue manager has been deleted.

This action can be requested only by a full repository queue manager.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### ***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

### **QMID(*qmid*)**

The identifier of the queue manager to be forcibly removed.

### **QMNAME(*qmname*)**

The name of the queue manager to be forcibly removed.

### **QUEUES**


Specifies whether cluster queues owned by the queue manager being force removed are removed from the cluster.

#### **NO**

Cluster queues owned by the queue manager being force removed are not removed from the cluster. This is the default.

#### **YES**

Cluster queues owned by the queue manager being force removed are removed from the cluster in addition to the cluster queue manager itself. The cluster queues are removed even if the cluster queue manager is not visible in the cluster, perhaps because it was previously force removed without the QUEUES option.

 On z/OS, **N** and **Y** are accepted synonyms of **NO** and **YES**.

## Related reference

[RESET CLUSTER: Forcibly removing a queue manager from a cluster](#)

## RESET QMGR (reset a queue manager)

Use the MQSC command RESET QMGR as part of your backup and recovery procedures.

## Using MQSC commands

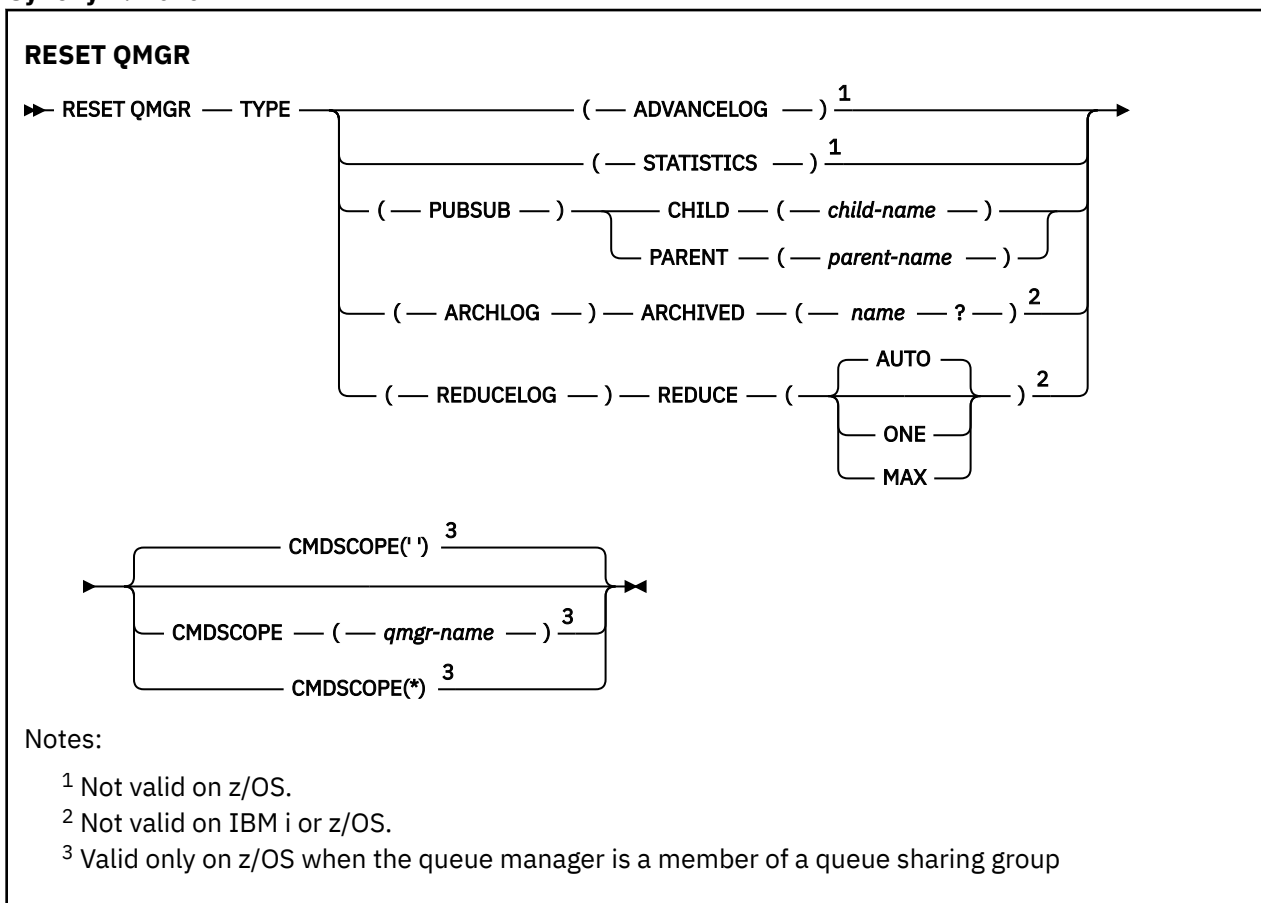
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

**Multi** Use the **TYPE (ARCHLOG)** option to notify the queue manager that all log extents, up to the specified one, have been archived. If the log management type is not ARCHIVE, the command fails. Use the **TYPE (REDUCELOG)** option to request that the queue manager reduces the number of log extents, provided they are no longer required.

- [Syntax diagram](#)
- [“Usage notes for RESET QMGR” on page 900](#)
- [“Parameter descriptions for RESET QMGR” on page 902](#)

**Synonym:** None



## Usage notes for RESET QMGR

- You can use this command to request that the queue manager starts writing to a new log extent, making the previous log extent available for backup. See [Updating a backup queue manager](#). Alternatively, you can use this command to request that the queue manager ends the current statistics collection

period and writes the collected statistics. You can also use this command to forcibly remove a publish/subscribe hierarchical connection for which this queue manager is nominated as either the parent or the child in the hierarchical connection.

- The queue manager might refuse a request to advance the recovery log, if advancing the recovery log would cause the queue manager to become short of space in the active log.
- You are unlikely to use **RESET QMGR TYPE(PUBSUB)** other than in exceptional circumstances. Typically the child queue manager uses **ALTER QMGR PARENT(' ')** to remove the hierarchical connection.
- When you need to disconnect from a child or parent queue manager with which the queue manager has become unable to communicate, you must issue the **RESET QMGR TYPE (PUBSUB)** command from a queue manager. When using this command, the remote queue manager is not informed of the canceled connection. It might, therefore, be necessary to issue the **ALTER QMGR PARENT(' ')** command at the remote queue manager. If the child queue manager is not manually disconnected, it is forcibly disconnected and the parent status is set to REFUSED.
- If you are resetting the parent relationship, issue the **ALTER QMGR PARENT(' ')** command, otherwise the queue manager attempts to re-establish the connection when the publish/subscribe capability of the queue manager is later enabled.
- Successful completion of the **RESET QMGR TYPE(PUBSUB)** command does not mean that the action completed. To check for true completion, see the **RESET QMGR TYPE(PUBSUB)** step in [Checking that async commands for distributed networks have finished](#).
- You must specify one only of **ADVANCELOG, STATISTICS, PUBSUB, ARCHLOG** or **REDUCELOG**.

## Usage notes for TYPE(ARCHLOG)

Multi

This option requires change authority on the queue manager object.

The command fails if the log extent is not recognized, or is the current log.

If, for some reason, the programmatic way that your enterprise notifies your log extents are archived is not working, and the disk is filling up with log extents, your administrator can use this command.

You need to determine yourself, the name to pass in from your archiving process, as to what has already been archived.

## Usage notes for TYPE(REDUCELOG)

Multi

This option requires change authority on the queue manager object.

You should not need this command in normal circumstances. In general, when using automatic management of log files, you should leave it up to the queue manager to reduce the number of log extents as necessary.

For circular logging, you can use this option to remove inactive secondary log extents. A growth in secondary log extents is usually noticed by an increase in disk usage, often due to some specific issue in the past.

**Note:** For circular logging the command might not be able reduce the log extents by the required number immediately. In that case, the command returns, and the reduction takes place asynchronously at some later point.

For linear logging this can remove log extents that are not required for recovery (and have been archived if you are using archive log management) as noticed by a high value for [REUSESZ](#) on the **DISPLAY QMSTATUS** command.

You should run this command only after some specific event that has caused the number of log extents to be extraordinarily large.

The command blocks until the chosen number of extents have been deleted. Note that the command does not return the number of extents that have been removed, but a queue manager error log message is written, indicating what has taken place.

## Parameter descriptions for RESET QMGR

### TYPE

#### ADVANCELOG

Requests that the queue manager starts writing to a new log extent, making the previous log extent available for backup. See [Updating a backup queue manager](#). This command is accepted only if the queue manager is configured to use linear logging.

#### Multi ARCHLOG

##### ARCHIVED ( *name* )

Notifies the queue manager that this extent, and all logically earlier ones, have been archived. The extent name is, for example, S0000001.LOG or AMQA000001 on IBM i.

#### PUBSUB

Requests that the queue manager cancels the indicated publish/subscribe hierarchical connection. This value requires that one of the CHILD or PARENT attributes is specified:

##### CHILD

The name of the child queue manager for which the hierarchical connection is to be forcibly canceled. This attribute is used only with TYPE(PUBSUB). It cannot be used together with PARENT.

##### PARENT

The name of a parent queue manager for which the hierarchical connection is to be forcibly canceled. This attribute is used only with TYPE(PUBSUB). It cannot be used together with CHILD.

#### Multi REDUCELOG

##### REDUCE

Requests the queue manager to reduce the number of inactive or superfluous log extents and the way in which the log extents are reduced.

The value can be one of the following:

##### AUTO

Reduce the log extents by an amount chosen by the queue manager.

##### ONE

Reduce the log extents by one extent, if possible.

##### MAX

Reduce the log extents by the maximum number possible.

#### STATISTICS

Requests that the queue manager ends the current statistics collection period and writes the collected statistics.

#### z/OS CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE must be blank, or the local queue manager, if QSGDISP is set to GROUP.

..

The command runs on the queue manager on which it was entered. This value is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name other than the queue manager on which it was entered, only if you are using a shared queue environment and if the command server is enabled.

**\***

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of setting this value is the same as entering the command on every queue manager in the queue sharing group.

## **z/OS RESET QSTATS (report and reset queue performance data) on z/OS**

Use the MQSC command RESET QSTATS to report performance data for a queue and then to reset that data.

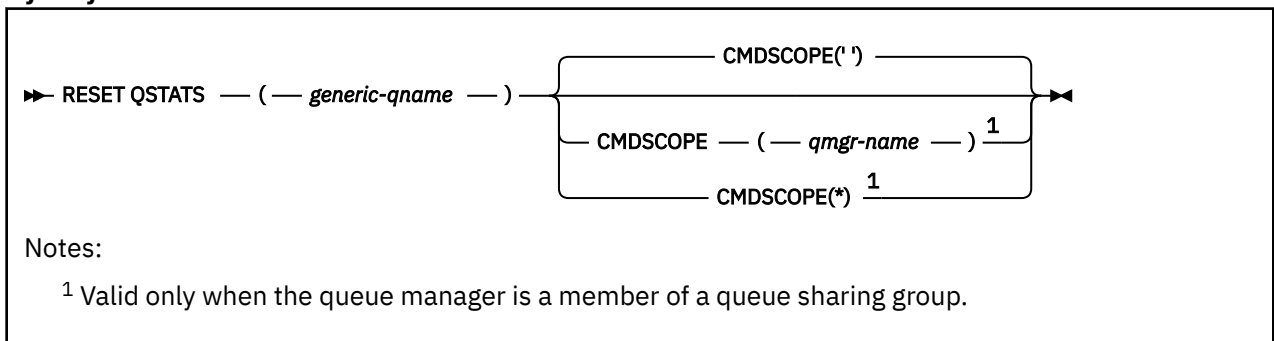
### **Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for RESET QSTATS” on page 903](#)
- [“Parameter descriptions for RESET QSTATS” on page 904](#)

**Synonym:** None



### **Usage notes for RESET QSTATS**

1. If there is more than one queue with a name that satisfies the *generic q-name*, all those queues are reset.
2. Issue this command from an application, and not the z/OS console or its equivalent, to ensure that the statistical information is recorded.
3. The following information is kept for all queues, both private and shared. For shared queues each queue manager keeps an independent copy of the information:

#### **MSGIN**

Incremented each time a message is put to the shared queue

#### **MSGOUT**

Incremented each time a message is removed from the shared queue

#### **HIQDEPTH**

Calculated by comparing the current value for HIQDEPTH held by this queue manager with the new queue depth obtained from the coupling facility during every put operation. The depth of the queue is affected by all queue managers putting messages to the queue or getting messages from it.

To retrieve the information and obtain full statistics for a shared queue, specify **CMDSCOPE (\*)** to broadcast the command to all queue managers in the queue sharing group.

The peak queue depth approximates to the maximum of all the returned HIQDEPTH values, the total MQPUT count approximates to the sum of all the returned MSGIN values, and the total MQGET count approximates to the sum of all the returned MSGOUT values.

4. If the PERFMEV attribute of the queue manager is DISABLED, the command fails.

## Parameter descriptions for RESET QSTATS

### *generic-qname*

The name of the local queue with a disposition of QMGR, COPY, or SHARED, but not GROUP, with performance data that is to be reset.

A trailing asterisk (\*) matches all queues with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all queues.

The performance data is returned in the same format as parameters returned by DISPLAY commands. The data is:

### **QSTATS**

The name of the queue



The disposition of the queue, that is, QMGR, COPY, or SHARED.

### **RESETINT**

The number of seconds since the statistics were last reset.

### **HIQDEPTH**

The peak queue depth since the statistics were last reset.

### **MSGIN**

The number of messages that have been added to the queue by MQPUT and MQPUT1 calls since the statistics were last reset.

The count includes messages added to the queue in units of work that have not yet been committed, but the count is not decremented if the units of work are later backed out. The maximum displayable value is 999 999 999; if the number exceeds this value, 999 999 999 is displayed.

### **MSGOUT**

The number of messages removed from the queue by destructive (non-browse) MQGET calls since the statistics were last reset.

The count includes messages removed from the queue in units of work that have not yet been committed, but the count is not decremented if the units of work are subsequently backed out. The maximum displayable value is 999 999 999; if the number exceeds this value, 999 999 999 is displayed.

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.



\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## Example output

The following example, shows the output from the command on z/OS.

```

12.44.16 STC16696 CSQM201I !MQ13 CSQMDRTC  RESET QSTATS DETAILS  902
902      QSTATS(CICS01.INITQ)
902      QSGDISP(QMGR)
902      RESETINT(43)
902      HIQDEPTH(0)
902      MSGSIN(0)
902      MSGSOUT(0)
902      END QSTATS DETAILS
12.44.16 STC16696 CSQM201I !MQ13 CSQMDRTC  RESET QSTATS DETAILS  903
903      QSTATS(MQ13.DEAD.QUEUE)
903      QSGDISP(QMGR)
903      RESETINT(43)
903      HIQDEPTH(0)
903      MSGSIN(0)
903      MSGSOUT(0)
903      END QSTATS DETAILS
12.44.16 STC16696 CSQM201I !MQ13 CSQMDRTC  RESET QSTATS DETAILS  904
904      QSTATS(SYSTEM.ADMIN.ACTIVITY.QUEUE)
904      QSGDISP(QMGR)
904      RESETINT(43)
904      HIQDEPTH(0)
904      MSGSIN(0)
904      MSGSOUT(0)

```

## **z/OS** RESET SMDS (reset shared message data sets) on z/OS

Use the MQSC command RESET SMDS to modify availability or status information relating to one or more shared message data sets associated with a specific application structure.

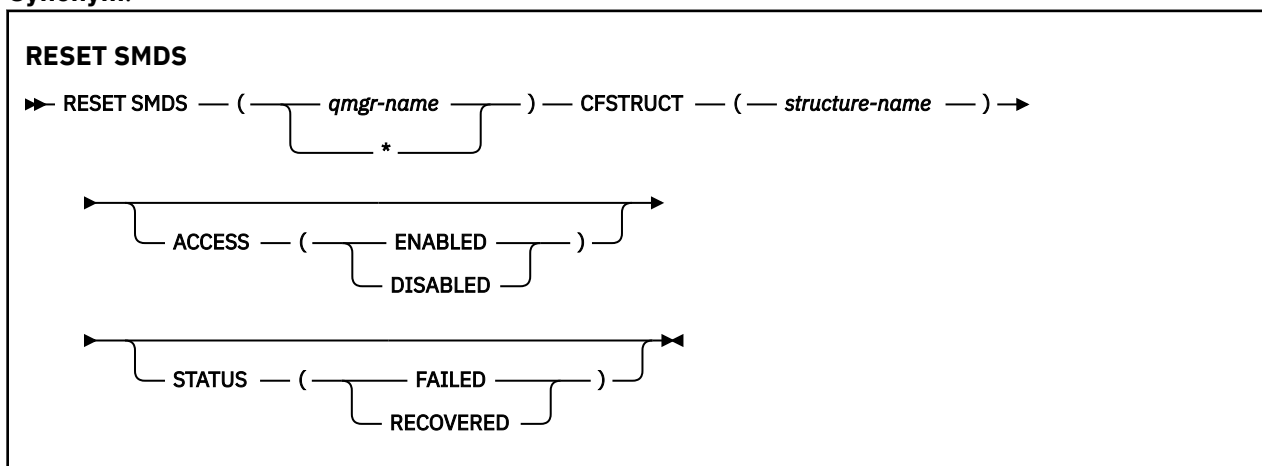
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for RESET SMDS” on page 906](#)

Synonym:



## Parameter descriptions for RESET SMDS

This command is only supported when the CFSTRUCT definition is currently using the option OFFLOAD(SMDS).

### **SMDS(*qmgr-name*\*)**

Specify the queue manager for which the shared message data set availability or status information is to be modified, or an asterisk to modify the information for all data sets associated with the specified CFSTRUCT.

### **CFSTRUCT( *structure-name* )**

Specify the coupling facility application structure for which the availability or status information for one or more shared message data sets is to be modified.

### **ACCESS( ENABLED|DISABLED )**

This keyword is used to enable and disable access to a shared message data set, making it available or unavailable to the queue managers in the group.

This keyword is useful when a shared message data set is required to be temporarily unavailable, for example while moving it to a different volume. In this instance, the keyword would be used to mark the data set as ACCESS(DISABLED) causing all of the queue managers to close it normally and deallocate it. When the data set is ready to be used, it can be marked as ACCESS(ENABLED) allowing the queue managers to access it again.

#### **ENABLED**

Use the ENABLED parameter to enable access to the shared message data set after previously disabling access, or to retry access after an error has caused the availability state to be set to ACCESS(SUSPENDED).

#### **DISABLED**

Use the DISABLED parameter to indicate that the shared message data set cannot be used until the access has been changed back to ENABLED. Any queue managers currently connected to the shared message data set are disconnected from it.

### **STATUS(FAILED | RECOVERED)**

This keyword is used to specify that a shared message data set requires recovery/repair, or to reset the STATUS of the data set from FAILED.

If you have detected that a data set is in need of repair, this keyword can be used to manually mark the data set as STATUS(FAILED). If the queue manager detects that the data set requires repair, it automatically marks it as STATUS(FAILED). Then if RECOVER CFSTRUCT is used to successfully complete a repair to the data set, the queue manager automatically marks it as STATUS(RECOVERED). If another method is used to successfully repair the data set, this keyword can be used to manually mark the data set as STATUS(RECOVERED). It is not necessary to manually alter the ACCESS, as it is automatically changed to SUSPENDED while the STATUS is FAILED and then back to ENABLED when the STATUS is set to RECOVERED.

#### **FAILED**

Use the FAILED parameter to indicate that the shared message data set needs to be recovered or repaired, and should not be used until this has been completed. This is only allowed if the current state is STATUS(ACTIVE) or STATUS(RECOVERED). If the current availability state is ACCESS(ENABLED) and is not changed on the same command, this sets ACCESS(SUSPENDED) to prevent further attempts to use the shared message data set until it has been repaired. Any queue managers currently connected to the shared message data set are forced to disconnect from it, by closing and deallocating the data set. This status may be set automatically if a permanent I/O error occurs when accessing a shared message data set or if a queue manager determines that header information in the data set is invalid or is inconsistent with the current state of the structure.

#### **RECOVERED**

Use the RECOVERED parameter to reset the state from STATUS(FAILED) if the shared message data set does not actually need to be recovered, for example if it was merely temporarily unavailable. If the current availability state (after any change specified on the same command) is ACCESS(SUSPENDED), this sets ACCESS(ENABLED) to allow the owning queue manager to open

the shared message data set and perform restart processing, after which the status is changed to STATUS(ACTIVE) and other queue managers can use it again.

## **z/OS** RESET TPIPE (reset sequence numbers for an IMS Tpipe) on z/OS

Use the MQSC command RESET TPIPE to reset the recoverable sequence numbers for an IMS Tpipe used by the IBM MQ - IMS bridge.

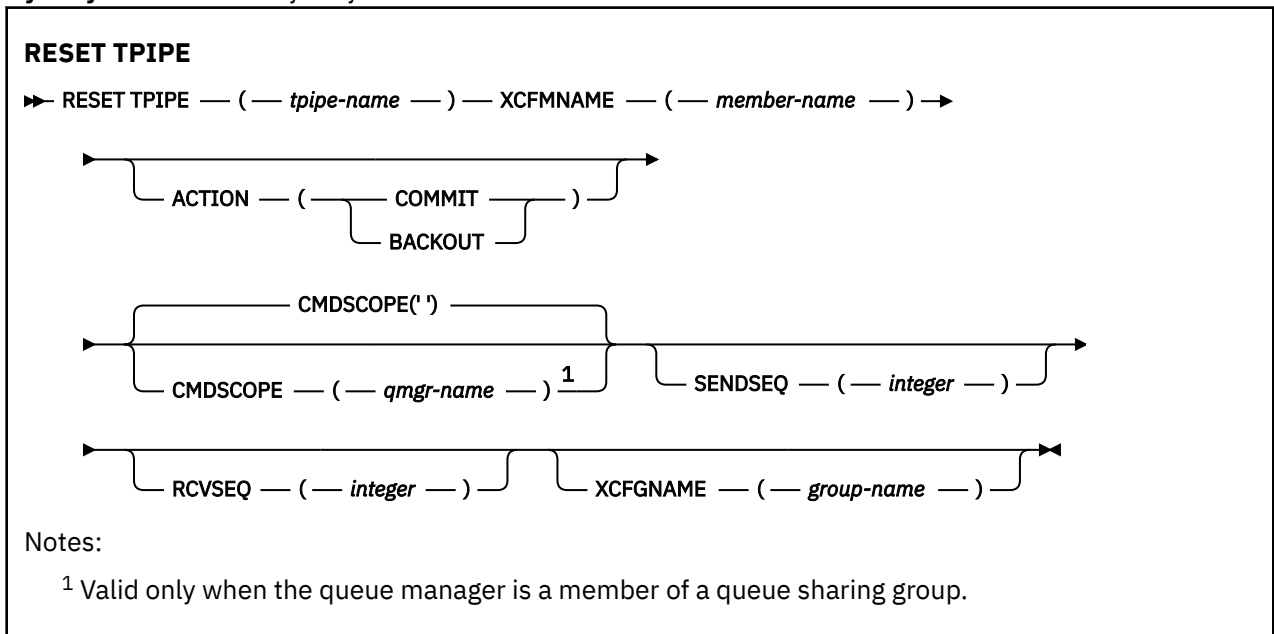
### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 907](#)
- [“Parameter descriptions for RESET TPIPE” on page 907](#)

**Synonym:** There is no synonym for this command.



### Usage notes

1. This command is used in response to the resynchronization error reported in message CSQ2020E, and initiates resynchronization of the Tpipe with IMS.
2. The command fails if the queue manager is not connected to the specified XCF member.
3. The command fails if the queue manager is connected to the specified XCF member, but the Tpipe is open.

### Parameter descriptions for RESET TPIPE

#### ( *tpipe-name* )

The name of the Tpipe to be reset. This is required.

#### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

**ACTION**

Specifies whether to commit or back out any unit of recovery associated with this Tpipe. This is required if there is such a unit of recovery reported in message CSQ2020E; otherwise it is ignored.

**COMMIT**

The messages from IBM MQ are confirmed as having already transferred to IMS ; that is, they are deleted from the IBM MQ - IMS bridge queue.

**BACKOUT**

The messages from IBM MQ are backed out; that is, they are returned to the IBM MQ - IMS bridge queue.

**SENDSEQ( *integer* )**

The new recoverable sequence number to be set in the Tpipe for messages sent by IBM MQ and to be set as the partner's receive sequence number. It must be hexadecimal and can be up to 8 digits long, and can optionally be enclosed by X ' ' . It is optional; if omitted, the sequence number is not changed but the partner's receive sequence is set to the IBM MQ send sequence number.

**RCVSEQ( *integer* )**

The new recoverable sequence number to be set in the Tpipe for messages received by IBM MQ and to be set as the partner's send sequence number. It must be hexadecimal and can be up to 8 digits long, and can optionally be enclosed by X ' ' . It is optional; if omitted, the sequence number is not changed but the partner's send sequence is set to the IBM MQ receive sequence number.

**XCFGNAME( *group-name* )**

The name of the XCF group to which the Tpipe belongs. This is 1 through 8 characters long. It is optional; if omitted, the group name used is that specified in the OTMACON system parameter.

**XCFMNAME( *member-name* )**


The name of the XCF member within the group specified by XCFGNAME to which the Tpipe belongs. This is 1 through 16 characters long, and is required.

**RESOLVE CHANNEL (ask a channel to resolve in-doubt messages)**

Use the MQSC command **RESOLVE CHANNEL** to request a channel to commit or back out in-doubt messages.

**Using MQSC commands**

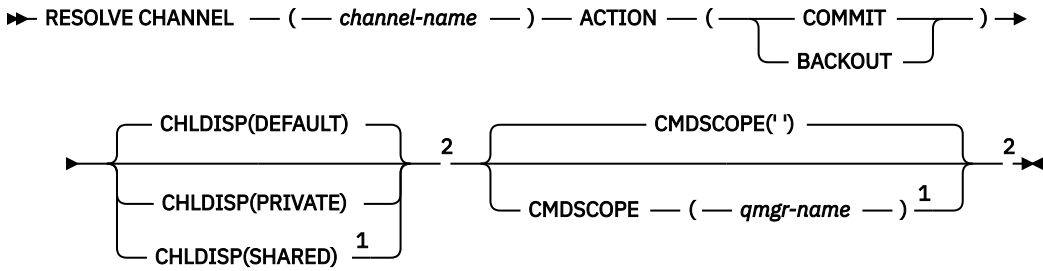
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for RESOLVE CHANNEL” on page 909](#)
- [“Parameter descriptions for RESOLVE CHANNEL” on page 909](#)

**Synonym:** RESOLVE CHL (RES CHL on z/OS )

## RESOLVE CHANNEL



### Notes:

- <sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.
- <sup>2</sup> Valid only on z/OS.

## Usage notes for RESOLVE CHANNEL

1. This command is used when the other end of a link fails during the confirmation period, and for some reason it is not possible to reestablish the connection.
2. In this situation the sending end remains in doubt as to whether the messages were received. Any outstanding units of work must be resolved by being backed out or committed.
3. If the resolution specified is not the same as the resolution at the receiving end, messages can be lost or duplicated.
4. **z/OS** On z/OS, the command server and the channel initiator must be running.
5. This command can be used only for sender (SDR), server (SVR), and cluster-sender (CLUSDR) channels (including those that have been defined automatically).
6. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager's repository.

## Parameter descriptions for RESOLVE CHANNEL

### (channel-name)

The name of the channel for which in-doubt messages are to be resolved. This is required.

### ACTION

Specifies whether to commit or back out the in-doubt messages (this is required):

#### COMMIT

The messages are committed, that is, they are deleted from the transmission queue

#### BACKOUT

The messages are backed out, that is, they are restored to the transmission queue

### **z/OS** CHLDISP

This parameter applies to z/OS only and can take the values of:

- DEFAULT
- PRIVATE
- SHARED

If this parameter is omitted, then the DEFAULT value applies. This is taken from the default channel disposition attribute, **DEFCDISP**, of the channel object.

In conjunction with the various values of the **CMDSCOPE** parameter, this parameter controls two types of channel:

**SHARED**

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of SHARED.

**PRIVATE**

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than SHARED.

**Note:** This disposition is **not** related to the disposition set by the disposition of the queue sharing group of the channel definition.

The combination of the **CHLDISP** and **CMDSCOPE** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.

The various combinations of **CHLDISP** and **CMDSCOPE** are summarized in the following table:

<i>Table 179. CHLDISP and CMDSCOPE for RESOLVE CHANNEL</i>		
<b>CHLDISP</b>	<b>CMDSCOPE( ) or CMDSCOPE (local-qmgr)</b>	<b>CMDSCOPE (qmgr-name)</b>
PRIVATE	Resolve private channel on the local queue manager	Resolve private channel on the named queue manager
SHARED	Resolve a shared channel on all active queue managers.  This might automatically generate a command using <b>CMDSCOPE</b> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.  The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.	Not permitted

**z/OS CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

If **CHLDISP** is set to SHARED, **CMDSCOPE** must be blank or the local queue manager.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.



**ACTION**

Specifies whether to commit or back out the in-doubt threads:

**COMMIT**

Commits the threads

**BACKOUT**

Backs out the threads

**CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. This is the default value.

***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

**NID**

Origin identifier. Specifies the thread or threads to be resolved.

**( *origin-id* )**

This is as returned by the DISPLAY CONN command, and is of the form *origin-node.origin-urid*, where:

- *origin-node* identifies the originator of the thread, except RRSBATCH where it is omitted.
- *origin-urid* is the hexadecimal number assigned to the unit of recovery by the originating system for the specific thread to be resolved.

When *origin-node* is present there must be a period (.) between it and *origin-urid*.

You can specify multiple identifiers separated by a commas to resolve more than one thread.

**(\*)**

Resolves all threads associated with the connection.

**QMNAME**

Specifies that if the designated queue manager is INACTIVE, IBM MQ should search information held in the coupling facility about units of work, performed by the indicated queue manager, that match the connection name and origin identifier.

Matching units of work are either committed or backed out according to the ACTION specified.

Only the shared portion of the unit of work are resolved by this command.

As the queue manager is necessarily inactive, local messages are unaffected and remain locked until the queue manager restarts, or after restarting, connects with the transaction manager.

Examples:

```
RESOLVE INDOUBT(CICSA) ACTION(COMMIT) NID(CICSA.ABCDEF0123456789)
RESOLVE INDOUBT(CICSA) ACTION(BACKOUT) NID(*)
```




## RESUME QMGR (resume a cluster queue manager)

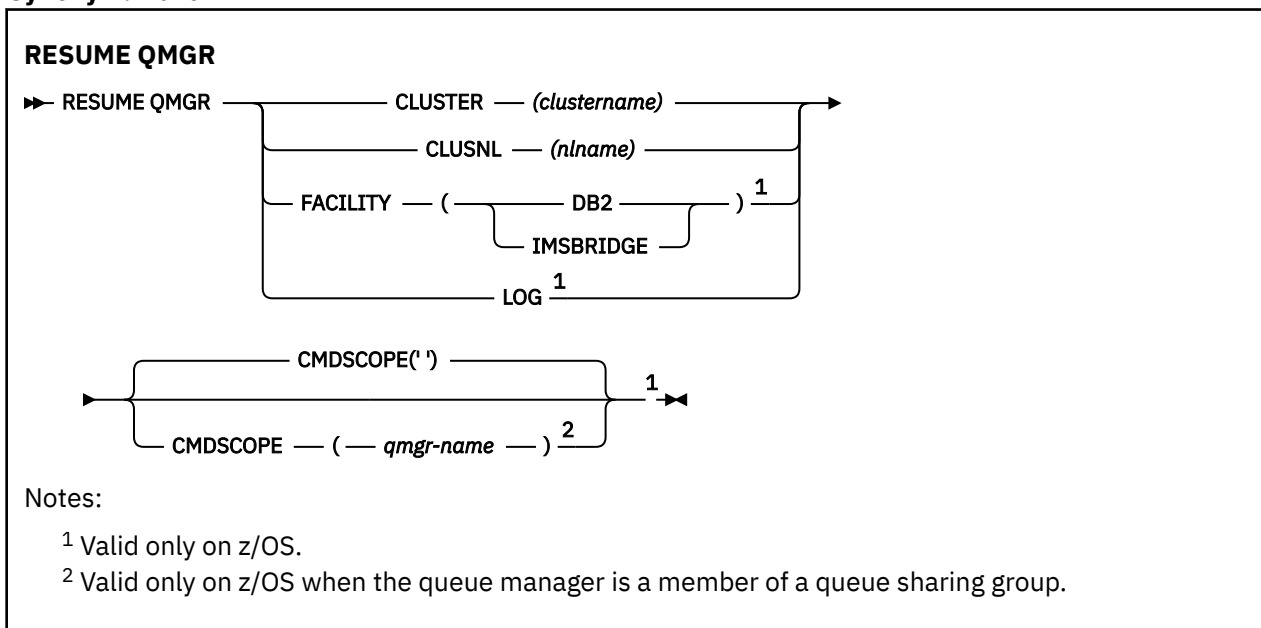
Use the MQSC command RESUME QMGR to inform other queue managers in a cluster that the local queue manager is available again for processing and can be sent messages. It reverses the action of the SUSPEND QMGR command.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
-  See “Using RESUME QMGR on z/OS” on page 913
- [“Usage notes” on page 913](#)
- [“Parameter descriptions for RESUME QMGR” on page 914](#)

**Synonym:** None



### Using RESUME QMGR on z/OS



RESUME QMGR can be used on z/OS. Depending on the parameters used on the command, it may be issued from various sources. For an explanation of the symbols in this table, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Command	Command Sources	Notes
RESUME QMGR CLUSTER/CLUSNL	CR	Ensure the channel initiator is running
RESUME QMGR FACILITY	CR	
RESUME QMGR LOG	C	

### Usage notes

1.   The command is valid only on AIX and Linux.

2. **z/OS** On z/OS, if you define CLUSTER or CLUSNL:
  - a. The command fails if the channel initiator has not been started.
  - b. Any errors are reported to the console on the system where the channel initiator is running; they are not reported to the system that issued the command.
3. **z/OS** On z/OS, you cannot issue RESUME QMGR CLUSTER (*clustername*) or RESUME QMGR FACILITY commands from CSQINP2.
4. **z/OS** This command, with the CLUSTER and CLUSNL parameters, is **not** available on the reduced function form of IBM MQ for z/OS supplied with WebSphere Application Server.
5. **z/OS** On z/OS, the SUSPEND QMGR and RESUME QMGR commands are supported through the console only. However, all the other SUSPEND and RESUME commands are supported through the console and command server.

## Parameter descriptions for RESUME QMGR

### **CLUSTER** (*clustername*)

The name of the cluster for which availability is to be resumed.

### **CLUSNL** (*nlname*)

The name of the namelist specifying a list of clusters for which availability is to be resumed.

### **FACILITY**

Specifies the facility to which connection is to be re-established.

#### **Db2**

Re-establishes connection to Db2.

#### **IMSBRIDGE**

Resumes normal IMS bridge activity.

This parameter is only valid on z/OS.

### **LOG**

Resumes logging and update activity for the queue manager that was suspended by a previous SUSPEND QMGR command. Valid on z/OS only. If LOG is specified, the command can be issued only from the z/OS console.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

## **z/OS RVERIFY SECURITY (set a user reverification flag) on z/OS**

Use the MQSC command RVERIFY SECURITY to set a reverification flag for all specified users. The user is reverified the next time that security is checked for that user.

## Using MQSC commands on z/OS

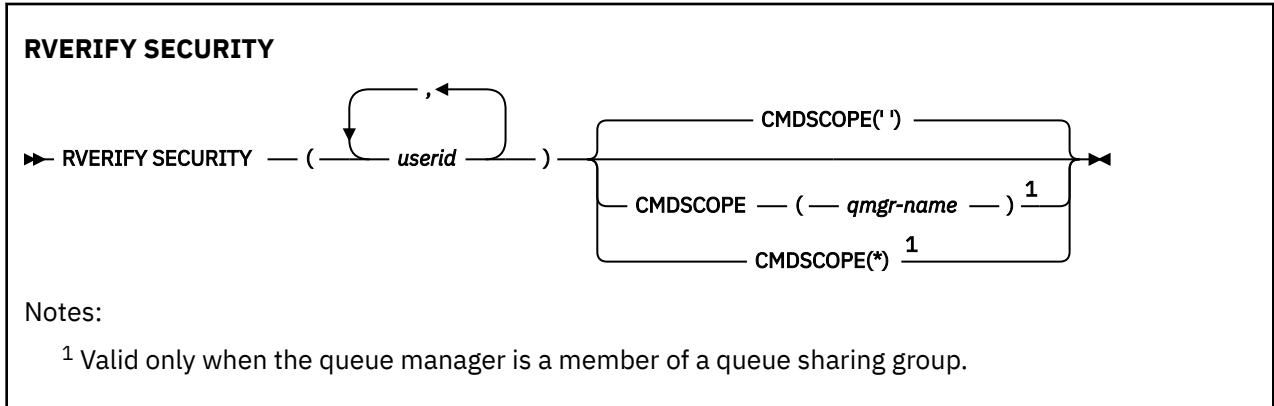
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for RVERIFY SECURITY” on page 915](#)

**Synonym:** REV SEC

REVERIFY SECURITY is another synonym for RVERIFY SECURITY



## Parameter descriptions for RVERIFY SECURITY

### *(userid...)*

You must specify one or more user IDs. Each user ID specified is signed off and signed back on again the next time that a request is issued on behalf of that user that requires security checking.

### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## **SET ARCHIVE (change archive system settings) on z/OS**

Use the MQSC command SET ARCHIVE to dynamically change certain archive system parameter values initially set by your system parameter module at queue manager startup.

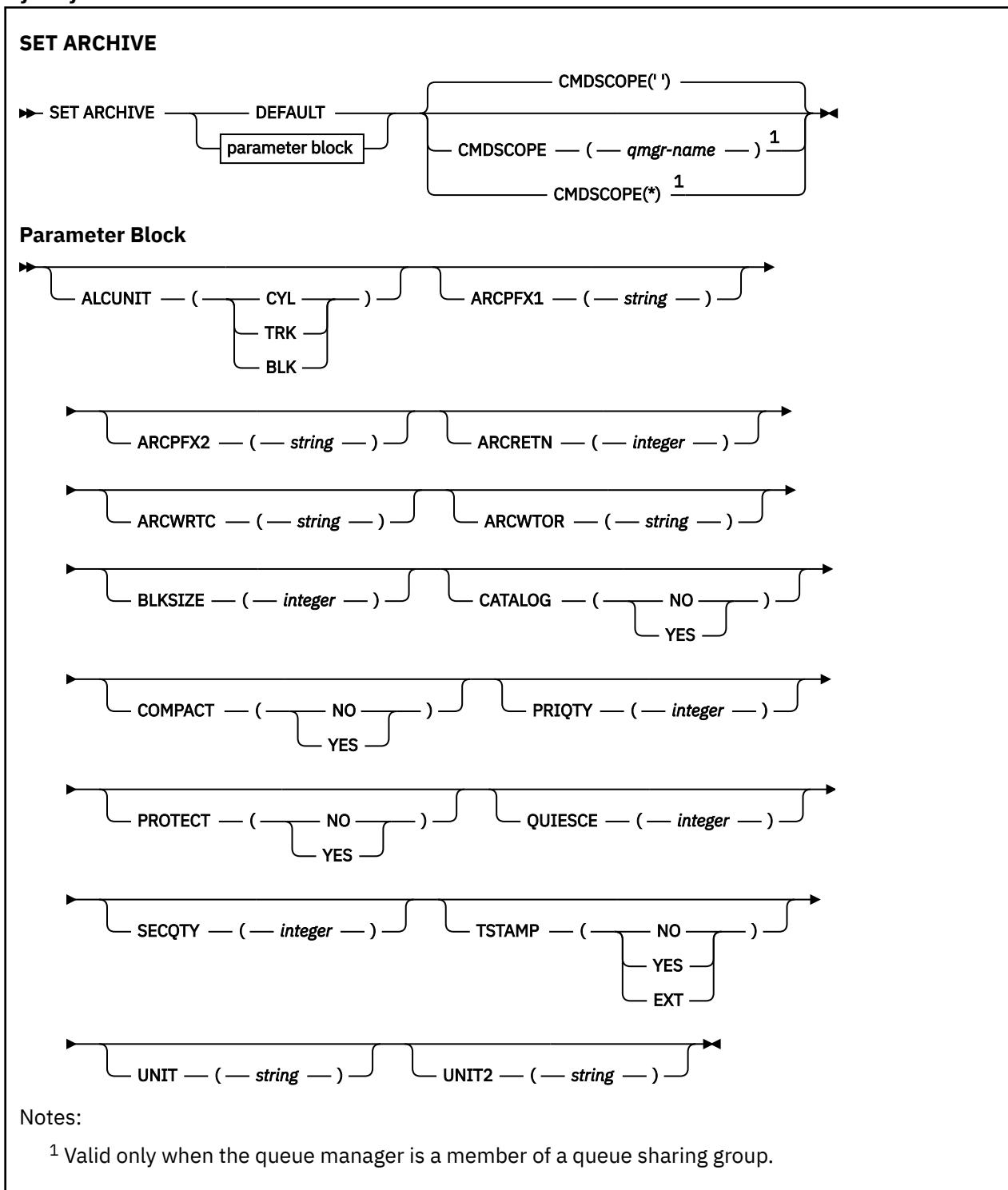
### **Using MQSC commands on z/OS**

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for SET ARCHIVE” on page 916](#)
- [“Parameter descriptions for SET ARCHIVE” on page 917](#)
- [“Parameter block” on page 917](#)

**Synonym:** SET ARC



### Usage notes for SET ARCHIVE

1. The new values will be used at the next archive log offload.

2. The queue manager picks up the values in ZPARM, so the **SET ARCHIVE** values you used in the previous cycle are lost.

To permanently change the values, either change the CSQ6SYSP parameters and regenerate the parameter module, or put the **SET ARCHIVE** commands into a data set in the CSQINP2 concatenation.

## Parameter descriptions for SET ARCHIVE

### CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

''

The command runs on the queue manager on which it was entered. This is the default value.

#### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which it was entered, only if you are using a queue sharing group environment and if the command server is enabled.

You cannot use CMDSCOPE(*qmgr-name*) for commands issued from the first initialization input data set, CSQINP1.

\*


The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE(\*) for commands issued from CSQINP1.

### DEFAULT

Resets all the archive system parameters to the values set at queue manager startup.

## Parameter block

 For a full description of these parameters, see [Using CSQ6ARVP](#).

Parameter block is any one or more of the following parameters that you want to change:

### ALCUNIT

Specifies the unit in which primary and secondary space allocations are made.

Specify one of:

#### **CYL**

Cylinders

#### **TRK**

Tracks

#### **BLK**

Blocks

### ARCPFX1

Specifies the prefix for the first archive log data set name.

See the TSTAMP parameter for a description of how the data sets are named and for restrictions on the length of ARCPFX1.

### ARCPFX2


Specifies the prefix for the second archive log data set name.

See the `TSTAMP` parameter for a description of how the data sets are named and for restrictions on the length of `ARCPFX2`.

#### **ARCRETN**

Specifies the retention period, in days, to be used when the archive log data set is created.

The parameter must be in the range zero - 9999.

 For more information about discarding archive log data sets, see [Discarding archive log data sets](#).

#### **ARCWRTC**

Specifies the list of z/OS routing codes for messages about the archive log data sets to the operator.

Specify up to 14 routing codes, each with a value in the range 1 through 16. You must specify at least one code. Separate codes in the list by commas, not by blanks.

For more information about z/OS routing codes, see *Routing codes* in [Message description](#) in one of the volumes of the *z/OS MVS System Messages* manuals.

#### **ARCWTOR**

Specifies whether a message is to be sent to the operator and a reply received before attempting to mount an archive log data set.

Other IBM MQ users might be forced to wait until the data set is mounted, but they are not affected while IBM MQ is waiting for the reply to the message.

Specify either:

##### **YES**

The device needs a long time to mount archive log data sets. For example, a tape drive. (The synonym is **Y**.)

##### **NO**

The device does not have long delays. For example, DASD. (The synonym is **N**.)

#### **BLKSIZE**

Specifies the block size of the archive log data set. The block size you specify must be compatible with the device type you specify in the `UNIT` parameter.

The parameter must be in the range 4 097 through 28 672. The value you specify is rounded up to a multiple of 4 096.

This parameter is ignored for data sets that are managed by the storage management subsystem (SMS).

#### **CATALOG**

Specifies whether archive log data sets are cataloged in the primary integrated catalog facility (ICF) catalog.

Specify either:

##### **NO**

Archive log data sets are not cataloged. (The synonym is **N**.)

##### **YES**

Archive log data sets are cataloged. (The synonym is **Y**.)

#### **COMPACT**

Specifies whether data written to archive logs is to be compacted. This option applies only to a 3480 or 3490 device that has the improved data recording capability (IDRC) feature. When this feature is turned on, hardware in the tape control unit writes data at a much higher density than normal, allowing for more data on each volume. Specify **NO** if you do not use a 3480 device with the IDRC feature or a 3490 base model, with the exception of the 3490E. Specify **YES** if you want the data to be compacted.

Specify either:

**NO**

Do not compact the data sets. (The synonym is **N**.)

**YES**

Compact the data sets. (The synonym is **Y**.)

**PRIQTY**

Specifies the primary space allocation for DASD data sets in ALCUNITs.

The value must be greater than zero.

This value must be sufficient for a copy of either the log data set or its corresponding BSDS, whichever is the larger.

**PROTECT**

Specifies whether archive log data sets are to be protected by discrete ESM (external security manager) profiles when the data sets are created.

Specify either:

**NO**

Profiles are not created. (The synonym is **N**.)

**YES**

Discrete data set profiles are created when logs are offloaded. (The synonym is **Y**.) If you specify YES:

- ESM protection must be active for IBM MQ.
- The user ID associated with the IBM MQ address space must have authority to create these profiles.
- The TAPEVOL class must be active if you are archiving to tape.

Otherwise, offloads will fail.

**QUIESCE**

Specifies the maximum time in seconds allowed for the quiesce when an ARCHIVE LOG command is issued with MODE QUIESCE specified.

The parameter must be in the range 1 through 999.

**SECQTY**

Specifies the secondary space allocation for DASD data sets in ALCUNITs.

The parameter must be greater than zero.

**TSTAMP**

Specifies whether the archive log data set name has a time stamp in it.

Specify either:

**NO**

Names do not include a time stamp. (The synonym is **N**.) The archive log data sets are named:

```
arcpxi.A nnnnnnn
```

Where *arcpxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpxi* can have up to 35 characters.

**YES**

Names include a time stamp. (The synonym is **Y**.) The archive log data sets are named:

```
arcpxi.cyyddd.T hhmsst.A nnnnnnn
```

where *c* is 'D' for the years up to and including 1999 or 'E' for the year 2000 and later, and *arcpxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpxi* can have up to 19 characters.

**EXT**

Names include a time stamp. The archive log data sets are named:

```
arcpfxi.D yyyydd.T hhmsst.A nnnnnn
```

Where *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 17 characters.

**UNIT**

Specifies the device type or unit name of the device that is used to store the first copy of the archive log data set.

Specify a device type or unit name of 1 through 8 characters.

If you archive to DASD, you can specify a generic device type with a limited volume range.

**UNIT2**

Specifies the device type or unit name of the device that is used to store the second copy of the archive log data sets.

Specify a device type or unit name of 1 through 8 characters.

If this parameter is blank, the value set for the UNIT parameter is used.

**Multi****SET AUTHREC (set authority records) on Multiplatforms**

Use the MQSC command SET AUTHREC to set authority records associated with a profile name.

**Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions” on page 922](#)
- [Usage notes for SET AUTHREC](#)

See [“setmqaut \(grant or revoke authority\)” on page 193](#) for more information on the options that you can select.



# SET AUTHREC

→ SET AUTHREC (— PROFILE — (— *profile-name* —) →)

→ OBJTYPE — (— AUTHINFO —) →

- CHANNEL
- CLNTCONN
- COMMINFO
- LISTENER
- NAMELIST
- PROCESS
- QUEUE
- QMGR
- RQMNAME
- SERVICE
- TOPIC

→ PRINCIPAL — (— *principal-name* —) →

→ GROUP — (— *group-name* —) →

→ AUTHADD — (— NONE —) →

- ALTUSR
- BROWSE
- CHG
- CLR
- CONNECT
- CRT
- DLT
- DSP
- GET
- INQ
- PUT
- PASSALL
- PASSID
- SET
- SETALL
- SETID
- SUB
- RESUME
- PUB
- SYSTEM
- CTRL
- CTRLX
- ALL
- ALLADM
- ALLMQI

→ AUTHRMV — (— NONE —) →

- ALTUSR
- BROWSE
- CHG
- CLR
- CONNECT
- CRT
- DLT
- DSP
- GET
- INQ
- PUT
- PASSALL
- PASSID
- SET
- SETALL
- SETID
- SUB
- RESUME
- PUB
- SYSTEM
- CTRL
- CTRLX
- ALL
- ALLADM
- ALLMQI

→ SERVCOMP — (— *service-component* —) →

## Parameter descriptions

### **PROFILE**(*profile-name*)

The name of the object or generic profile for which to display the authority records. This parameter is required unless the **OBJTYPE** parameter is QMGR, in which case it can be omitted.

See [Using OAM generic profiles on AIX, Linux, and Windows](#) for more information on generic profiles and wildcard characters.

### **OBJTYPE**

The type of object referred to by the profile. Specify one of the following values:

#### **AUTHINFO**

Authentication information record

#### **CHANNEL**

Channel

#### **CLNTCONN**

Client connection channel

#### **COMMINFO**

Communication information object

#### **LISTENER**

Listener

#### **NAMELIST**

Namelist

#### **PROCESS**

Process

#### **QUEUE**

Queue

#### **QMGR**

Queue manager

#### **RQMNAME**

Remote queue manager

#### **SERVICE**

Service

#### **TOPIC**

Topic

### **PRINCIPAL**(*principal-name*)

A principal name. This is the name of a user for whom to set authority records for the specified profile. On IBM MQ for Windows, the name of the principal can optionally include a domain name, specified in this format: `user@domain`.

You must specify either **PRINCIPAL** or **GROUP**.

### **GROUP**(*group-name*)

A group name. This is the name of the user group for which to set authority records for the specified profile. You can specify one name only and it must be the name of an existing user group.

**Windows** For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following format:

```
GroupName@domain
```

You must specify either **PRINCIPAL** or **GROUP**.

### **AUTHADD**

A list of authorizations to add in the authority records. Specify any combination of the following values:

**NONE**

No authorization

**ALTUSR**

Specify an alternative user ID on an MQI call

**BROWSE**

Retrieve a message from a queue by issuing an **MQGET** call with the BROWSE option

**CHG**

Change the attributes of the specified object, using the appropriate command set

**CLR**

Clear a queue or a topic

**CONNECT**

Connect an application to a queue manager by issuing an **MQCONN** call

**CRT**

Create objects of the specified type using the appropriate command set

**DLT**

Delete the specified object using the appropriate command set

**DSP**

Display the attributes of the specified object using the appropriate command set

**GET**

Retrieve a message from a queue by issuing an **MQGET** call

**INQ**

Make an inquiry on a specific queue by issuing an **MQINQ** call

**PUT**

Put a message on a specific queue by issuing an **MQPUT** call

**PASSALL**

Pass all context

**PASSID**

Pass the identity context

**SET**

Set attributes on a queue by issuing an **MQSET** call

**SETALL**

Set all context on a queue

**SETID**

Set the identity context on a queue

**SUB**

Create, alter, or resume a subscription to a topic using the **MQSUB** call

**RESUME**

Resume a subscription using the **MQSUB** call

**PUB**

Publish a message on a topic using the **MQPUT** call

**SYSTEM**

Give authority to principals or groups, who are authorized to carry out privileged operations on the queue manager, for internal system operations.

**CTRL**

Start and stop the specified channel, listener, or service, and ping the specified channel

**CTRLX**

Reset or resolve the specified channel

**ALL**

Use all operations relevant to the object

all authority is equivalent to the union of the authorities alladm, allmqi, and system appropriate to the object type.

**ALLADM**

Perform all administration operations relevant to the object

**ALLMQI**

Use all MQI calls relevant to the object

**AUTHRMV**

A list of authorizations to remove from the authority records. Specify any combination of the following values:

**NONE**

No authorization

**ALTUSR**

Specify an alternative user ID on an MQI call

**BROWSE**

Retrieve a message from a queue by issuing an **MQGET** call with the BROWSE option

**CHG**

Change the attributes of the specified object, using the appropriate command set

**CLR**

Clear a queue or a topic

**CONNECT**

Connect an application to a queue manager by issuing an **MQCONN** call

**CRT**

Create objects of the specified type using the appropriate command set

**DLT**

Delete the specified object using the appropriate command set

**DSP**

Display the attributes of the specified object using the appropriate command set

**GET**

Retrieve a message from a queue by issuing an **MQGET** call

**INQ**

Make an inquiry on a specific queue by issuing an **MQINQ** call

**PUT**

Put a message on a specific queue by issuing an **MQPUT** call

**PASSALL**

Pass all context

**PASSID**

Pass the identity context

**SET**

Set attributes on a queue by issuing an **MQSET** call

**SETALL**

Set all context on a queue

**SETID**

Set the identity context on a queue

**SUB**

Create, alter, or resume a subscription to a topic using the **MQSUB** call

**RESUME**

Resume a subscription using the **MQSUB** call

**PUB**

Publish a message on a topic using the **MQPUT** call

**SYSTEM**

Use queue manager for internal system operations

**CTRL**

Start and stop the specified channel, listener, or service, and ping the specified channel

**CTRLX**

Reset or resolve the specified channel

**ALL**

Use all operations relevant to the object

all authority is equivalent to the union of the authorities alladm, allmqi, and system appropriate to the object type.

**ALLADM**

Perform all administration operations relevant to the object

**ALLMQI**

Use all MQI calls relevant to the object

**Note:** To use SETID or SETALL authority, authorizations must be granted on both the appropriate queue object and also on the queue manager object.

**SERVCOMP(service-component)**

The name of the authorization service for which information is to be set.

If you specify this parameter, it specifies the name of the authorization service to which the authorizations apply. If you omit this parameter, the authority record is set using the registered authorization services in turn in accordance with the rules for chaining authorization services.

**Usage notes for SET AUTHREC**

The list of authorizations to add and the list of authorizations to remove must not overlap. For example, you cannot add display authority and remove display authority with the same command. This rule applies even if the authorities are expressed using different options. For example, the following command fails because DSP authority overlaps with ALLADM authority:

```
SET AUTHREC PROFILE(*) OBJTYPE(Queue) PRINCIPAL(PRINC01) AUTHADD(DSP) AUTHRMV(ALLADM)
```

The exception to this overlap behavior is with the ALL authority. The following command first adds ALL authorities then removes the SETID authority:

```
SET AUTHREC PROFILE(*) OBJTYPE(Queue) PRINCIPAL(PRINC01) AUTHADD(ALL) AUTHRMV(SETID)
```

The following command first removes ALL authorities then adds the DSP authority:

```
SET AUTHREC PROFILE(*) OBJTYPE(Queue) PRINCIPAL(PRINC01) AUTHADD(DSP) AUTHRMV(ALL)
```

Regardless of the order in which they are provided on the command, the ALL are processed first.

**Related concepts**

[OAM user-based permissions on AIX and Linux](#)

**Related reference**

[“dmpmqaut \(dump MQ authorizations\)” on page 55](#)

Dump a list of current authorizations for a range of IBM MQ object types and profiles.

[“setmqaut \(grant or revoke authority\)” on page 193](#)

Change the authorizations to a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

[“DISPLAY AUTHREC \(display authority records\) on Multiplatforms” on page 646](#)

Use the MQSC command DISPLAY AUTHREC to display the authority records associated with a profile name.

## SET CHLAUTH (create or modify a channel authentication record)

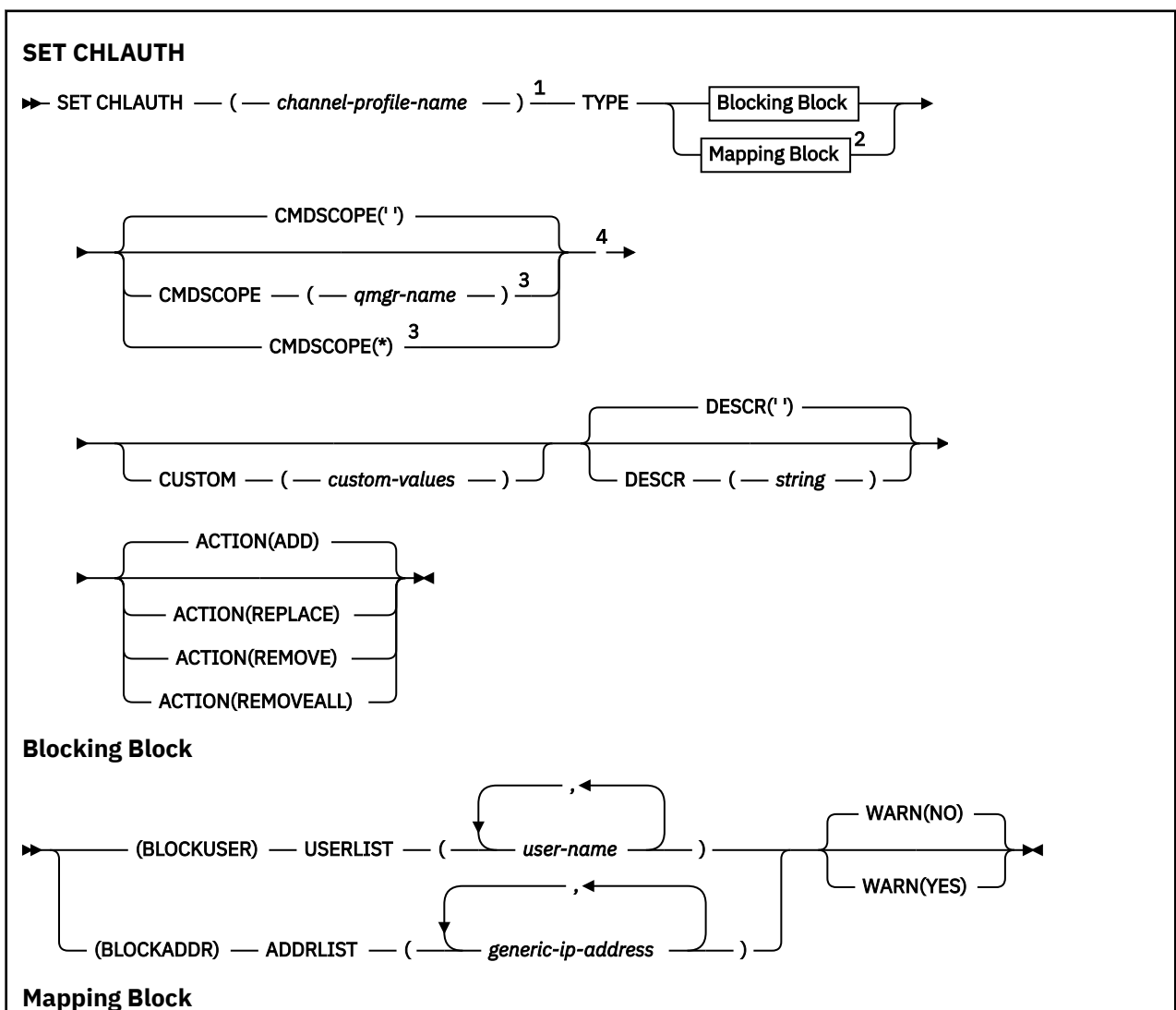
Use the MQSC command **SET CHLAUTH** to create or modify a channel authentication record.

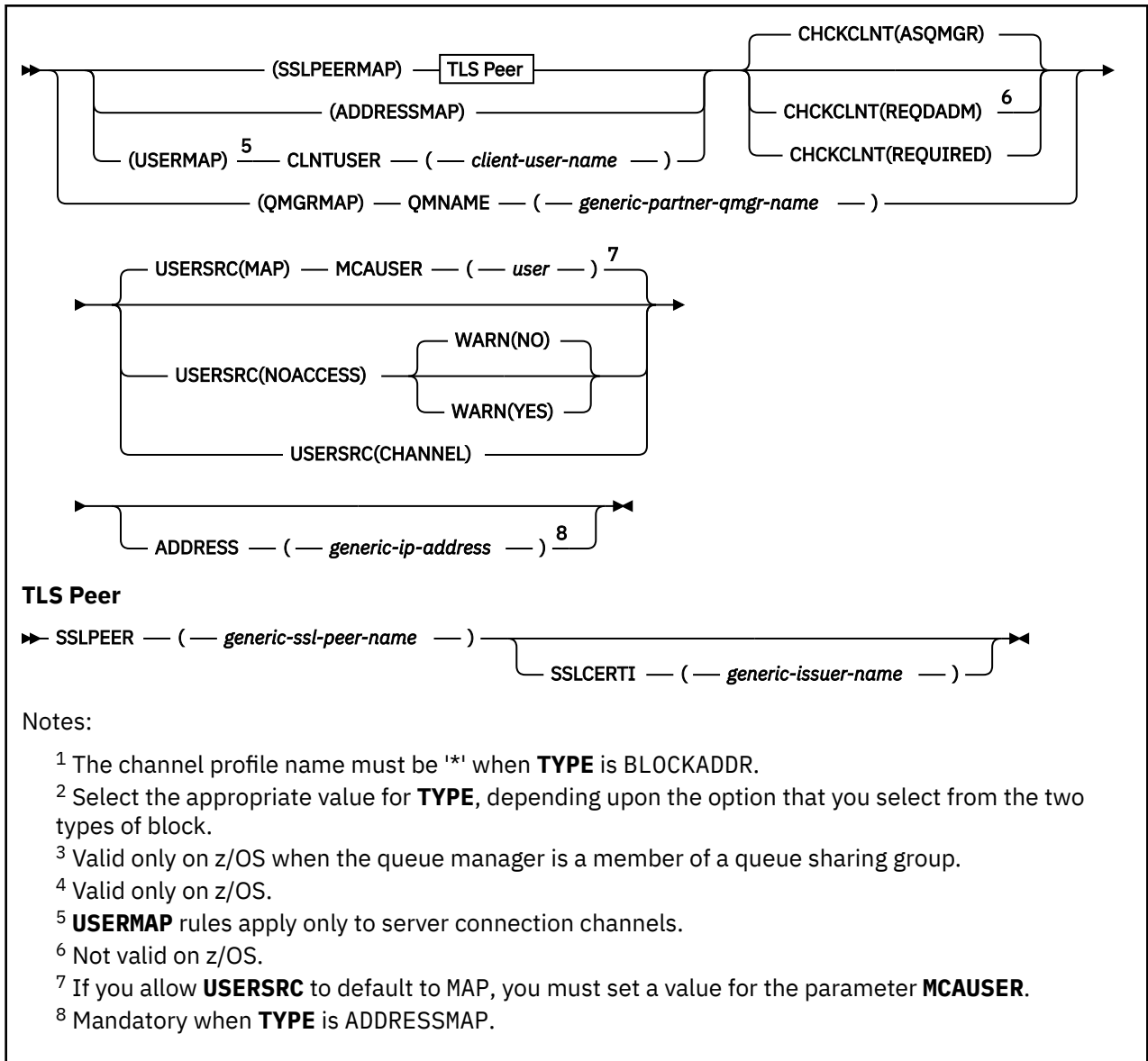
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [Usage notes](#)
- [Parameters](#)





## Usage notes

The following table shows which parameters are valid for each value of **ACTION**:

Parameter	Action		
	ADD or REPLACE	REMOVE	REMOVEALL
CHLAUTH	✓	✓	✓
TYPE	✓	✓	✓
z/OS	✓	✓	✓
z/OS CMDScope			
ACTION	✓	✓	✓
ADDRESS	✓	✓	
ADDRLIST	✓	✓	

Parameter	Action		
	ADD or REPLACE	REMOVE	REMOVEALL
CHCKCLNT	✓		
CLNTUSER	✓	✓	
MCAUSER	✓		
QMNAME	✓	✓	
SSLCERTI	✓	✓	
SSLPEER	✓	✓	
USERLIST	✓	✓	
USERSRC	✓		
WARN	✓		
DESCR	✓		

Note the following:

- **CHLAUTH** rules can be used for any channels
- **USERMAP** rules are valid, only for server connection channels.
- Changes, such as mapping the **MCAUSER** of the channel, take effect only when starting a channel.

Therefore, if a channel is already running, that channel must be stopped, and restarted, for the **CHLAUTH** rule changes to take effect.

## Parameters

### *channel-profile-name*

The name of the channel or set of channels for which you are setting channel authentication configuration. You can use one or more asterisks (\*), in any position, as wildcards to specify a set of channels. If you set **TYPE** to **BLOCKADDR**, you must set the generic channel name to a single asterisk, which matches all channel names. On z/OS the generic-channel-name must be in quotes if it contains an asterisk.

### **TYPE**

The **TYPE** parameter must follow the **channel-profile-name** parameter.

The type of channel authentication record for which to set allowed partner details or mappings to **MCAUSER**. This parameter is required. The following values can be used:

#### **BLOCKUSER**

This channel authentication record prevents a specified user or users from connecting. **BLOCKUSER** must be accompanied by a **USERLIST** parameter.

#### **BLOCKADDR**

This channel authentication record prevents connections from a specified IP address or addresses. **BLOCKADDR** must be accompanied by an **ADDRLIST** parameter. **BLOCKADDR** operates at the listener before the channel name is known.

#### **SSLPEERMAP**

This channel authentication record maps TLS Distinguished Names (DNs) to **MCAUSER** values. **SSLPEERMAP** must be accompanied by an **SSLPEER** parameter.



## ADDRESSMAP

This channel authentication record maps IP addresses to **MCAUSER** values. ADDRESSMAP must be accompanied by an **ADDRESS** parameter. ADDRESSMAP operates at the channel.

## USERMAP

This channel authentication record maps asserted user IDs to **MCAUSER** values. **USERMAP** must be accompanied by a **CLNTUSER** parameter.

## QMGRMAP

This channel authentication record maps remote queue manager names to **MCAUSER** values. QMGRMAP must be accompanied by a **QMNAME** parameter.

## ACTION

The action to perform on the channel authentication record. The following values are valid:

### ADD

Add the specified configuration to a channel authentication record. This is the default value.

For **TYPE** parameter options SSLPEERMAP, ADDRESSMAP, USERMAP and QMGRMAP, if the specified configuration exists, the command fails.

For **TYPE** parameter options BLOCKUSER and BLOCKADDR, the configuration is added to the list.

### REPLACE

Replace the current configuration of a channel authentication record.

For **TYPE** parameter options SSLPEERMAP, ADDRESSMAP, USERMAP and QMGRMAP, if the specified configuration exists, it is replaced with the new configuration. If it does not exist it is added.

For **TYPE** parameter options BLOCKUSER and BLOCKADDR, the configuration specified replaces the current list, even if the current list is empty. If you replace the current list with an empty list, this acts like REMOVEALL.

### REMOVE

Remove the specified configuration from the channel authentication records. Note, that if the configuration does not exist the command still works. If you remove the last entry from a list, this acts like REMOVEALL.

### REMOVEALL

Remove all members of the list and thus the whole record (for BLOCKADDR and BLOCKUSER) or all previously defined mappings (for ADDRESSMAP, SSLPEERMAP, QMGRMAP and USERMAP) from the channel authentication records. This option cannot be combined with specific values supplied in **ADDRLIST**, **USERLIST**, **ADDRESS**, **SSLPEER**, **QMNAME** or **CLNTUSER**. If the specified **TYPE** has no current configuration, the command still succeeds.

## ADDRESS

The filter to be used to compare with the IP address or host name of the partner queue manager or client at the other end of the channel. Channel authentication records containing hostnames are only checked if the queue manager is configured to look them up with **REVDNS (ENABLED)**. Details of the values that are allowed as host names are defined in the IETF documents [RFC 952](#) and [RFC 1123](#). Hostname matching is not case sensitive.

This parameter is mandatory with **TYPE (ADDRESSMAP)**

This parameter is also valid when **TYPE** is SSLPEERMAP, USERMAP, or QMGRMAP and **ACTION** is ADD, REPLACE, or REMOVE. You can define more than one channel authentication object with the same main identity, for example the same TLS peer name, with different addresses. However, you cannot define channel authentication records with overlapping address ranges for the same main identity. See [“Generic IP addresses for channel authentication records”](#) on page 933 for more information about filtering IP addresses.

If the address is generic then it must be in quotes.

## ADDRLIST

A list of up to 256 generic IP addresses which are banned from accessing this queue manager on any channel. This parameter is only valid with **TYPE (BLOCKADDR)**. For more information about filtering IP addresses, see [“Generic IP addresses for channel authentication records”](#) on page 933.

If the address is generic then it must be in quotes.

## CHKCLNT

Specifies whether the connection that matches this rule and is being allowed with **USERSRC (CHANNEL)** or **USERSRC (MAP)**, must also specify a valid authentication credentials in the MQCSP structure. If a password is supplied, it cannot contain single quotation marks (').

## REQDADM


Valid authentication credentials are required for the connection to be allowed if it is using a privileged user ID.


A privileged user is one that has full administrative authorities for IBM MQ. For more information, see [Privileged users](#).

Any connections using a non-privileged user ID are not required to provide authentication credentials.

The credentials are checked using the user repository details provided in the authentication information object that is referenced by the queue manager's **CONNAUTH** attribute. If no user repository details are provided, so that connection authentication is not enabled on the queue manager, the connection is not successful.

If an application provides a user ID and password, these credentials are authenticated by the queue manager against the password store indicated by the **AUTHTYPE** attribute in the authentication information object. The connection is only allowed to continue if the user ID and password are valid.

 If an application provides an authentication token, and the queue manager is configured to accept authentication tokens, the token is validated using the configuration specified in the AuthToken stanza of the `qm.ini` file. The connection is only allowed to continue if the token is issued by a trusted issuer.


 This option is not valid on z/OS platforms.

## REQUIRED

Valid authentication credentials are required for the connection to be allowed.

The credentials are checked using the user repository details provided in the authentication information object that is referenced by the queue manager's **CONNAUTH** attribute. If no user repository details are provided, so that connection authentication is not enabled on the queue manager, the connection is not successful.

If an application provides a user ID and password, these credentials are authenticated by the queue manager against the password store indicated by the **AUTHTYPE** in the authentication information object. The connection is only allowed to continue if the user ID and password are valid. The password cannot contain single quotation marks (').

 If an application provides an authentication token, and the queue manager is configured to accept authentication tokens, the token is validated using the configuration specified in the AuthToken stanza of the `qm.ini` file. The connection is only allowed to continue if the token is issued by a trusted issuer.

If an application does not provide any authentication credentials, the connection is rejected.

## ASQMGR

In order for the connection to be allowed, it must meet the connection authentication requirements defined on the queue manager.

If the queue manager's **CONNAUTH** attribute specifies an authentication information object, and the value of **CHCKCLNT** in the authentication information object is **REQUIRED**, the connection fails unless valid authentication credentials are supplied. If the queue manager's **CONNAUTH** attribute does not specify an authentication information object, or the value of **CHCKCLNT** in the authentication information object is not **REQUIRED**, the authentication credentials are not required.



**Attention:** If you select **REQUIRED**, or **REQDADM** on **Multiplatforms**, and you have not set the **CONNAUTH** attribute on the queue manager, or if the value of **CHCKCLNT** is **NONE** in the **AUTHINFO** object that is referenced by the queue manager's **CONNAUTH** attribute, the connection fails. On **Multiplatforms**, you receive message **AMQ9793**. On **z/OS**, you receive message **CSQX793E**.

This parameter is valid only with **TYPE (USERMAP)**, **TYPE (ADDRESSMAP)**, and **TYPE (SSLPEERMAP)** and only when **USERSRC** is not set to **NOACCESS**. It only applies to inbound connections which are **SVRCONN** channels.

Example rules that use this attribute:

- Anything in the defined network can use an asserted user ID if valid authentication credentials are supplied:

```
SET CHLAUTH('*.*.SVRCONN') +
  TYPE(ADDRESSMAP) ADDRESS('192.0.2.*') +
  USERSRC(CHANNEL) CHCKCLNT(REQUIRED)
```




- This rule ensures that TLS authentication must succeed before processing client connection authentication according to the policy set on the queue manager:

```
SET CHLAUTH('SSL.APP1.SVRCONN') +
  TYPE(SSLPEERMAP) SSLPEER('CN="Steve Smith", L="BankA"') +
  MCAUSER(SSMITH) CHCKCLNT(ASQMGR)
```

## CLNTUSER

The client asserted user ID to be mapped to a new user ID, allowed through unchanged, or blocked.

This can be one of the following user IDs:

- The user ID flowed from the client indicating the user ID the client side process is running under.
- The user ID presented by the client in the MQCSP structure on an MQCONN call.
-    The user ID in the user claim of an authentication token presented by the client using the MQCSP structure on an MQCONN call.

The maximum length of the string is **MQ\_CLIENT\_USER\_ID\_LENGTH**.

## **CMDSCOPE**

This parameter applies to **z/OS** only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### **qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect is the same as entering the command on every queue manager in the queue sharing group.

### **CUSTOM**

Reserved for future use.

### **DESCR**

Provides descriptive information about the channel authentication record, which is displayed when you issue the DISPLAY CHLAUTH command. It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

**Note:** Use characters from the coded character set identifier (CCSID) for this queue manager. Other characters might be translated incorrectly if the information is sent to another queue manager.

### **MCAUSER**

The user identifier to be used when the inbound connection matches the TLS DN, IP address, client asserted user ID or remote queue manager name supplied.

This parameter is mandatory with **USERSRC (MAP)** and is valid when **TYPE** is SSLPEERMAP, ADDRESSMAP, USERMAP, or QMGRMAP.

If you use lowercase user IDs you must enclose them in quotation marks: For example:

```
SET CHLAUTH('SYSTEM.DEF.SVRCONN') TYPE(USERMAP) CLNTUSER('johndoe') +
  USERSRC(MAP) MCAUSER('JOHNDOE1') +
  ADDRESS('::FFFF:9.20.4.136') +
  DESCR('Client from z/Linux machine') +
  ACTION(REPLACE)
```

This allows the lowercase user ID to use channel SYSTEM.DEF.SVRCONN on IP address ::FFFF:9.20.4.136. The MCA user for the connection is JOHNDOE1.

If you display the Channel Status (CHS) of the channel, the output is MCAUSER(JOHNDOE1).

This parameter can be used only when **ACTION** is ADD or REPLACE.

### **QMNAME**

The name of the remote partner queue manager, or pattern that matches a set of queue manager names, to be mapped to a user ID or blocked.

This parameter is valid only with **TYPE (QMGRMAP)**.

If the queue manager name is generic then it must be in quotes.

### **SSLCERTI**

This parameter is additional to the **SSLPEER** parameter.

**SSLCERTI** restricts matches to being within certificates issued by a particular Certificate Authority.

A blank **SSLCERTI** acts like a wildcard, matches any Issuer Distinguished Name.

### **SSLPEER**

The filter to use to compare with the Subject Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel.

The **SSLPEER** filter is specified in the standard form used to specify a Distinguished Name. For more information, see [IBM MQ rules for SSLPEER values](#).

The maximum length of the parameter is 1024 bytes.

### **USERLIST**

A list of up to 100 user IDs which are banned from use of this channel or set of channels. Use the special value \*MQADMIN to mean privileged or administrative users. The definition of this value depends on the operating system, as follows:

- **Windows** On Windows, all members of the mqm group, the Administrators group and SYSTEM.
- **Linux** **AIX** On AIX and Linux, all members of the mqm group.
- **IBM i** On IBM i, the profiles (users) qmqm and qmqmadm and all members of the qmqmadm group, and any user defined with the \*ALLOBJ special setting.
- **z/OS** On z/OS, the user ID that the channel initiator, queue manager and advanced message security address spaces are running under.

For more information about privileged users, see [Privileged users](#).

This parameter is only valid with **TYPE (BLOCKUSER)**.

## USERSRC

The source of the user ID to be used for **MCAUSER** at run time. The following values are valid:

### MAP

Inbound connections that match this mapping use the user ID specified in the **MCAUSER** attribute. This is the default value.

### NOACCESS

Inbound connections that match this mapping have no access to the queue manager and the channel ends immediately.

### CHANNEL

Inbound connections that match this mapping use the flowed user ID or any user defined on the channel object in the **MCAUSER** field.

Note that **WARN** and **USERSRC (CHANNEL)**, or **USERSRC (MAP)** are incompatible. This is because channel access is never blocked in these cases, so there is never a reason to generate a warning.

## WARN

Indicates whether this record operates in warning mode.

### NO

This record does not operate in warning mode. Any inbound connection that matches this record is blocked. This is the default value.

### YES

This record operates in warning mode. Any inbound connection that matches this record and would therefore be blocked is allowed access. If channel events are configured, a channel event message is created showing the details of what would have been blocked, see [Channel Blocked](#). The connection is allowed to continue. An attempt is made to find another record that is set to WARN(NO) to set the credentials for the inbound channel.

If you want message AMQ9787 to be generated, you must add **Ch1authIssueWarn=y** to the [Channels stanza](#) of the `qm.ini` file.

## Related concepts

[Channel authentication records](#)

## Related tasks

[Securing remote connectivity to the queue manager](#)

## ***Generic IP addresses for channel authentication records***

In the various commands that create and display channel authentication records, you can specify certain parameters as either a single IP address or a pattern to match a set of IP addresses.

When you create a channel authentication record, using the MQSC command **SET CHLAUTH** or the PCF command **Set Channel Authentication Record**, you can specify a generic IP address in various contexts. You can also specify a generic IP address in the filter condition when you display a channel authentication record using the commands **DISPLAY CHLAUTH** or **Inquire Channel Authentication Records**.

You can specify the address in any of the following ways:

- a single IPv4 address, such as 192.0.2.0
- a pattern based on an IPv4 address, including an asterisk (\*) as a wildcard. The wildcard represents one or more parts of the address, depending on context. For example, the following values are all valid:
  - 192.0.2.\*
  - 192.0.\*
  - 192.0.\*.2
  - 192.\*.2
  - \*
- a pattern based on an IPv4 address, including a hyphen (-) to indicate a range, for example 192.0.2.1-8
- a pattern based on an IPv4 address, including both an asterisk and a hyphen, for example 192.0.\*.1-8
- a single IPv6 address, such as 2001:DB8:0:0:0:0:0:0
- a pattern based on an IPv6 address including an asterisk (\*) as a wildcard. The wildcard represents one or more parts of the address, depending on context. For example, the following values are all valid:
  - 2001:DB8:0:0:0:0:0:\*
  - 2001:DB8:0:0:0:\*
  - 2001:DB8:0:0:0:0:0:1
  - 2001:\*.1
  - \*
- a pattern based on an IPv6 address, including a hyphen (-) to indicate a range, for example 2001:DB8:0:0:0:0:0:0-8
- a pattern based on an IPv6 address, including both an asterisk and a hyphen, for example 2001:DB8:0:0:0:0:0:0-8

If your system supports both IPv4 and IPv6, you can use either address format. IBM MQ recognizes IPv4 mapped addresses in IPv6.

Certain patterns are invalid:

- A pattern cannot have fewer than the required number of parts, unless the pattern ends with a single trailing asterisk. For example 192.0.2 is invalid, but 192.0.2.\* is valid.
- A trailing asterisk must be separated from the rest of the address by the appropriate part separator (a dot (.) for IPv4, a colon (:) for IPv6). For example, 192.0\* is not valid because the asterisk is not in a part of its own.
- A pattern may contain additional asterisks provided that no asterisk is adjacent to the trailing asterisk. For example, 192.\*.2.\* is valid, but 192.0.\*\* is not valid.
- An IPv6 address pattern cannot contain a double colon and a trailing asterisk, because the resulting address would be ambiguous. For example, 2001::\* could expand to 2001:0000:\*, 2001:0000:0000:\* and so on

### Related tasks

[Mapping an IP address to an MCAUSER user ID](#)

## SET LOG (notify completion of log archiving) on Multiplatforms

On Multiplatforms, use the MQSC command SET LOG to notify the queue manager that archiving of a log extent is complete. If the log management type is not ARCHIVE, the command fails.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for SET LOG” on page 935](#)

- [“Usage notes” on page 935](#)

**Synonym:** SET LOG

### SET LOG

```
▶ SET LOG — ARCHIVED — ( — name — ) ▶
```

## Parameter descriptions for SET LOG

### ARCHIVED ( *name* )

The extent name, for example, S0000001.LOG.

## Usage notes

This command requires change authority on the queue manager object.

The command fails if the log extent is not recognized, or is being written.

The command does not fail if the extent has already been marked as having been archived.

Extents prefixed with the letter R are extents that are waiting to be reused so these extents cannot be passed to **SET LOG ARCHIVED**.

Any extent (prefixed with S) can be archived and passed to **SET LOG ARCHIVED**, except for the current extent. So extents needed for restart or media recovery, or both, can be archived and passed to **SET LOG ARCHIVED** because the queue manager has finished writing to them.

Note that extents can be archived and passed to **SET LOG ARCHIVED** in any order - not necessarily in the order in which they were written.

A message is written to the error log if the queue manager is notified about an extent more than once, either from this command, or the [“RESET QMGR \(reset a queue manager\)” on page 900](#) command.

This command is not valid on IBM i.

## SET LOG (change log system settings) on z/OS

On z/OS, use the MQSC command SET LOG to dynamically change certain log system parameter values that were initially set by your system parameter module at queue manager startup.

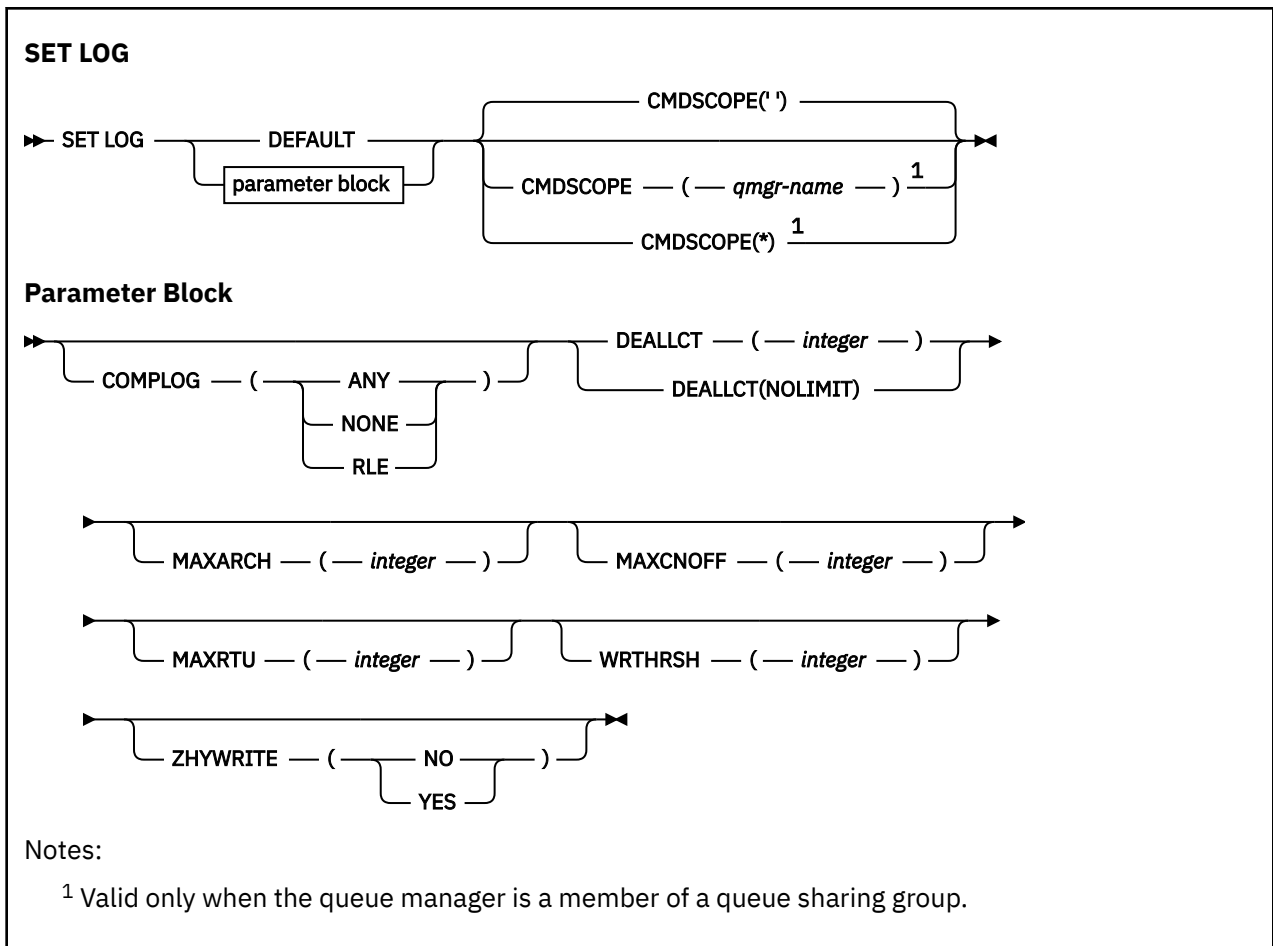
## Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for SET LOG” on page 936](#)
- [“Parameter descriptions for SET LOG” on page 936](#)
- [“Parameter block” on page 937](#)

**Synonym:** SET LOG



## Usage notes for SET LOG

1. Any changes to WRTHRSH take immediate effect.
2. Any change to MAXARCH takes effect for the next scheduled offload (that is, not for any offload in progress at the time the command is issued).

## Parameter descriptions for SET LOG

### CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

"

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which it was entered, only if you are using a queue sharing group environment and if the command server is enabled. You cannot use `CMDSCOPE(qmgr-name)` for commands issued from the first initialization input data set, CSQINP1.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.


You cannot use `CMDSCOPE(*)` for commands issued from CSQINP1.



## DEFAULT

Reset all the log system parameters to the values specified at queue manager startup.

## Parameter block

 For a full description of these parameters, see [Using CSQ6LOGP](#).

Parameter block is any one or more of the following parameters that you want to change:

### COMPLOG

This parameter specifies whether compression is used by the queue manager when writing log records. Any compressed records are automatically decompressed irrespective of the current COMPLOG setting.

The possible values are:

#### ANY


Enable the queue manager to select the compression algorithm that gives the greatest degree of log record compression. Using this option currently results in RLE compression.

#### NONE

No log data compression is used. This is the default value.

#### RLE

Log data compression is performed using run-length encoding (RLE).

 For more details about log compression, see [Log compression](#).

### DEALLCT

Specifies the length of time that an allocated archive read tape unit is allowed to remain unused before it is deallocated. You are recommended to specify the maximum possible values, within system constraints, for both options to achieve the optimum performance for reading archive tapes.

This, together with the MAXRTU parameter, allows IBM MQ to optimize archive log reading from tape devices.

The possible values are:

#### *integer*

Specifies the maximum time in minutes, in the range 0 through 1439. Zero means that a tape unit is deallocated immediately.

#### **NOLIMIT or 1440**

Indicates that the tape unit is never deallocated.

### MAXARCH

Specifies the maximum number of archive log volumes that can be recorded in the BSDS. When this number is exceeded, recording begins again at the start of the BSDS.

Use a decimal number in the range 10 through 1000.

### MAXCNOFF

Maximum number of concurrent log offload tasks.

Specify a decimal number between 1 and 31. If no value is specified the default of 31 applies.

Configure a number lower than the default if your archive logs are allocated on a tape device, and there are constraints on the number of such devices that can be concurrently allocated to the queue manager.

### MAXRTU( *integer* )

Specifies the maximum number of dedicated tape units that can be allocated to read archive log tape volumes. This overrides the value for MAXRTU set by CSQ6LOGP in the archive system parameters.

This, together with the DEALLCT parameter, allows IBM MQ to optimize archive log reading from tape devices.

#### **Note:**

1. The integer value can be in the range 1 - 99.
2. If the number specified is greater than the current specification, the maximum number of tape units allowable for reading archive logs increases.
3. If the number specified is less than the current specification, tape units that are not being used are immediately deallocated to adjust to the new value. Active, or premounted, tape units remain allocated.
4. A tape unit is a candidate for deallocation because of a lowered value only if there is no activity for the unit.
5. When you are asked to mount an archive tape and you reply CANCEL, the MAXRTU value is reset to the current number of tape units.

For example, if the current value is 10, but you reply CANCEL to the request for the seventh tape unit, the value is reset to six.

#### **WRTHRSH**

Specifies the number of 4 KB output buffers to be filled before they are written to the active log data sets.

The larger the number of buffers, the less often the write takes place, and this improves the performance of IBM MQ. The buffers might be written before this number is reached if significant events, such as a commit point, occur.

Specify the number of buffers in the range 1 through 256.

#### **ZHYWRITE**

Specifies whether writes to the active logs are made with zHyperWrite being enabled.

For more information on enabling active logs with zHyperWrite, see [Using zHyperWrite with IBM MQ active logs](#).

The value can be:

##### **NO**

zHyperWrite is not enabled.

##### **YES**

zHyperWrite is enabled.

## **SET POLICY (set security policy) on Multiplatforms**

Use the MQSC command SET POLICY to set a security policy.

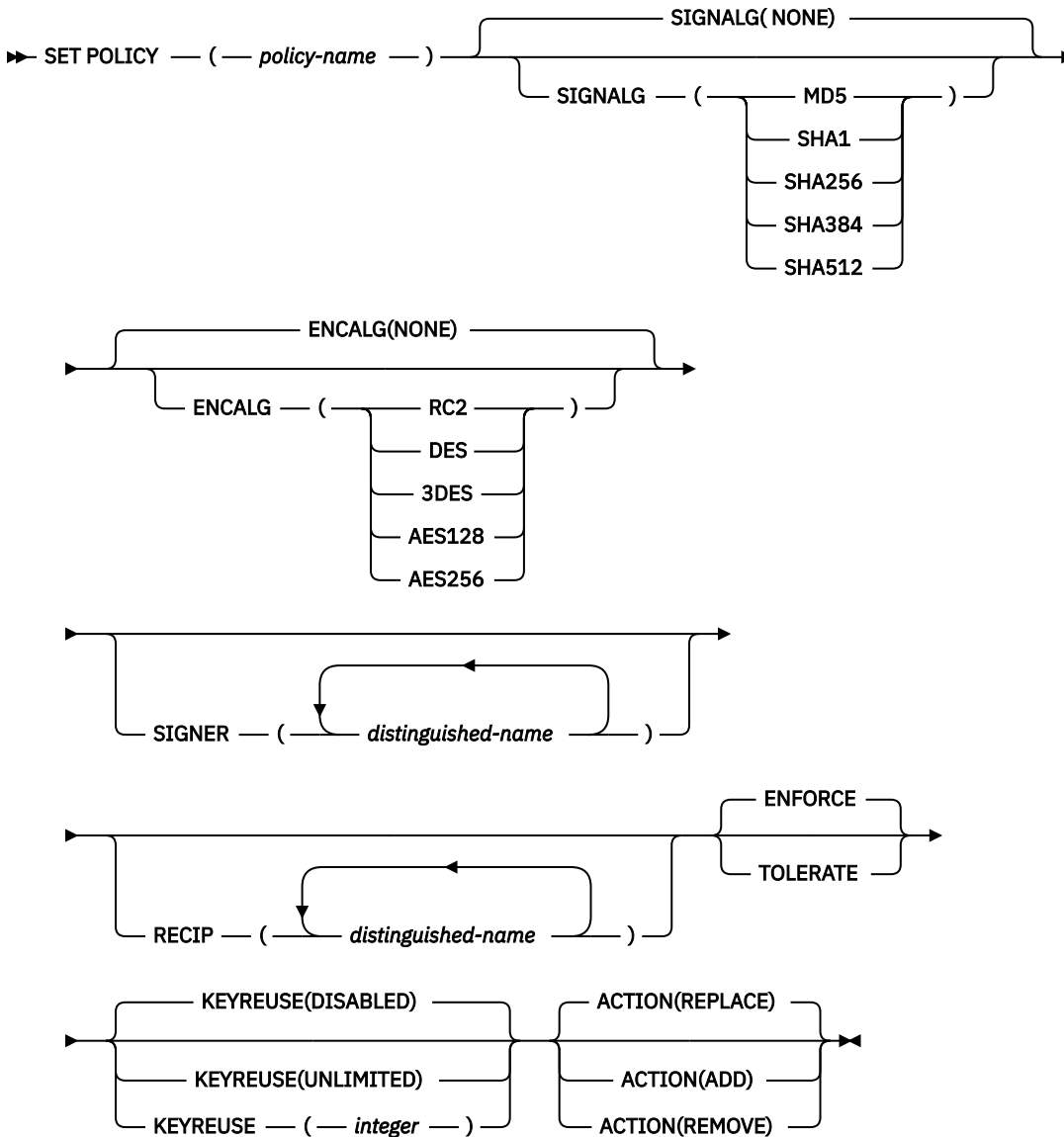
### **Using MQSC commands**

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for SET POLICY” on page 939](#)

**Important:** You must have an Advanced Message Security (AMS) license installed to issue this command. If you attempt to issue the **SET POLICY** command without an AMS license installed, you receive message AMQ7155 - License file not found or not valid.

## SET POLICY



### Parameter descriptions for SET POLICY

#### **(policy-name)**

Name of the policy, required.

The policy name must match the name of the queue which is to be protected.

#### **SIGNALG**




Specifies the digital signature algorithm from one of the following values:

- NONE
- **Deprecated** MD5
- **Deprecated** SHA-1
- SHA256
- SHA384
- SHA512

The default value is NONE.

### **ENCALG**

Specifies the digital encryption algorithm from one of the following values:

- NONE
-  [RC2](#)
-  [DES](#)
-  [3DES](#)
- AES128
- AES256

The default value is NONE.

### **RECIP (*distinguished-name*)**

Specifies the message distinguished name (DN) of the recipient, that is, the certificate of a DN provided used to encrypt a given message.

#### **Notes:**

1. The attributes names for DNs must be provided in capital letters.
2. Commas must be used as a name separator.
3. You must specify at least one recipient, if you use any encryption algorithm other than NONE.

You can specify multiple **RECIP** parameters on the same policy.

### **SIGNER (*distinguished-name*)**

Specifies a signature DN that is validated during the message retrieval. Only messages signed by the user, with a DN provided, are accepted during retrieval.

#### **Notes:**

1. The attributes name for DNs must be provided in capital letters.
2. Commas must be used as a name separator.
3. You can specify signature DNs, only if you use any signature algorithm other than NONE.

You can specify multiple **SIGNER** parameters on the same policy.

### **ENFORCE**

Specifies that all messages must be protected when retrieved from the queue.

Any unprotected message encountered is moved to the SYSTEM.PROTECTION.ERROR.QUEUE.

**ENFORCE** is the default value.

### **TOLERATE**

Specifies that the messages that are not protected when retrieved from the queue can ignore the policy.

**TOLERATE** is optional and exists to facilitate staged implementation, where:

- Policies have been applied to queues, but those queues might already contain unprotected messages, or
- Queues might still receive messages from remote systems that do not yet have the policy set.

### **KEYREUSE**

Specify the number of times that an encryption key can be re-used, in the range 1-9999999, or the special values *DISABLED* or *UNLIMITED*.

Note that this is a maximum number of times a key can be reused, therefore a value of 1 means, at most, two messages can use the same key.

## DISABLED

Prevents a symmetric key from being reused

## UNLIMITED

Allows a symmetric key to be reused any number of times.

*DISABLED* is the default value.



**Attention:** Key reuse is valid only for CONFIDENTIALITY policies, that is, **SIGNALG** set to *NONE* and **ENCALG** set to an algorithm value. For all other policy types, you must omit the parameter, or set the **KEYREUSE** value to *DISABLED*.

## ACTION

Specify the action for the parameters supplied, as they apply to any existing policy, using one of the following values:

### REPLACE

Has the effect of replacing any existing policy with the parameters supplied.

### ADD

Has the effect that signers and recipients parameters have an additive effect. That is, if a signer or recipient is specified, and does not already exist in a preexisting policy, the signer or recipient value is added to the existing policy definition.

### REMOVE

Has the opposite effect of *ADD*. That is, if any of the signer or recipient values specified exist in a preexisting policy, those values are removed from the policy definition.


*REPLACE* is the default value.

## Related reference


[“DISPLAY POLICY \(display a security policy\) on Multiplatforms” on page 755](#)

Use the MQSC command **DISPLAY POLICY** to display a security policy.

[“setmqspl \(set security policy\)” on page 216](#)

Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy.  On z/OS you use the command with the CSQOUTIL utility.

[“dspmqspl \(display security policy\)” on page 99](#)

Use the **dspmqspl** command to display a list of all policies and details of a named policy.  On z/OS you use the command with the CSQOUTIL utility.

## **SET SYSTEM (change system settings) on z/OS**

Use the MQSC command SET SYSTEM to dynamically change certain general system parameter values that were initially set from your system parameter module at queue manager startup. To permanently change these, either change the CSQ6SYSP parameters and regenerate the parameter module, or put the SET SYSTEM commands into a data set in the CSQINP2 concatenation.

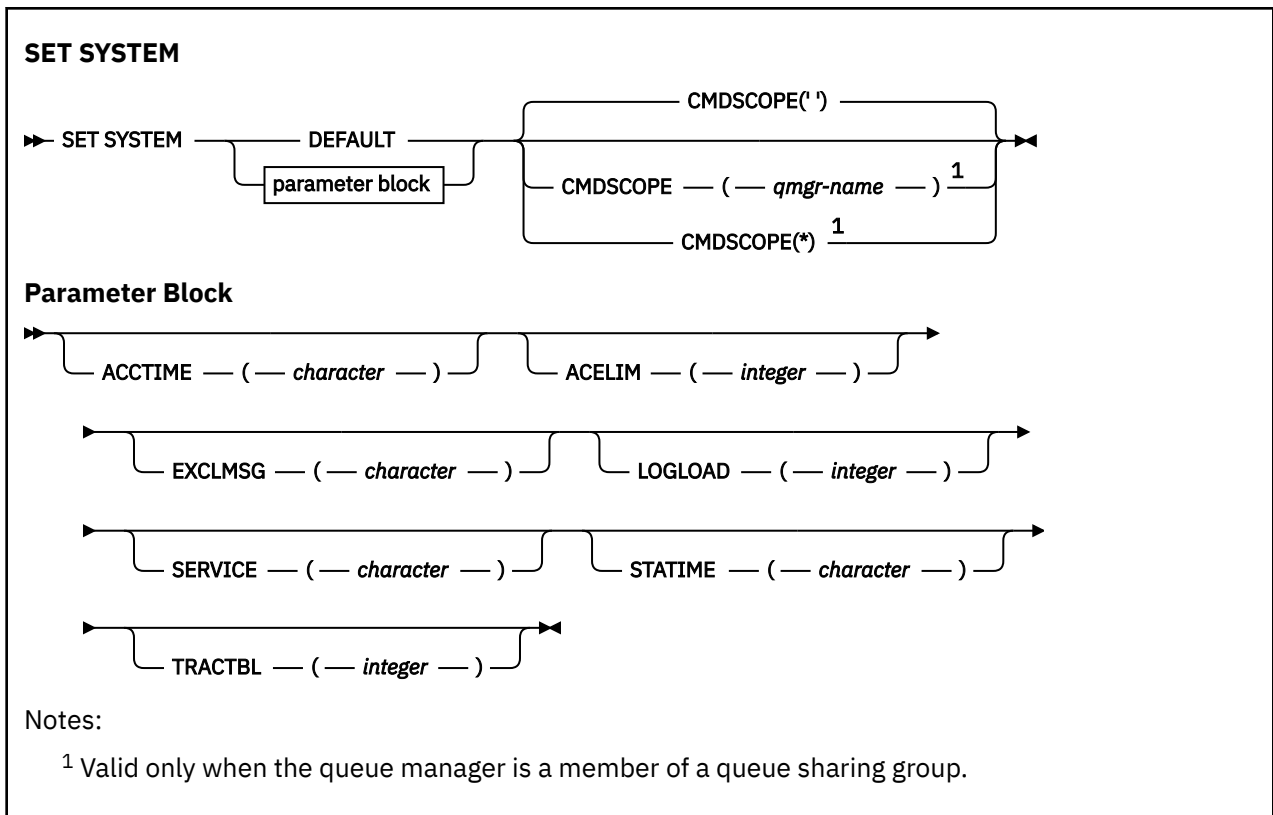
## Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for SET SYSTEM” on page 942](#)
- [“Parameter descriptions for SET SYSTEM” on page 942](#)
- [“Parameter block” on page 943](#)

**Synonym:** None



The CTHREAD, IDFORE, and IDBACK parameters are ignored in IBM WebSphere MQ 7.1 or later, but are still allowed for compatibility with earlier versions. Any attempt to change the value of one of these parameters sets it to a default value of 32767.

## Usage notes for SET SYSTEM

The new values take immediate effect, with the possible exception of STATIME, **V9.3.0** ACCTIME, and TRACTBL.

Changes to STATIME **V9.3.0** and ACCTIME take effect when the current interval expires, unless the new interval is less than the unexpired portion of the current interval, in which case SMF data is gathered immediately and the new interval then takes effect.

For TRACTBL, if there is any trace currently in effect, the existing trace table continues to be used, and its size is unchanged. A new global trace table is only obtained for a new START TRACE command. If a new trace table is created with insufficient storage, the old trace table continues to be used, and the message CSQW153E is displayed.

## Parameter descriptions for SET SYSTEM

### CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which it was entered, only if you are using a queue sharing group environment and if the command server is enabled.

You cannot use CMDSCOPE( *qmgr-name*) for commands issued from the first initialization input data set, CSQINP1.

\*


The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

You cannot use CMDSCOPE(\*) for commands issued from CSQINP1.

#### DEFAULT

Resets all the general system parameters to the values set at queue manager startup.

## Parameter block

 For a full description of these parameters, see [Using CSQ6SYSP](#).

Parameter block is any one or more of the following parameters that you want to change:

### ACCTIME

Specifies the interval, in minutes and seconds, between consecutive gatherings of accounting data.

Specify a number in the range 0 through 1440 minutes in the format 'mmm', or in the range 0 through 1439 minutes, and 0 - 59 seconds, in the format 'mmm.ss'. The default is 30 minutes.

Specify a number, either -1, or in the range 0 through 1440 minutes in the format 'mmm', or in the range 0 through 1439 minutes, and 0 - 59 seconds, in the format 'mmm.ss'.

#### Notes:

- When specifying an interval of seconds only, you must prefix the interval with a value of 0. The smallest possible interval is one second: '0.01'.
- If you specify a value of 0, accounting data is collected at the SMF data collection broadcast. See [Using System Management Facility](#) for more information.
- If you specify a value of -1, which is the default, accounting data is collected using the STATIME value.

For example:

'0.30' sets an interval of 30 seconds.

'5.30' sets an interval of 5 minutes and 30 seconds.

'30' sets an interval of 30 minutes.

### ACELIM

Specifies the maximum size of the ACE storage pool in 1 KB blocks. The number must be in the range 0-9999999. The default value of zero means no imposed constraint, beyond what is available in the system.

You should only set a value for ACELIM on queue managers that have been identified as using exorbitant quantities of ECSA storage. Limiting the ACE storage pool has the effect of limiting the number of connections in the system, and so, the amount of ECSA storage used by a queue manager.

Once the queue manager reaches the limit it is not possible for applications to obtain new connections. The lack of new connections causes failures in MQCONN processing, and applications coordinated through RRS are likely to experience failures in any IBM MQ API.

An ACE represents approximately 12.5% of the total ECSA required for the thread-related control blocks for a connection. So, for example, specifying ACELIM=5120 would be expected to cap the total amount of ECSA allocated by the queue manager (for thread-related control blocks) at approximately 40960K; that is 5120 multiplied by 8.

In order to cap the amount total amount of ECSA allocated by the queue manager, for thread-related control blocks at 5120K, an ACELIM value of 640 is required.

You can use SMF 115 subtype 5 records, produced by statistics CLASS(3) trace, to monitor the size of the 'ACE/PEB' storage pool, and hence set an appropriate value for ACELIM.

You can obtain the total amount of ECSA storage used by the queue manager, for control blocks, from SMF 115 subtype 7 records, written by statistics CLASS(2) trace. The total amount of ECSA storage used is the sum of the QSRSPHBGF and QSRSPHBGV fields.

For more information about SMF 115 statistics records, see [Interpreting IBM MQ performance statistics](#).

Note that, you should consider setting ACELIM as a mechanism to protect a z/OS image from a badly behaving queue manager, rather than as a means to control application connections to a queue manager.

## EXCLMSG

Specify a list of message identifiers to be excluded from being written to any log. Messages in this list are not sent to the z/OS console and hardcopy log. As a result using the EXCLMSG parameter to exclude messages is more efficient from a CPU perspective than using z/OS mechanisms such as the message processing facility list and should be used instead where possible. This list is dynamic and is updated using the SET SYSTEM command.

The default value is an empty list ( ).

Message identifiers are supplied without the CSQ prefix and without the action code suffix (I-D-E-A). For example, to exclude message CSQX500I, add X500 to this list. This list can contain a maximum of 16 message identifiers.

To be eligible to be included in the list, the message must be issued after normal startup of the MSTR or CHIN address spaces and begin with the one of the following characters E, H, I, J, L, M, N, P, R, T, V, W, X, Y, 2, 3, 5, 9.

Message identifiers that are issued as a result of processing commands can be added to the list, however are not excluded.

For example:

```
SET SYSTEM EXCLMSG(X511,X512)
```

suppresses the channel started and channel no longer active messages.

## LOGLOAD

Specifies the number of log records that IBM MQ writes between the start of one checkpoint and the next. IBM MQ starts a new checkpoint after the number of records that you specify has been written.

Specify a value in the range 200 through 16 000 000.

## SERVICE

This parameter is reserved for use by IBM.

## STATIME

**V 9.3.0** From IBM MQ for z/OS 9.2.4, specifies the time, in minutes and seconds, between consecutive gatherings of statistics data. If ACCTIME is not set, or is -1, also specifies the time, between consecutive gatherings of accounting data.

If you specify a value of 0, data is collected at the SMF data collection broadcast.

Specify a number in the range 0 through 1440 minutes in the format 'mmm', or in the range 0 through 1439 minutes, and 0 - 59 seconds, in the format 'mmm.ss'. The default is 30 minutes.

Specify a number, either -1, or in the range 0 through 1440 minutes in the format 'mmm', or in the range 0 through 1439 minutes, and 0 - 59 seconds, in the format 'mmm.ss'.

**Note:** When specifying an interval of seconds only, you must prefix the interval with a value of 0. The smallest possible interval is one second: '0.01'.

For example:



- '0.30' sets an interval of 30 seconds.
- '5.30' sets an interval of 5 minutes and 30 seconds.
- '30' sets an interval of 30 minutes.

### TRACTBL

Specifies the default size, in 4 KB blocks, of trace table where the global trace facility stores IBM MQ trace records.

Specify a value in the range 1 through 999.

**Note:** Storage for the trace table is allocated in the ECSA. Therefore, you must select this value with care.

## START CHANNEL (start a channel)

Use the MQSC command **START CHANNEL** to start a channel.

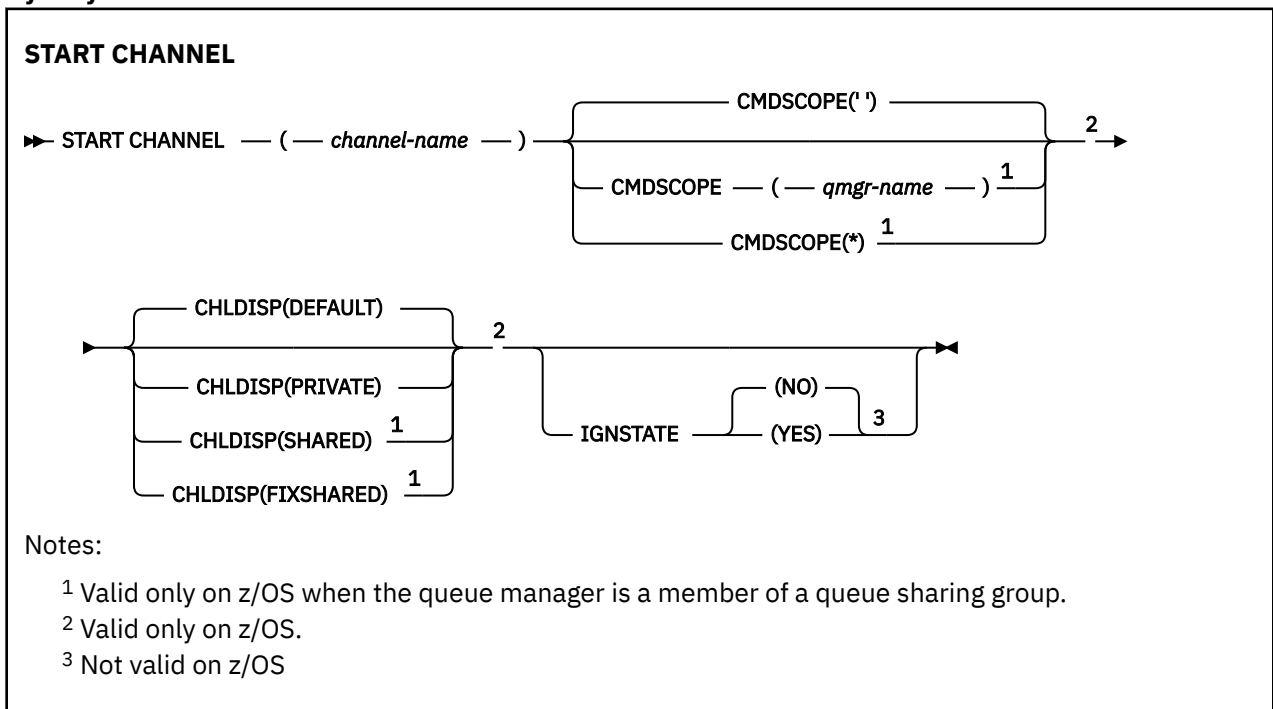
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 945](#)
- [“Parameter descriptions for START CHANNEL” on page 946](#)

**Synonym:** STA CHL



### Usage notes

1. **z/OS** On z/OS, the command server and the channel initiator must be running.
2. This command can be issued to a channel of any type except CLNTCONN channels (including those that have been defined automatically). If, however, it is issued to a receiver (RCVR), server-connection (SVRCONN) or cluster-receiver (CLUSRCVR) channel, the only action is to enable the channel, not to start it.

3. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager's repository.

## Parameter descriptions for START CHANNEL

### (channel-name)

The name of the channel definition to be started. This is required for all channel types. The name must be that of an existing channel.

### CHLDISP

This parameter applies to z/OS only and can take the values of:

- DEFAULT
- PRIVATE
- SHARED
- FIXSHARED

If this parameter is omitted, then the DEFAULT value applies. This is taken from the default channel disposition attribute, **DEFCDISP**, of the channel object.

In conjunction with the various values of the **CMDSCOPE** parameter, this parameter controls two types of channel:

#### SHARED

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of SHARED.

#### PRIVATE

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than SHARED.

**Note:** This disposition is not related to the disposition set by the disposition of the queue sharing group of the channel definition.

The combination of the **CHLDISP** and **CMDSCOPE** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On every active queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of **CHLDISP** and **CMDSCOPE** are summarized in the following table:

<i>Table 181. CHLDISP and CMDSCOPE for START CHANNEL</i>			
<b>CHLDISP</b>	<b>CMDSCOPE( ) or CMDSCOPE (local-qmgr)</b>	<b>CMDSCOPE (qmgr-name)</b>	<b>CMDSCOPE(*)</b>
PRIVATE	Start as a private channel on the local queue manager	Start as a private channel on the named queue manager	Start as a private channel on all active queue managers

Table 181. CHLDISP and CMDSCOPE for START CHANNEL (continued)

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)	CMDSCOPE(*)
SHARED	<p>For a shared SDR, RQSTR, and SVR channel, start as a shared channel on the most suitable queue manager in the group.</p> <p>For a shared RCVR and SVRCONN channel, start the channel as a shared channel on all active queue managers.</p> <p>For a shared CLUSSDR or CLUSRCVR channel, this option is not permitted.</p> <p>This might automatically generate a command using CMDSCOPE and send it to the appropriate queue managers. If there is no definition for the channel on the queue managers to which the command is sent, or if the definition is unsuitable for the command, the action fails there.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted	Not permitted
FIXSHARED	<p>For a shared SDR, RQSTR, and SVR channel, with a nonblank CONNAME, start as a shared channel on the local queue manager.</p> <p>For all other types, this option is not permitted.</p>	<p>For a shared SDR, RQSTR, and SVR with a nonblank CONNAME, start as a shared channel on the named queue manager.</p> <p>For all other types, this option is not permitted.</p>	Not permitted

Channels started with **CHLDISP (FIXSHARED)** are tied to the specific queue manager; if the channel initiator on that queue manager stops for any reason, the channels are not recovered by another queue manager in the group. For more information about SHARED and FIXSHARED channels, see [Starting a shared channel](#).

 **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

If **CHLDISP** is set to SHARED, **CMDSCOPE** must be blank or the local queue manager.

..

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

This option is not permitted if **CHLDISP** is FIXSHARED.

#### Multi **IGNSTATE**

Specifies whether the command fails if the channel is already running. The possible values are:

##### **NO**

The command fails if the channel is already running. This is the default value.

##### **YES**

The command succeeds regardless of the current state of the channel.

#### ALW **START CHANNEL (start a channel) MQTT**

Use the MQSC command START CHANNEL to start an MQ Telemetry channel.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

The START CHANNEL (MQTT) command is only valid for MQ Telemetry channels. Supported platforms for MQ Telemetry are AIX, Linux, Windows.

**Synonym:** STA CHL

#### **START CHANNEL**

▶ START CHANNEL — ( — *channel-name* — ) — CHLTYPE — ( — MQTT — ) ▶

### Parameter descriptions for START CHANNEL

#### **(channel-name)**

The name of the channel definition to be started. The name must be that of an existing channel.

#### **CHLTYPE**

Channel type. The value must be MQTT.

#### z/OS **START CHINIT (start a channel initiator) on z/OS**

Use the MQSC command START CHINIT to start a channel initiator.

### Using MQSC commands on z/OS

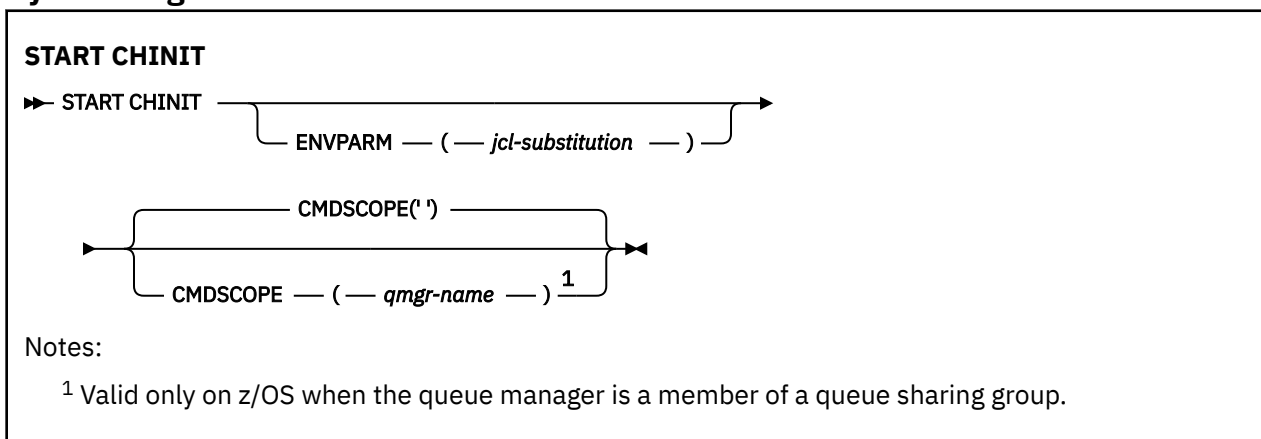
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 949](#)
- [“Parameter descriptions for START CHINIT” on page 949](#)

**Synonym:** STA CHI

## Syntax diagram



## Usage notes

1. The command server must be running.
2. Although START CHINIT is permitted from CSQINP2, its processing is not complete (and the channel initiator is not available) until after CSQINP2 processing has finished. For these commands, consider using [CSQINPX](#) instead.

## Parameter descriptions for START CHINIT

### CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

### ENVPARM(*jcl-substitution*)

The parameters and values to be substituted in the JCL procedure (xxxxCHIN, where xxxx is the queue manager name) that is used to start the channel initiator address space.

### *jcl-substitution*

One or more character strings of the form keyword=value enclosed in single quotation marks. If you use more than one character string, separate the strings by commas and enclose the entire list in single quotation marks, for example ENVPARM('HLQ=CSQ,VER=520').

This parameter is valid only on z/OS.

### INITQ(*string*)

The name of the initiation queue for the channel initiation process. This is the initiation queue that is specified in the definition of the transmission queue.

The initiation queue on z/OS is always SYSTEM.CHANNEL.INITQ).

## Related concepts

[Command resource security checking for alias queues and remote queues](#)

## **START CMDSERV (start the command server) on z/OS**

Use the MQSC command START CMDSERV to initialize the command server.

### Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12C. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for START CMDSERV” on page 950](#)

**Synonym:** STA CS

#### **START CMDSERV**

▶▶ START CMDSERV ◀◀

### Usage notes for START CMDSERV


1. START CMDSERV starts the command server and allows it to process commands in the system-command input queue (SYSTEM.COMMAND.INPUT), mover commands, and commands using CMDSCOPE.
2. If this command is issued through the initialization files or through the operator console before work is released to the queue manager (that is, before the command server is started automatically), it overrides any earlier STOP CMDSERV command and allows the queue manager to start the command server automatically by putting it into an ENABLED state.
3. If this command is issued through the operator console while the command server is in a STOPPED or DISABLED state, it starts the command server and allows it to process commands on the system-command input queue, mover commands, and commands using CMDSCOPE immediately.
4. If the command server is in a RUNNING or WAITING state (including the case when the command is issued through the command server itself), or if the command server has been stopped automatically because the queue manager is closing down, no action is taken, the command server remains in its current state, and an error message is returned to the command originator.
5. START CMDSERV can be used to restart the command server after it has been stopped, either because of a serious error in handling command messages, or commands using the CMDSCOPE parameter.


## **START LISTENER (start a channel listener)**

Use the MQSC command START LISTENER to start a channel listener.

### Using MQSC commands

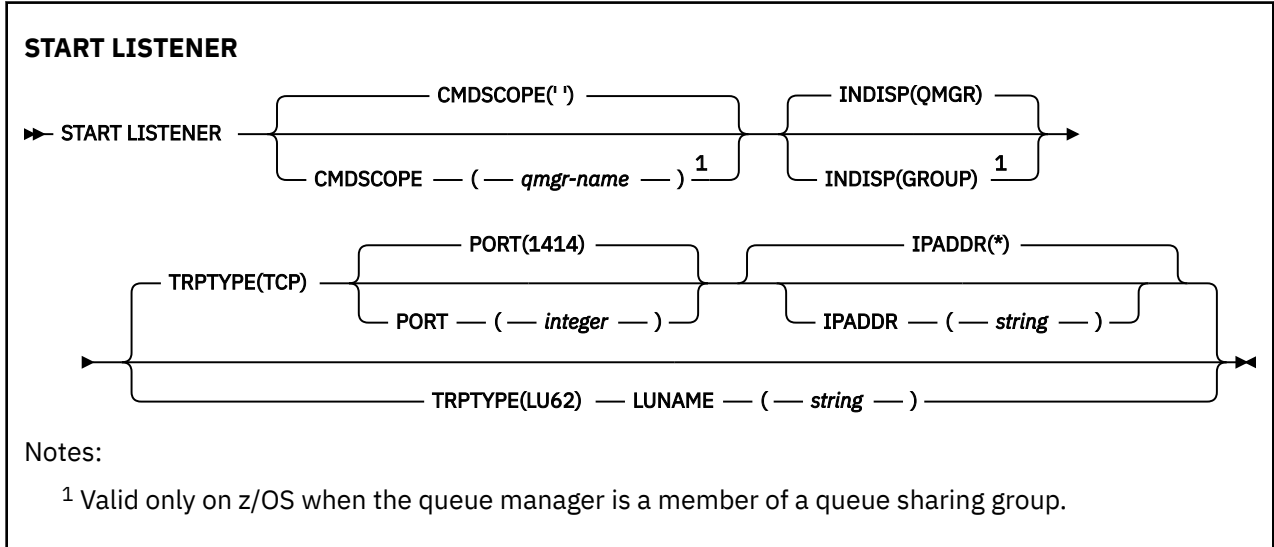
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

 You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

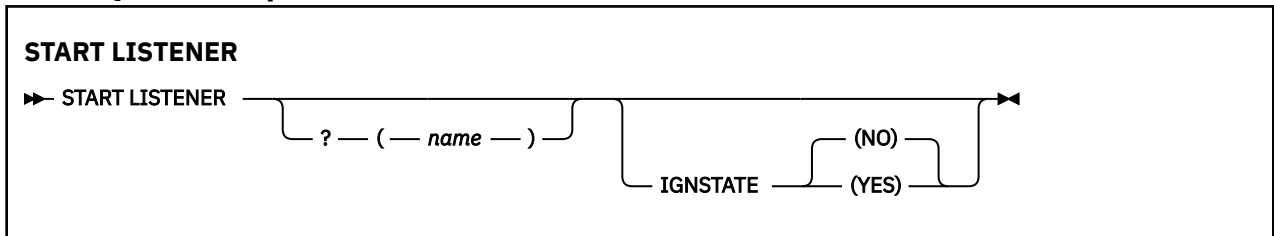
-  [Syntax diagram for IBM MQ for z/OS](#)
- [Syntax diagram for IBM MQ on other platforms](#)
- [“Usage notes” on page 951](#)
- [“Parameter descriptions for START LISTENER” on page 952](#)

**Synonym:** STA LSTR

## IBM MQ for z/OS



## IBM MQ on other platforms



## Usage notes

1. On z/OS:
  - a. The command server and the channel initiator must be running.
  - b. If IPADDR is not specified, the listener listens on all available IPv4 and IPv6 addresses.
  - c. For TCP/IP, it is possible to listen on multiple addresses and port combinations.
  - d. For each START LISTENER for TCP/IP request, the address and port combination is added to the list of combinations upon which the listener is currently listening.
  - e. A START LISTENER for TCP/IP request fails if it specifies the same, or a subset or superset of an existing, combination of addresses and ports upon which a TCP/IP listener is currently listening.
  - f. If you are starting a listener on a specific address to provide a secure interface with a security product, for example a firewall, it is important to ensure there is no linkage to the other non-secure interfaces in the system.


You should disable IP forwarding and routing from other non-secure interfaces so that packets arriving at the other interface do not get passed to this specific address.


Consult the appropriate TCP/IP documentation for information on how to do this.
2. On Multiplatforms, this command is valid only for channels for which the transmission protocol (TRPTYPE) is TCP.

## Parameter descriptions for START LISTENER

### ( *name* )

Name of the listener to be started. If you specify this parameter, you cannot specify any other parameters.

If you do not specify a name  (on platforms other than z/OS ), the SYSTEM.DEFAULT.LISTENER.TCP is started.

 This parameter is not valid on z/OS.

### **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

### **INDISP**

Specifies the disposition of the inbound transmissions that are to be handled. The possible values are:

#### **QMGR**

Listen for transmissions directed to the queue manager. This is the default.

#### **GROUP**

Listen for transmissions directed to the queue sharing group. This is allowed only if there is a shared queue manager environment.

This parameter is valid only on z/OS.

### **IPADDR**

IP address for TCP/IP specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric form. This is valid only if the transmission protocol (TRPTYPE) is TCP/IP.

This parameter is valid only on z/OS.

### **LUNAME( *string* )**

The symbolic destination name for the logical unit as specified in the APPC side information data set. (This must be the same LU that was specified for the queue manager, using the LUNAME parameter of the ALTER QMGR command.)

This parameter is valid only for channels with a transmission protocol (TRPTYPE) of LU 6.2. A START LISTENER command that specifies TRPTYPE(LU62) must also specify the LUNAME parameter.

This parameter is valid only on z/OS.

### **PORT( *port-number* )**

Port number for TCP. This is valid only if the transmission protocol (TRPTYPE) is TCP.

This parameter is valid only on z/OS.

### **TRPTYPE**

Transport type to be used. This is optional.

#### **TCP**

TCP. This is the default if TRPTYPE is not specified.



## LU62

SNA LU 6.2.

This parameter is valid only on z/OS.

### Multi **IGNSTATE**

Specifies whether the command fails if the listener is already running. The possible values are:

#### **NO**

The command fails if the listener is already running. This is the default value.

#### **YES**

The command succeeds regardless of the current state of the listener.

### **z/OS** **START QMGR (start queue manager) on z/OS**

Use the MQSC command **START QMGR** to initialize the queue manager. From IBM MQ 9.2.0, you can also use this command to prepare its data for backwards migration using the **BACKMIG** parameter.

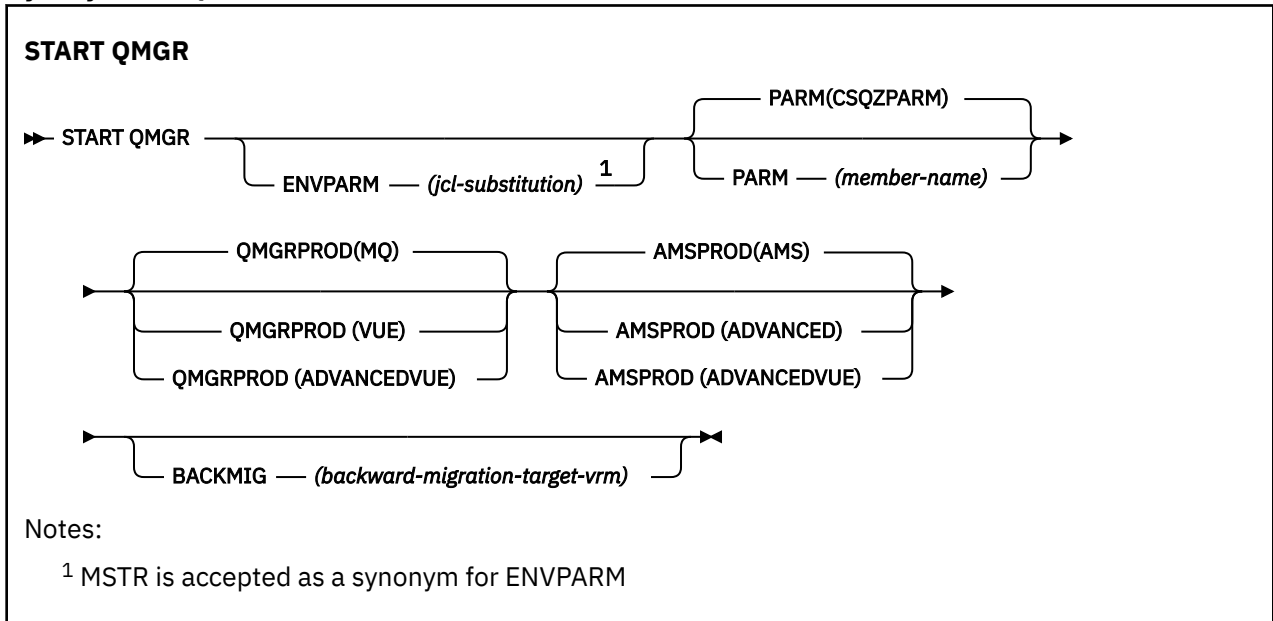
## Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources C. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 953](#)
- [“Parameter descriptions for START QMGR” on page 954](#)

**Synonym:** STA QMGR



## Usage notes

When the command has been completed, the queue manager is active and available to CICS, IMS, batch, and TSO applications, unless you specify the **BACKMIG** parameter. If you specify the **BACKMIG** parameter, the queue manager shuts down and is ready for backwards migration, before becoming available to any applications.

If you specify the BACKMIG parameter, together with a value for *backward-migration-target-vm* that is not valid, the queue manager ends abnormally.

The startup parameters **QMGRPROD** and **AMSPROD** indicate against which product that component should have its usage recorded.

You can specify the attribute for the queue manager:

- As a parameter on the START QMGR command
- As a part of the PARM on the EXEC PGM statement in the MSTR JCL procedure
- As part of the compiled queue manager ZPARMS, using the [CSQ6USGP](#) macro
- As a default value if not specified elsewhere.

If you specify the attribute by more than one of the above mechanisms, the order of the items in the preceding list defines the order of precedence from highest to lowest. The default value is used if you do not explicitly specify an attribute.

If you specify an attribute that is not valid, an error message is issued and queue manager startup ends.

## Parameter descriptions for START QMGR

These are optional.

### ENVPARM(*jcl-substitution*)

The parameters and values to be substituted in the JCL procedure (xxxxMSTR, where xxxx is the queue manager name) that is used to start the queue manager address space.

#### *jcl-substitution*

One or more character strings of the form:

```
keyword=value
```

enclosed in single quotation marks. If you use more than one character string, separate the strings by commas and enclose the entire list in single quotation marks, for example ENVPARM('HLQ=CSQ,VER=520').

MSTR is accepted as a synonym for ENVPARM

### PARM(*member-name*)

The load module that contains the queue manager initialization parameters. *member-name* is the name of a load module provided by the installation.


The default is CSQZPARM, which is provided by IBM MQ.

### QMGRPROD

Specifies the product ID against which the queue manager usage is to be recorded. The value can be one of the following:

#### MQ


The queue manager is a stand-alone IBM MQ for z/OS product, with product ID 5655-MQ9.

 Prior to IBM MQ for z/OS 9.1.3, this is the default value if the SCUEAUTH library is not part of the queue manager STEPLIB.

From IBM MQ for z/OS 9.1.3 this is the default value.

#### VUE

The queue manager is a stand-alone VUE product, with product ID 5655-VU9.

 Prior to IBM MQ for z/OS 9.1.3, this is the default value if the SCUEAUTH library is part of the queue manager STEPLIB

#### ADVANCEDVUE


The queue manager is part of an IBM MQ Advanced for z/OS Value Unit Edition product, with product ID 5655-AV1.

## AMSPROD

Specifies the product ID against which the queue manager usage is to be recorded. The value can be one of the following:

### AMS

Advanced Message Security (AMS) is a stand-alone Advanced Message Security for z/OS product, with product ID 5655-AM9.

 This is the default value, unless the attribute for the queue manager indicates IBM MQ Advanced for z/OS Value Unit Edition.

### ADVANCED

AMS is part of an IBM MQ Advanced for z/OS product, with product ID 5655-AV9.

### ADVANCEDVUE

AMS is part of an IBM MQ Advanced for z/OS Value Unit Edition product, with product ID 5655-AV1. This is the default value, if the attribute for the queue manager is also **ADVANCEDVUE**.

## BACKMIG( *backward-migration-target-rrm* )

The queue manager is to start up and perform backward migration actions such that it can be restarted at the *backward-migration-target-rrm*, and then the queue manager is to shut down without ever being available to any applications.

### *backward-migration-target-rrm*

The version, release and modification number for the target version for backward migration, for example 910. Backward migration to this release must be allowed.

The release, if any, to which backward migration is allowed, is indicated in the `CSQY039I` message at queue manager start up. If the `CSQY039I` message is not emitted, backward migration is not supported to any release.



**Attention:** Follow the process documented in [Reverting a queue manager to a previous version on z/OS](#) when using the BACKMIG parameter.

Multi

## START SERVICE (start a service) on Multiplatforms

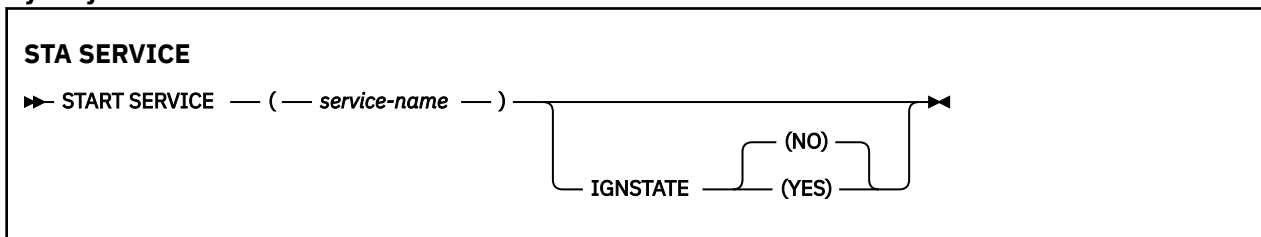
Use the MQSC command **START SERVICE** to start a service. The identified service definition is started within the queue manager and inherits the environment and security variables of the queue manager.

## Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Parameter descriptions for START SERVICE” on page 955](#)

### Synonym:



## Parameter descriptions for START SERVICE

### ( *service-name* )

The name of the service definition to be started. This is required. The name must that of an existing service on this queue manager.

If the service is already running, and the operating system task is active, an error is returned.

## Multi IGNSTATE

Specifies whether the command fails if the service is already running. The possible values are:

### NO

The command fails if the service is already running. This is the default value.

### YES

The command succeeds regardless of the current state of the service.

### Related concepts

[Working with services](#)

### Related tasks

[Managing services](#)

[Using a server service object](#)

[Using a command service object](#)

## z/OS

### START SMDSCONN (restart a shared message data set connection)

#### on z/OS

Use the MQSC command START SMDSCONN to enable a previously stopped connection from this queue manager to the specified shared message data sets, allowing them to be allocated and opened again.

#### Using MQSC commands on z/OS

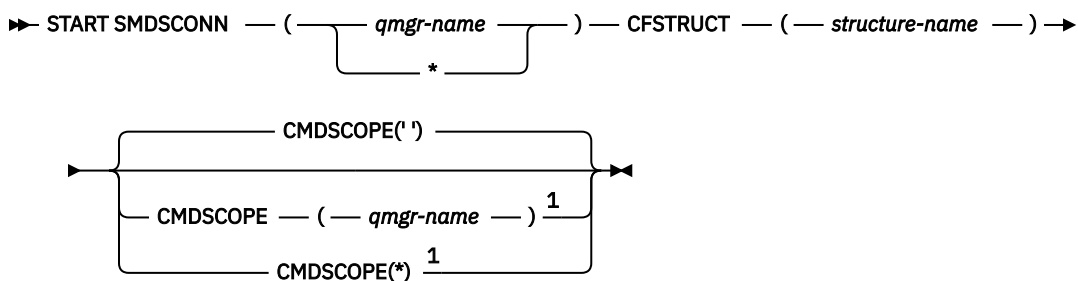
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for START SMDSCONN” on page 956](#)

#### Synonym:

#### START SMDSCONN



#### Notes:

<sup>1</sup> Valid only when the queue manager is a member of a queue sharing group.

#### Parameter descriptions for START SMDSCONN

This command is used after connections have been put into the AVAIL(STOPPED) state by a previous STOP SMDSCONN command. It can also be used to signal to the queue manager to retry a connection which is in the AVAIL(ERROR) state after a previous error.

**SMDSCONN(*qmgr-name* | \*)**

Specify the queue manager which owns the shared message data set for which the connection is to be started or an asterisk to start connections to all shared message data sets associated with the specified structure.

**CFSTRUCT(*structure-name*)**

Specify the structure name for which shared message data set connections are to be started.

**CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## **START TRACE (start trace) on z/OS**

Use the MQSC command START TRACE to start traces.

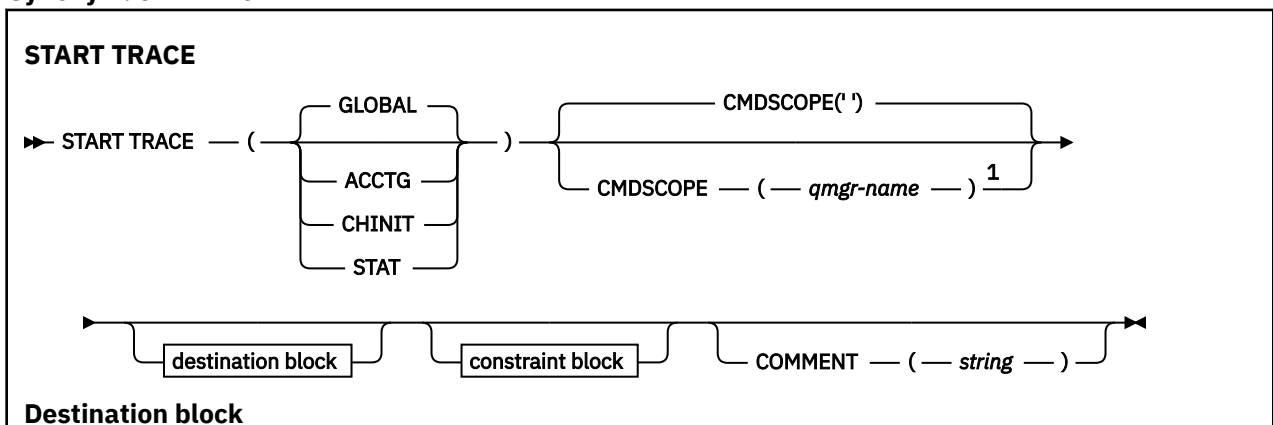
**Using MQSC commands on z/OS**

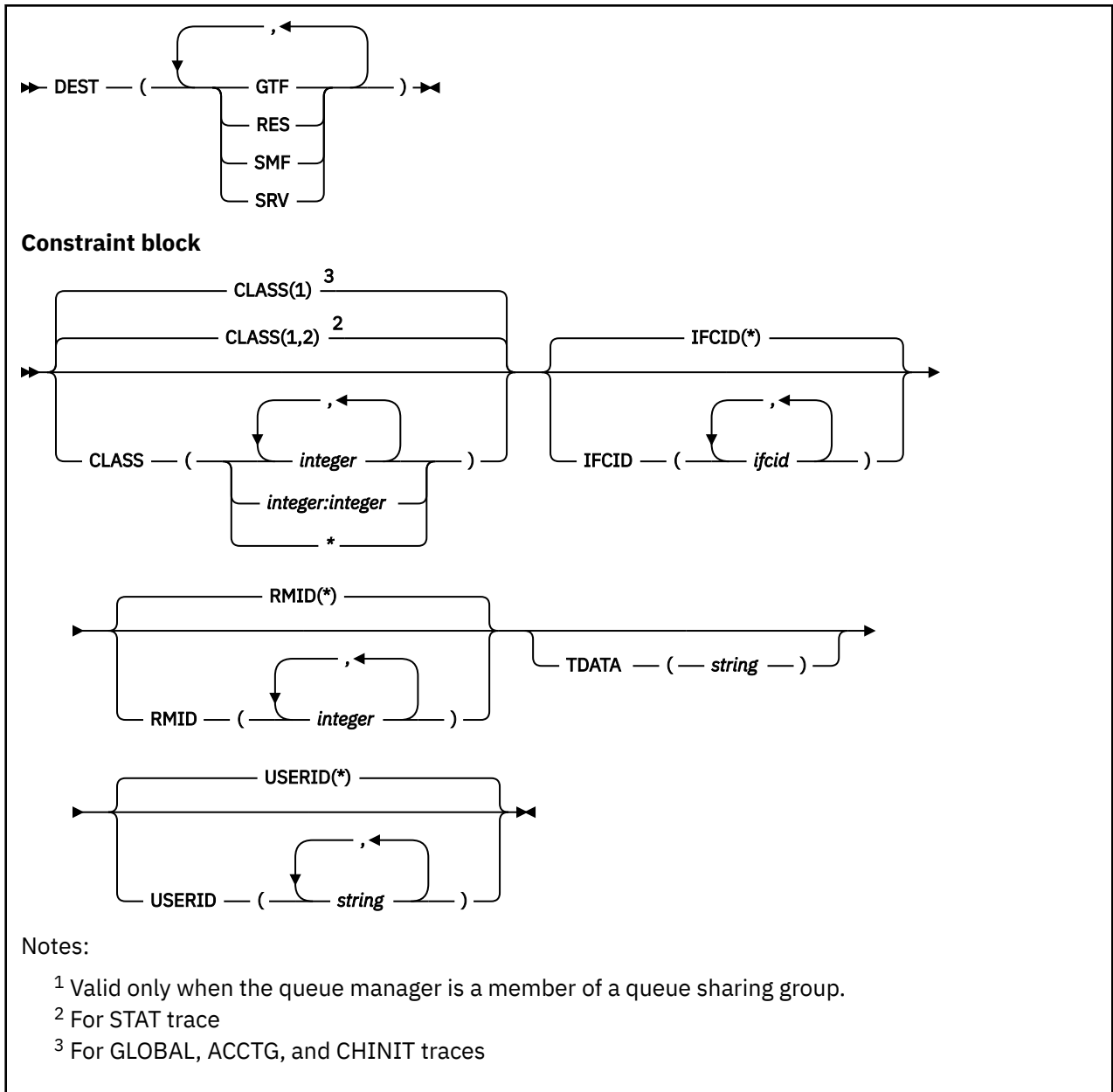
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes” on page 958](#)
- [“Parameter descriptions for START TRACE” on page 958](#)
- [“Destination block” on page 959](#)
- [“Constraint block” on page 960](#)

**Synonym:** STA TRACE





## Usage notes

When you issue this command, a trace number is returned in message number CSQW130I. You can use this trace number (TNO) in ALTER TRACE, DISPLAY TRACE, and STOP TRACE commands.

## Parameter descriptions for START TRACE

If you do not specify a trace type to be started, the default (GLOBAL) trace is started. The types are:

### ACCTG

Enables accounting data which provides information about how applications are interacting with the queue manager in the form of SMF 116 records. The synonym is A.

**Note:** Accounting data can be lost if the accounting trace is started or stopped while applications are running. For information about the conditions that must be satisfied for successful collection of accounting data, see [Using IBM MQ trace](#).

### CHINIT

This includes data from the channel initiator. The synonym is CHI or DQM. If tracing for the channel initiator is started, it stops if the channel initiator stops.

Note that you cannot issue START TRACE(CHINIT) if the command server or the channel initiator is not running.

#### **GLOBAL**

This includes data from the entire queue manager except the channel initiator. The synonym is G.

#### **STAT**

Enables high level statistics about the state of the queue manager in the form of SMF 115 records. The synonym is S.

#### **CMDSCOPE**

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### ***qmgr-name***

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

#### **COMMENT( *string* )**

Specifies a comment that is reproduced in the trace output record (except in the resident trace tables). It can be used to record why the command was issued.

*string* is any character string. It must be enclosed in single quotation marks if it includes a blank, comma, or special character.

### **Destination block**

#### **DEST**

Specifies where the trace output is to be recorded. More than one value can be specified, but do not use the same value twice.

The meaning of each value is as follows:

#### **GTF**

The z/OS Generalized Trace Facility (GTF). If used, the GTF must be started and accepting user (USR) records before the START TRACE command is issued.

#### **RES**

A wrap-around table residing in the ECSA, or a data space for CHINIT.

#### **SMF**

The System Management Facility (SMF). If used, the SMF must be functioning before the START TRACE command is issued. The SMF record numbers used by IBM MQ are 115 and 116. For SMF record type 115, subtypes 1, 2, and 215 are provided for the performance statistics trace.

#### **SRV**

A serviceability routine reserved for IBM use only; not for general use.

**Note:** If your IBM support center need you to use this destination for your trace data they will supply you with module CSQWVSER. If you try to use destination SRV without CSQWVSER an error message is produced at the IBM console when you issue the START TRACE command.

Allowed values, and the default value, depend on the type of trace started, as shown in the following table:

Type	GTF	RES	SMF	SRV
GLOBAL	Allowed	Default	No	Allowed
STAT	No	No	Default	Allowed
ACCTG	Allowed	No	Default	Allowed
CHINIT	No	Default	No	Allowed

## Constraint block

The constraint block places optional constraints on the kinds of data collected by the trace. The allowed constraints depend on the type of trace started, as shown in the following table:


Type	CLASS	IFCID	RMID	USERID
GLOBAL	Allowed	Allowed	Allowed	Allowed
STAT	Allowed	No	No	No
ACCTG	Allowed	No	No	No
CHINIT	Allowed	Allowed	No	No

### CLASS

Introduces a list of classes of data gathered. The classes allowed, and their meaning, depend on the type of trace started:

**(\*)**

For GLOBAL and CHINIT traces, starts traces for all classes of data.

 For ACCTG and STAT traces, starts traces for classes 1 to 3. Channel initiator statistics and channel accounting data are not started with CLASS(\*), and must be started with CLASS(4). Queue statistics are not started with CLASS(\*), and must be started with CLASS(5).



**Attention:** You can specify a comma-separated list of classes, for example TRACE(ACCTG) CLASS(01,03,04); there is no CLASS2. To stop these classes you have started, you must specify CLASS(01,03,04) on the STOP command. That is, you must specify the full range of classes that are active on the STOP command before you can restart the classes you require.

**(integer)**

Any number in the class column of the table that follows. You can use more than one of the classes that are allowed for the type of trace started. A range of classes can be specified as *m:n* (for example, CLASS(01:03)). If you do not specify a class, the default is to start class 1, except when you are using the **START TRACE (STAT)** command with no class where the default is to start class 1 and 2.

Class	Description
	<b>Global trace</b>
01	Reserved for IBM service
02	User parameter error detected in a control block
03	User parameter error detected on entry to MQI



<i>Table 184. Descriptions of trace events and classes (continued)</i>	
<b>Class</b>	<b>Description</b>
	User parameter error detected on exit from MQI
	User parameter error detected in a control block
04	Reserved for IBM service
	<b>Statistics trace</b>
01	Subsystem statistics
	Queue manager statistics
02	Queue manager storage summary statistics. Class 1 statistics must also be enabled to collect this class of data.
03	Queue manager storage detail summary. Class 1 statistics must also be enabled to collect this class of data.
04	Channel initiator statistics
V 9.3.0 V 9.3.0 05	Queue statistics
	<b>Accounting trace</b>
01	The processor time spent processing MQI calls and a count of MQPUT, MQPUT1 and MQGET calls
03	Enhanced accounting and statistical data
04	Channel accounting data
	<b>CHINIT trace</b>
01	Reserved for IBM service
04	Reserved for IBM service

**IFCID**

Reserved for IBM service.

**RMID**

Introduces a list of specific resource managers for which trace information is gathered. You cannot use this option for STAT, ACCTG, or CHINIT traces.

**(\*)**

Starts a trace for all resource managers.

This is the default.

**( integer )**

The identifying number of any resource manager in the following table. You can use up to 8 of the allowed resource manager identifiers; do not use the same one twice.

<i>Table 185. Resource Manager identifiers that are allowed</i>	
<b>RMID</b>	<b>Resource manager</b>
1	Initialization procedures
2	Agent services management

<i>Table 185. Resource Manager identifiers that are allowed (continued)</i>	
<b>RMID</b>	<b>Resource manager</b>
3	Recovery management
4	Recovery log management
6	Storage management
7	Subsystem support for allied memories
8	Subsystem support for subsystem interface (SSI) functions
12	System parameter management
16	Instrumentation commands, trace, and dump services
23	General command processing
24	Message generator
26	Instrumentation accounting and statistics
148	Connection manager
163	Topic Manager
197	CF manager
199	Functional recovery
200	Security management
201	Data management
211	Lock management
212	Message management
213	Command server
215	Buffer management
242	IBM MQ IMS - bridge
245	Db2 manager

#### **TDATA**

Reserved for IBM service.

#### **USERID**

Introduces a list of specific user IDs for which trace information is gathered. You cannot use this option for STAT, ACCTG, or CHINIT traces.

#### **(\*)**

Starts a trace for all user IDs. This is the default.

#### **(userid)**

Names a user ID. You can use up to 8 user IDs; a separate trace is started for each. The user ID is the primary authorization ID of the task, used by IBM MQ inside the queue manager. This is the userid displayed by the MQSC command DISPLAY CONN.

#### **Related tasks**

[Tracing on z/OS](#)

## STOP CHANNEL (stop a channel)

Use the MQSC command **STOP CHANNEL** to stop a channel.

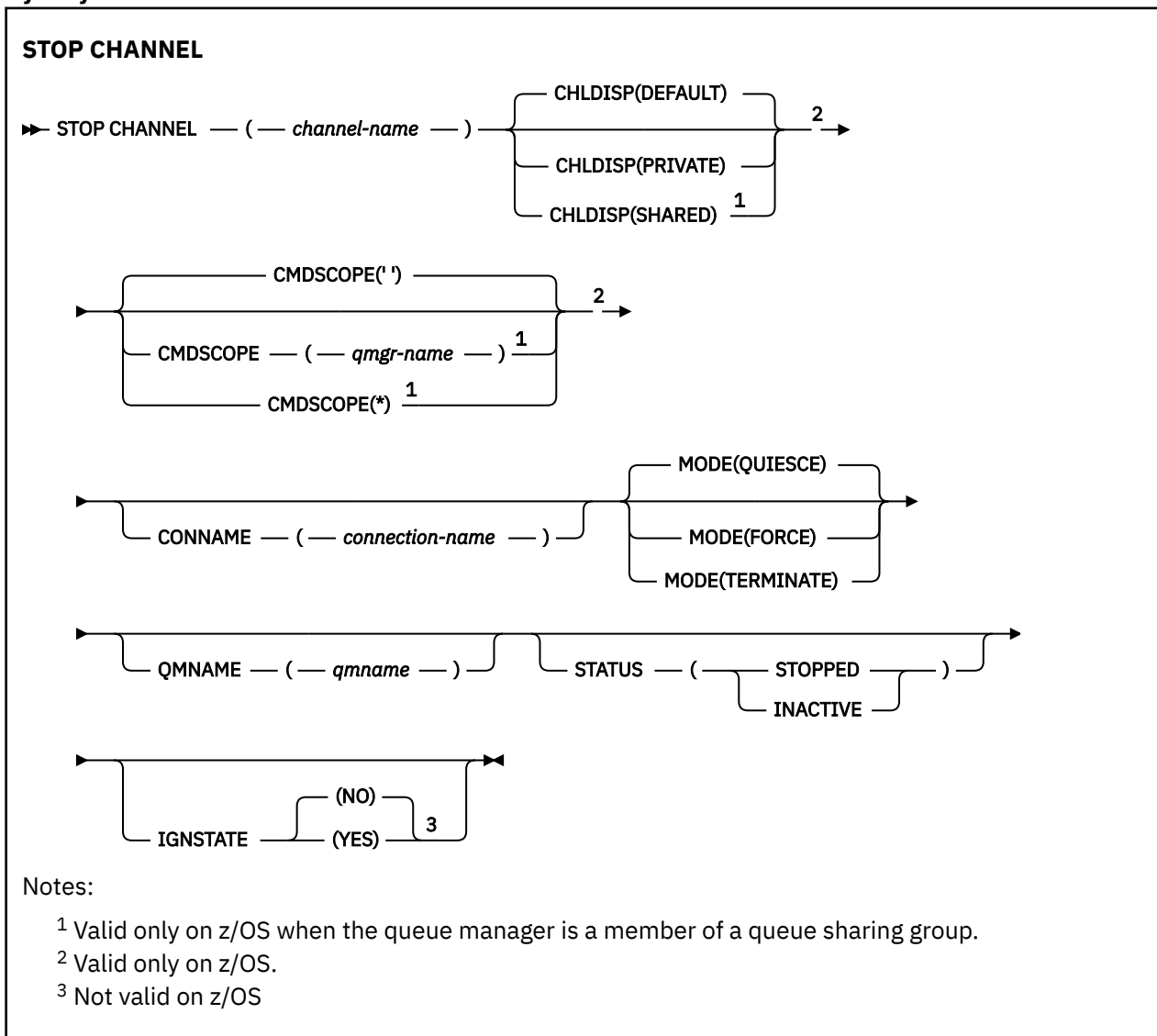
### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

**z/OS** You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).


- [Syntax diagram](#)
- [“Usage notes for STOP CHANNEL” on page 963](#)
- [“Parameter descriptions for STOP CHANNEL” on page 964](#)

**Synonym:** STOP CHL



### Usage notes for STOP CHANNEL

1. If you specify either **QMNAME** or **CONNAME**, **STATUS** must either be **INACTIVE** or not specified. Do not specify a **QMNAME** or **CONNAME** and **STATUS (STOPPED)**. It is not possible to have a channel stopped for one partner but not for others. This sort of function can be provided by a channel security exit. For more information about channel exits, see [Channel exit programs](#).

2.  On z/OS, the command server and the channel initiator must be running.
3. Any channels in STOPPED state need to be started manually; they are not started automatically. See [Restarting stopped channels](#) for information about restarting stopped channels.
4. This command can be issued to a channel of any type except CLNTCONN channels (including those that have been defined automatically).
5. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager repository.

## Parameter descriptions for STOP CHANNEL

### *(channel-name)*

The name of the channel to be stopped. This parameter is required for all channel types.

### **CHLDISP**

This parameter applies to z/OS only and can take the values of:

- DEFAULT
- PRIVATE
- SHARED

If this parameter is omitted, then the DEFAULT value applies. This is taken from the default channel disposition attribute, **DEFCDISP**, of the channel object.

In conjunction with the various values of the **CMDSCOPE** parameter, this parameter controls two types of channel:

#### **SHARED**

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of SHARED.

#### **PRIVATE**

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than SHARED.

**Note:** This disposition is not related to the disposition set by the disposition of the queue sharing group of the channel definition.

The combination of the **CHLDISP** and **CMDSCOPE** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On every active queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of **CHLDISP** and **CMDSCOPE** are summarized in the following table:

<i>Table 186. CHLDISP and CMDSCOPE for STOP CHANNEL</i>			
<b>CHLDISP</b>	<b>CMDSCOPE() or CMDSCOPE (local- qmgr)</b>	<b>CMDSCOPE (qmgr- name)</b>	<b>CMDSCOPE(*)</b>
PRIVATE	Stop as a private channel on the local queue manager.	Stop as a private channel on the named queue manager	Stop as a private channel on all active queue managers
SHARED	<p>For RCVR and SVRCONN channels, stop as shared channel on all active queue managers.</p> <p>For SDR, RQSTR, and SVR channels, stop as a shared channel on the queue manager where it is running. If the channel is in an inactive state (not running), or if it is in RETRY state because the channel initiator on which it was running has stopped, a STOP request for the channel is issued on the local queue manager.</p> <p>This might automatically generate a command using <b>CMDSCOPE</b> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted	Not permitted

**z/OS** **CMDSCOPE**

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

If **CHLDISP** is set to SHARED, **CMDSCOPE** must be blank or the local queue manager.

''

The command runs on the queue manager on which it was entered. This is the default value.

**qmgr-name**

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## CONNNAME (*connection-name*)

Connection name. Only channels matching the specified connection name are stopped.

When issuing the **STOP CHANNEL** command using a **CONNNAME** parameter, ensure that the value specified in the **CONNNAME** parameter is exactly as shown in [“DISPLAY CHSTATUS \(display channel status\)” on page 685](#).

## MODE

Specifies whether the current batch is allowed to finish in a controlled manner. This parameter is optional.

### QUIESCE

This is the default.

**Multi** On [Multiplatforms](#), allows the current batch to finish processing.

**z/OS** On z/OS, the channel stops after the current message has finished processing. (The batch is then ended and no more messages are sent, even if there are messages waiting on the transmission queue.)

For a receiving channel, if there is no batch in progress, the channel waits for either of the following to take place before it stops:

- The next batch to start
- The next heartbeat (if heartbeats are being used)

For server-connection channels, allows the current connection to end.

If you issue a **STOP CHANNEL *channelname* MODE (QUIESCE)** command on a server-connection channel, the IBM MQ client infrastructure becomes aware of the stop request in a timely manner. This time is dependent upon the speed of the network.

If a client application is using the server-connection channel and is performing either of the following operations at the time that the command is issued, then the MQPUT or MQGET operation fails:

- An MQPUT operation with the PMO option MQPMO\_FAIL\_IF QUIESCING set.
- An MQGET operation with the GMO option MQGMO\_FAIL\_IF QUIESCING set.

The client application receives reason code MQRC\_CONNECTION QUIESCING.

If a client application is using the server-connection channel and is performing either of the following operations, then the client application is allowed to complete the MQPUT or MQGET operation:

- An MQPUT operation without the PMO option MQPMO\_FAIL\_IF QUIESCING set.
- An MQGET operation without the GMO option MQGMO\_FAIL\_IF QUIESCING set.

Any subsequent FAIL\_IF QUIESCING calls using this connection fail with MQRC\_CONNECTION QUIESCING. Calls which do not specify FAIL\_IF QUIESCING, are usually permitted to complete, although the application should complete such operations in a timely manner, to permit the channel to end.

If the client application is not performing an MQ API call when the server-connection channel is stopped, it becomes aware of the stop request as a result of issuing a subsequent call to IBM MQ and receives return code MQRC\_CONNECTION QUIESCING.

After sending the MQRC\_CONNECTION QUIESCING return code to the client, and allowing any outstanding MQPUT or MQGET operations to complete if necessary, the server ends the client connections for the server-connection channel.

Due to the imprecise timing of network operations, the client application should not attempt further MQ API operations.

## FORCE

For server-connection channels, breaks the current connection, returning MQRC\_CONNECTION\_QUIESCING or MQRC\_CONNECTION\_BROKEN. For other channel types, terminates transmission of any current batch. This is likely to result in in-doubt situations.

**z/OS** On IBM MQ for z/OS, specifying **FORCE** interrupts any message reallocation in progress, which might leave BIND\_NOT\_FIXED messages partially reallocated or out of order.

## TERMINATE

**z/OS** On z/OS, **TERMINATE** is synonymous with **FORCE**.

**Multi** On other platforms, **TERMINATE** terminates transmission of any current batch.

This allows the command to actually terminate the channel thread or process.

For server-connection channels, **TERMINATE** breaks the current connection, returning MQRC\_CONNECTION\_QUIESCING or MQRC\_CONNECTION\_BROKEN. Using **TERMINATE** can cause unpredictable results to occur.

**z/OS** On z/OS, specifying **TERMINATE** interrupts any message reallocation in progress, which might leave BIND\_NOT\_FIXED messages partially reallocated or out of order.

## QMNAME (*qmname*)

Queue manager name. Only channels matching the specified remote queue manager are stopped.

## STATUS

Specifies the new state of any channels stopped by this command. For more information about channels in STOPPED state, especially SVRCONN channels on z/OS, see [Restarting stopped channels](#).

### STOPPED

The channel is stopped. For a sender or server channel the transmission queue is set to **GET (DISABLED)** and NOTRIGGER.

This is the default if **QMNAME** or **CONNAME** are not specified.

### INACTIVE

The channel is inactive.

This is the default if **QMNAME** or **CONNAME** are specified.

## **Multi** IGNSTATE

Specifies whether the command fails if the channel is already stopped. The possible values are:

### NO

The command fails if the channel is already stopped. This is the default value.

### YES

The command succeeds regardless of the current state of the channel.

## **Windows** **Linux** **AIX** **STOP CHANNEL (stop a channel) MQTT**

Use the MQSC command STOP CHANNEL to stop an MQ Telemetry channel.

## Using MQSC commands

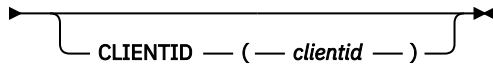
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

The STOP CHANNEL (MQTT) command is only valid for MQ Telemetry channels.

**Synonym:** STOP CHL

## STOP CHANNEL

➤ STOP CHANNEL — ( — *channel-name* — ) — CHLTYPE — ( — MQTT — ) →



## Usage notes for STOP CHANNEL

1. Any channels in STOPPED state need to be started manually; they are not started automatically.

## Parameter descriptions for STOP CHANNEL

### (*channel-name*)

The name of the channel to be stopped. This parameter is required for all channel types including MQTT channels.

### CHLTYPE

Channel type. The value must be MQTT.

### CLIENTID (*string*)

Client identifier. The client identifier is a 23-byte string that identifies an MQ Telemetry Transport client. When the STOP CHANNEL command specifies a CLIENTID, only the connection for the specified client identifier is stopped. If the CLIENTID is not specified, all the connections on the channel are stopped.

## STOP CHINIT (stop channel initiator) on z/OS

Use the MQSC command STOP CHINIT to stop a channel initiator. The command server must be running.

## Using MQSC commands on z/OS

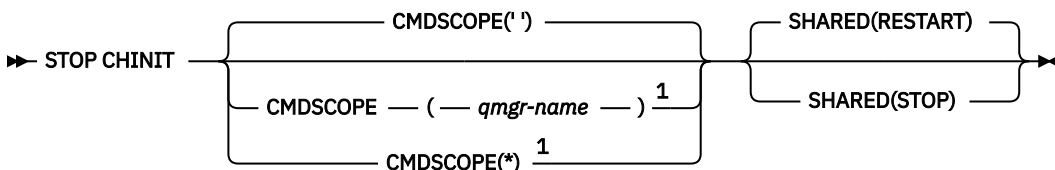
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for STOP CHINIT” on page 969](#)
- [“Parameter descriptions for STOP CHINIT” on page 969](#)

**Synonym:** STOP CHI

## STOP CHINIT



Notes:

- <sup>1</sup> Valid only when the queue manager is a member of a queue sharing group.



## Usage notes for STOP CHINIT

1. When you issue the STOP CHINIT command, IBM MQ stops any channels that are running in the following way:
  - Sender and server channels are stopped using STOP CHANNEL MODE(QUIESCE) STATUS(INACTIVE)
  - All other channels are stopped using STOP CHANNEL MODE(FORCE)See [“STOP CHANNEL \(stop a channel\)”](#) on page 963 for information about what this involves.
2. You might receive communications-error messages as a result of issuing the STOP CHINIT command.

## Parameter descriptions for STOP CHINIT

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

''

The command runs on the queue manager on which it was entered. This is the default value.

#### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### SHARED

Specifies whether the channel initiator should attempt to restart any active sending channels, started with CHLDISP(SHARED), that it owns on another queue manager. The possible values are:

#### RESTART

Shared sending channels are to be restarted. This is the default.

#### STOP

Shared sending channels are not to be restarted, so will become inactive.

(Active channels started with CHLDISP(FIXSHARED) are not restarted, and always become inactive.)

## **STOP CMDSERV (stop the command server) on z/OS**

Use the MQSC command STOP CMDSERV to stop the command server.

## Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12C. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Usage notes for STOP CMDSERV”](#) on page 970

**Synonym:** STOP CS

## STOP CMDSERV

▶▶ STOP CMDSERV ◀◀

### Usage notes for STOP CMDSERV

1. STOP CMDSERV stops the command server from processing commands in the system-command input queue (SYSTEM.COMMAND.INPUT), mover commands, and commands using CMDSCOPE.
2. If this command is issued through the initialization files or through the operator console before work is released to the queue manager (that is, before the command server is started automatically), it prevents the command server from starting automatically and puts it into a DISABLED state. It overrides an earlier START CMDSERV command.
3. If this command is issued through the operator console or the command server while the command server is in a RUNNING state, it stops the command server when it has finished processing its current command. When this happens, the command server enters the STOPPED state.
4. If this command is issued through the operator console while the command server is in a WAITING state, it stops the command server immediately. When this happens, the command server enters the STOPPED state.
5. If this command is issued while the command server is in a DISABLED or STOPPED state, no action is taken, the command server remains in its current state, and an error message is returned to the command originator.

Multi

## STOP CONN (stop a connection) on Multiplatforms

Use the MQSC command STOP CONN to break a connection between an application and the queue manager.

### Using MQSC commands

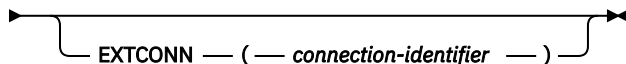
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Usage notes” on page 970](#)
- [“Parameter descriptions for STOP CONN” on page 970](#)

**Synonym:** STOP CONN

## STOP CONN

▶▶ STOP CONN — ( — *connection-identifier* — ) →



### Usage notes

There might be circumstances in which the queue manager cannot implement this command when the success of this command cannot be guaranteed.

### Parameter descriptions for STOP CONN

( *connection-identifier* )

The identifier of the connection definition for the connection to be broken.

When an application connects to IBM MQ, it is given a unique 24-byte connection identifier (ConnectionId). The value of CONN is formed by converting the last eight bytes of the ConnectionId to its 16-character hexadecimal equivalent.

### EXTCONN

The value of EXTCONN is based on the first sixteen bytes of the ConnectionId converted to its 32-character hexadecimal equivalent.

Connections are identified by a 24-byte connection identifier. The connection identifier comprises a prefix, which identifies the queue manager, and a suffix which identifies the connection to that queue manager. By default, the prefix is for the queue manager currently being administered, but you can specify a prefix explicitly by using the EXTCONN parameter. Use the CONN parameter to specify the suffix.

When connection identifiers are obtained from other sources, specify the fully qualified connection identifier (both EXTCONN and CONN) to avoid possible problems related to non-unique CONN values.

### Related reference

“DISPLAY CONN (display application connection information)” on page 725

Use the MQSC command **DISPLAY CONN** to display connection information about the applications connected to the queue manager. This is a useful command because it enables you to identify applications with long-running units of work.

## STOP LISTENER (stop a channel listener)

Use the MQSC command STOP LISTENER to stop a channel listener.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

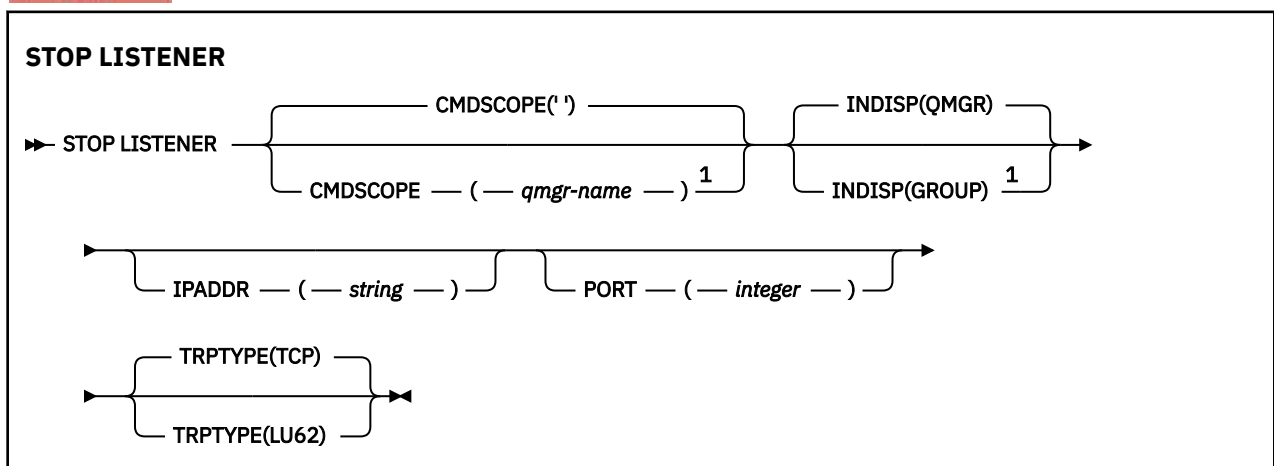
**z/OS** You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- **z/OS** [Syntax diagram for IBM MQ for z/OS](#)
- [Syntax diagram for IBM MQ on other platforms](#)
- **z/OS** [“Usage notes” on page 972](#)
- [“Parameter descriptions for STOP LISTENER” on page 972](#)

**Synonym:** STOP LSTR

### z/OS

**z/OS**

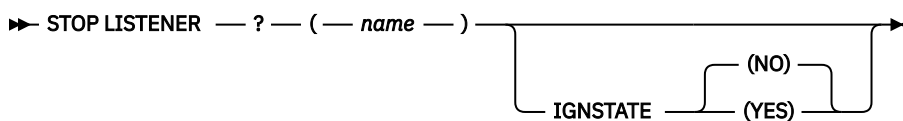


## Notes:

<sup>1</sup> Valid only on z/OS when the queue manager is a member of a queue sharing group.

## Other platforms

### STOP LISTENER



## Usage notes

The listener stops in quiesce mode (it disregards any further requests).

**z/OS** On z/OS:

- The command server and the channel initiator must be running.
- If a listener is listening on multiple addresses or ports, only the address and port combinations with the address, or port, specified are stopped.
- If a listener is listening on all addresses for a particular port, a stop request for a specific IPADDR with the same port fails.
- If neither an address nor a port is specified, all addresses and ports are stopped and the listener task ends.

## Parameter descriptions for STOP LISTENER

### (*name*)

Name of the listener to be stopped. If you specify this parameter, you cannot specify any other parameters.

This parameter is required on all platforms **z/OS** other than z/OS where it is not a supported parameter.

### **z/OS** CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

This parameter is valid only on z/OS.

### **z/OS** INDISP

Specifies the disposition of the inbound transmissions that the listener handles. The possible values are:

#### **QMGR**

Handling for transmissions directed to the queue manager. This is the default.

## GROUP

Handling for transmissions directed to the queue sharing group. This is allowed only if there is a shared queue manager environment.

This parameter is valid only on z/OS.

## z/OS IPADDR

IP address for TCP/IP specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric form. This is valid only if the transmission protocol (TRPTYPE) is TCP/IP.

This parameter is valid only on z/OS.

## z/OS PORT

The port number for TCP/IP. This is the port number on which the listener is to stop listening. This is valid only if the transmission protocol is TCP/IP.

This parameter is valid only on z/OS.

## z/OS TRPTYPE

Transmission protocol used. This is optional.

### TCP

TCP. This is the default if TRPTYPE is not specified.

### LU62

SNA LU 6.2.

This parameter is valid only on z/OS.

## Multi IGNSTATE

Specifies whether the command fails if the listener is already stopped. The possible values are:

### NO

The command fails if the listener is already stopped. This is the default value.

### YES

The command succeeds regardless of the current state of the listener.

## z/OS STOP QMGR (stop queue manager) on z/OS

Use the MQSC command STOP QMGR to stop the queue manager.

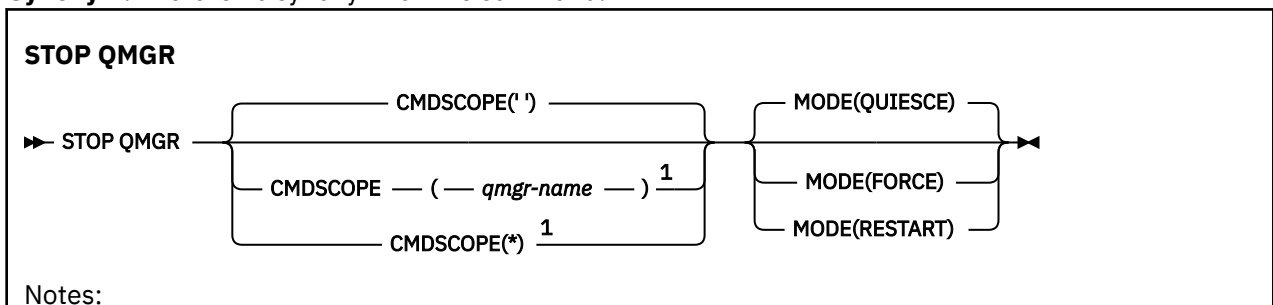
## Using MQSC commands on z/OS

For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for STOP QMGR” on page 974](#)

**Synonym:** There is no synonym for this command.



<sup>1</sup> Valid only when the queue manager is a member of a queue sharing group.

## Parameter descriptions for STOP QMGR

The parameters are optional.

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

### MODE

Specifies whether programs currently being executed are allowed to finish.

### QUIESCE

Allows programs currently being executed to finish processing. No new program is allowed to start. This is the default.

This option means that all connections to other address spaces must terminate before the queue manager stops. The system operator can determine whether any connections remain by using the DISPLAY CONN command, and can cancel remaining connections using z/OS commands.

This option deregisters IBM MQ from the z/OS automatic restart manager (ARM).

### FORCE

Terminates programs currently being executed, including utilities. No new program is allowed to start. This option might cause in-doubt situations.

This option might not work if all the active logs are full, and log archiving has not occurred. In this situation you must issue the z/OS command CANCEL to terminate.

This option deregisters IBM MQ from the z/OS automatic restart manager (ARM).

### RESTART

Terminates programs currently being executed, including utilities. No new program is allowed to start. This option might cause in-doubt situations.

This option might not work if all the active logs are full, and log archiving has not occurred. In this situation you must issue the z/OS command CANCEL to terminate.

This option does not deregister IBM MQ from ARM, so the queue manager is eligible for immediate automatic restart.

## STOP SERVICE (stop a service) on Multiplatforms

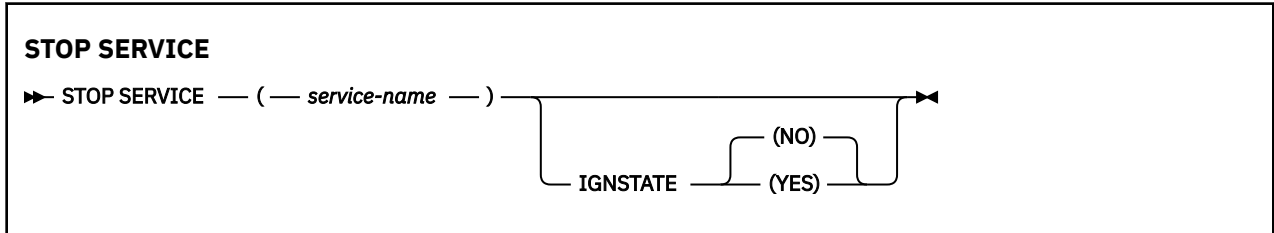
Use the MQSC command **STOP SERVICE** to stop a service.

### Using MQSC commands

For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

- [Syntax diagram](#)
- [“Usage notes” on page 975](#)
- [“Parameter descriptions for STOP SERVICE” on page 975](#)

#### Synonym:



### Usage notes

If the service is running, it is requested to stop. This command is processed asynchronously so might return before the service has stopped.

If the service that is requested to stop has no STOP command defined, an error is returned.

### Parameter descriptions for STOP SERVICE

#### (*service-name*)

The name of the service definition to be stopped. This is required. The name must that of an existing service on this queue manager.

#### IGNSTATE

Specifies whether the command fails if the service is already stopped. The possible values are:

##### NO

The command fails if the service is already stopped. This is the default value.

##### YES

The command succeeds regardless of the current state of the service.

### Related concepts

[Working with services](#)

### Related tasks

[Managing services](#)

[Using a server service object](#)

[Using a command service object](#)

### Related reference

[“ALTER SERVICE \(alter a service definition\) on Multiplatforms” on page 415](#)

Use the MQSC command **ALTER SERVICE** to alter the parameters of an existing IBM MQ service definition.

[“START SERVICE \(start a service\) on Multiplatforms” on page 955](#)

Use the MQSC command **START SERVICE** to start a service. The identified service definition is started within the queue manager and inherits the environment and security variables of the queue manager.

## **z/OS** STOP SMDSCONN (stop shared message data sets connection) on z/OS

Use the MQSC command STOP SMDSCONN to terminate the connection from this queue manager to one or more specified shared message data sets (causing them to be closed and deallocated) and to mark the connection as STOPPED.

### Using MQSC commands on z/OS

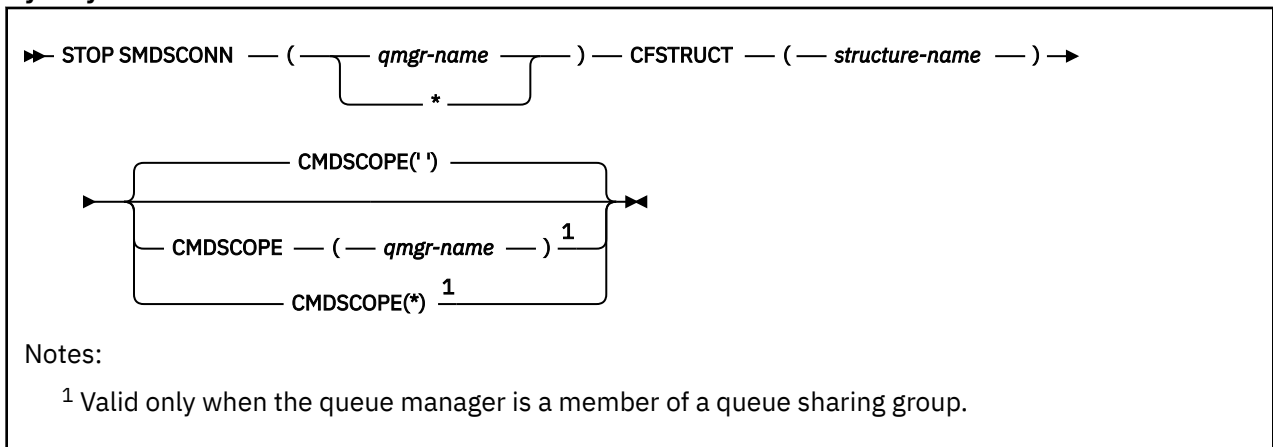
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 2CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [“Syntax diagram for STOP SMDSCONN” on page 976](#)
- [“Parameter descriptions for STOP SMDSCONN” on page 976](#)

### Syntax diagram for STOP SMDSCONN

Synonym:



### Parameter descriptions for STOP SMDSCONN

#### SMDSCONN

Specify the queue manager which owns the shared message data set for which the connection is to be stopped, or an asterisk to stop connections to all shared message data sets associated with the specified structure.

#### CFSTRUCT

Specify the structure name for which shared message data set connections are to be stopped.

#### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

#### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.



You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

\*

The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group. The effect of this is the same as entering the command on every queue manager in the queue sharing group.

## STOP TRACE (stop trace) on z/OS

Use the MQSC command STOP TRACE to stop tracing.

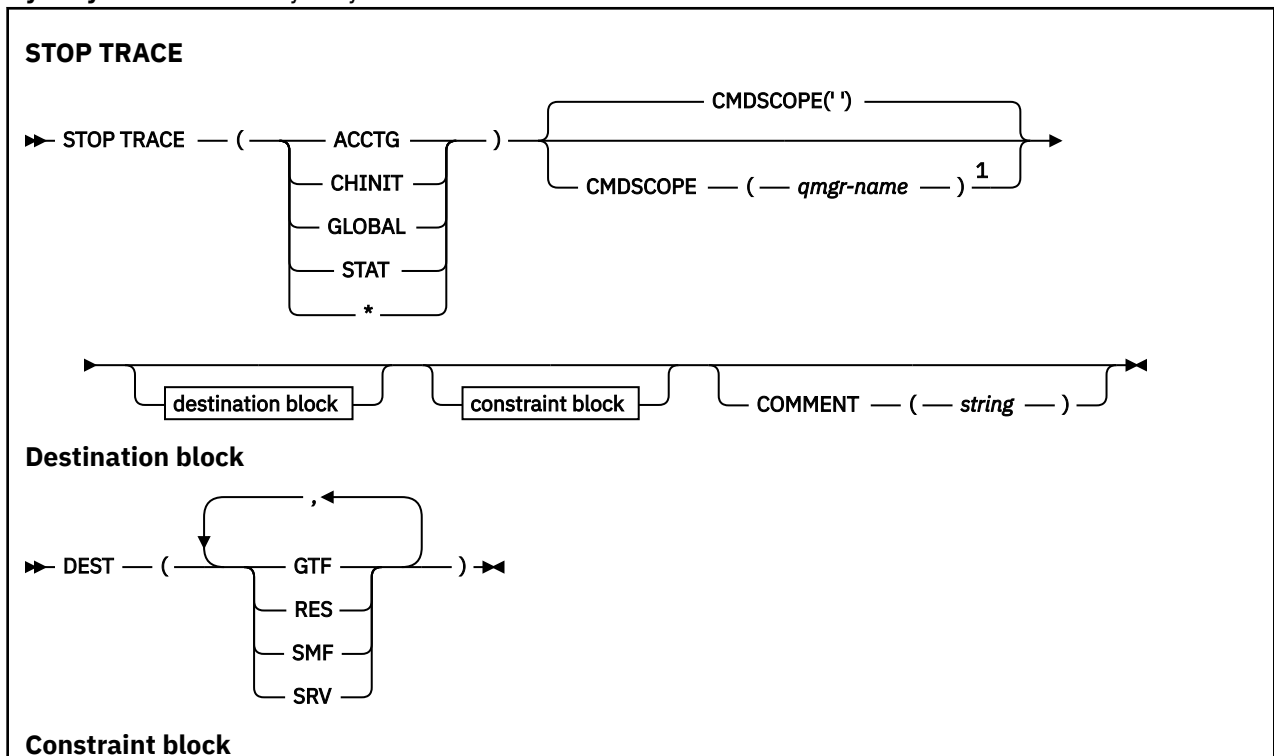
### Using MQSC commands on z/OS

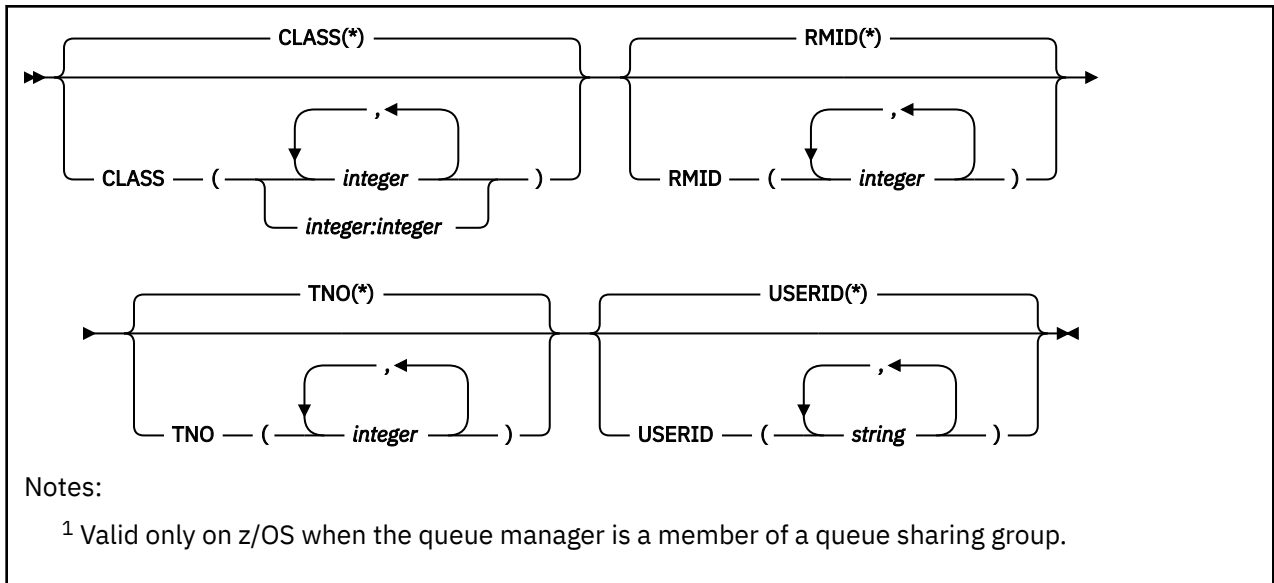
For information on how you use MQSC commands on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

You can issue this command from sources 12CR. For an explanation of the source symbols, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

- [Syntax diagram](#)
- [“Parameter descriptions for STOP TRACE” on page 978](#)
- [“Destination block” on page 979](#)
- [“Constraint block” on page 979](#)

**Synonym:** There is no synonym for this command.





## Parameter descriptions for STOP TRACE

Each option that you use limits the effect of the command to active traces that were started using the same option, either explicitly or by default, with exactly the same parameter values.

You must specify a trace type or an asterisk. STOP TRACE(\*) stops all active traces.

The trace types are:

### ACCTG

Accounting data (the synonym is A)

**Note:** Accounting data can be lost if the accounting trace is started or stopped while applications are running. For information about the conditions that must be satisfied for successful collection of accounting data, see [Using IBM MQ trace](#).

### CHINIT

Service data from the channel initiator. The synonym is CHI or DQM.

If the only trace running on the CHINIT is the one started automatically when the CHINIT was started, that tracing can be stopped only by explicitly stating the TNO for the default CHINIT trace (0). For example: STOP TRACE(CHINIT) TNO(0)

### GLOBAL

Service data from the entire queue manager except for the channel initiator. The synonym is G.

### STAT

Statistical data (the synonym is S)

\*

All active traces

### CMDSCOPE

This parameter specifies how the command runs when the queue manager is a member of a queue sharing group.

CMDSCOPE cannot be used for commands issued from the first initialization input data set CSQINP1.

..

The command runs on the queue manager on which it was entered. This is the default value.

### qmgr-name

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

### **COMMENT( *string* )**

Specifies a comment that is reproduced in the trace output record (except in the resident trace tables), and can be used to record why the command was issued.

*string* is any character string. It must be enclosed in single quotation marks if it includes a blank, comma, or special character.

## **Destination block**

### **DEST**

Limits the action to traces started for particular destinations. More than one value can be specified, but do not use the same value twice. If no value is specified, the list is not limited.

Possible values and their meanings are:

#### **GTF**

The Generalized Trace Facility

#### **RES**

A wrap-around table residing in the ECSA

#### **SMF**

The System Management Facility

#### **SRV**

A serviceability routine designed for problem diagnosis

## **Constraint block**

### **CLASS( *integer* )**

Limits the command to traces started for particular classes. See the START TRACE command for a list of allowed classes. A range of classes can be specified as *m:n* (for example, CLASS(01:03)). You cannot specify a class if you did not specify a trace type.

The default is CLASS(\*), which does not limit the command.



**Attention:** You can specify a comma-separated list of classes, for example TRACE(ACCTG) CLASS(01,03,04); there is no CLASS2. To stop these classes you have started, you must specify CLASS(01,03,04) on the STOP command. That is, you must specify the full range of classes that are active on the STOP command before you can restart the classes you require.

### **RMID( *integer* )**

Limits the command to traces started for particular resource managers. See the START TRACE command for a list of allowed resource manager identifiers.

Do not use this option with the STAT, ACCTG, or CHINIT trace type.

The default is RMID(\*), which does not limit the command.

### **TNO( *integer* )**

Limits the command to particular traces, identified by their trace number (0 to 32). Up to 8 trace numbers can be used. If more than one number is used, only one value for USERID can be used.

0 is the trace that the channel initiator can start automatically. Traces 1 to 32 are those for queue manager or the channel initiator that can be started automatically by the queue manager, or manually, using the START TRACE command.

The default is TNO(\*), which applies the command to all active traces with numbers 1 to 32, but **not** to the 0 trace. You can stop trace number 0 only by specifying it explicitly.

## USERID( *string* )

Limits the action of the STOP TRACE to traces started for particular user ID. Up to 8 user IDs can be used. If more than one user ID is used, only one value can be used for TNO. Do not use this option with the STAT, ACCTG, or CHINIT trace type.

The default is USERID(\*), which does not limit the command.

## SUSPEND QMGR (suspend a cluster queue manager)

Use the MQSC command **SUSPEND QMGR** to advise other queue managers in a cluster to avoid sending messages to the local queue manager if possible.

### Using MQSC commands

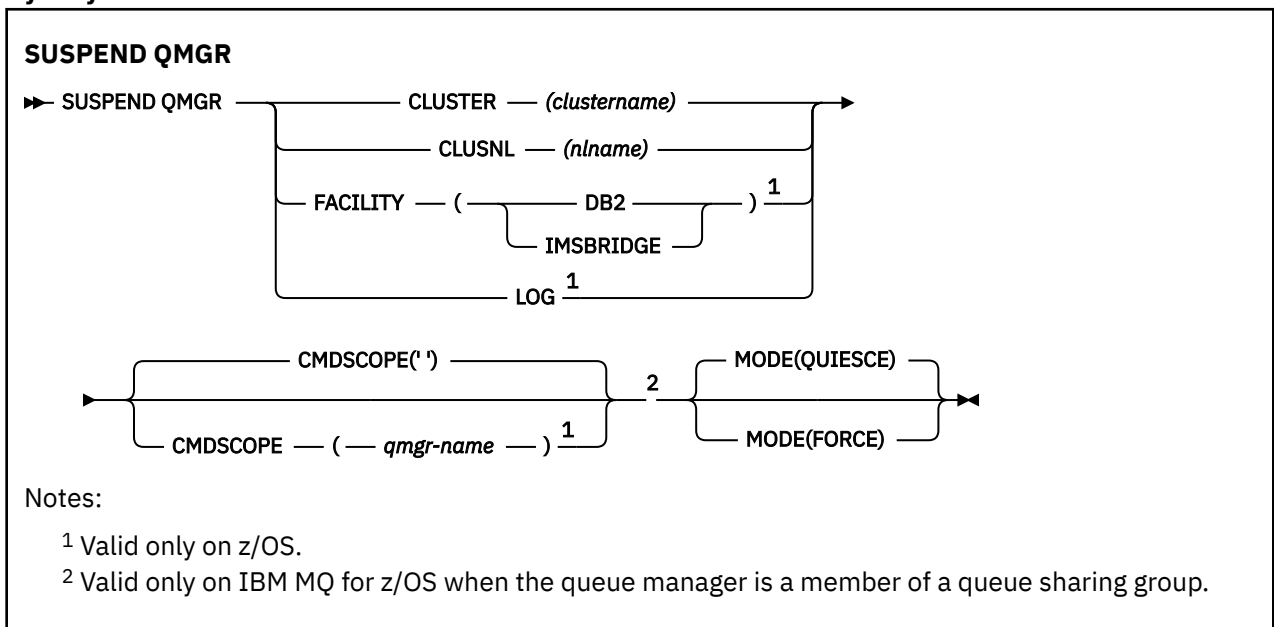
For information on how you use MQSC commands, see [Administering IBM MQ using MQSC commands](#).

For further details about using the **SUSPEND QMGR** and **RESUME QMGR** commands to remove a queue manager from a cluster temporarily, see [SUSPEND QMGR, RESUME QMGR and clusters](#).

**z/OS** On z/OS this command can also be used to suspend logging and update activity for the queue manager until a subsequent **RESUME QMGR** command is issued. Its action can be reversed by the **RESUME QMGR** command. This command does not mean that the queue manager is disabled.

- [Syntax diagram](#)
- **z/OS** See [“Using SUSPEND QMGR on z/OS”](#) on page 980
- **z/OS** [“Usage notes”](#) on page 981
- [“Parameter descriptions for SUSPEND QMGR”](#) on page 981

**Synonym:** None



### Using SUSPEND QMGR on z/OS

**z/OS**

**SUSPEND QMGR** can be used on z/OS. Depending on the parameters used on the command, it may be issued from various sources. For an explanation of the symbols in this table, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

Table 187. SUSPEND QMGR command and command sources

Command	Command Sources	Notes
SUSPEND QMGR CLUSTER/CLUSNL	CR	Ensure the channel initiator is running
SUSPEND QMGR FACILITY	CR	
SUSPEND QMGR LOG	C	

## Usage notes

**z/OS** On z/OS:

- If you define **CLUSTER** or **CLUSNL**, be aware of the following behavior:
  - The command fails if the channel initiator has not been started.
  - Any errors are reported to the system console where the channel initiator is running; they are not reported to the system that issued the command.
- The **SUSPEND QMGR** and **RESUME QMGR** commands are supported through the console only. However, all the other **SUSPEND** and **RESUME** commands are supported through the console and command server.

## Parameter descriptions for SUSPEND QMGR

The **SUSPEND QMGR** with the **CLUSTER** or **CLUSNL** parameters to specify the cluster or clusters for which availability is suspended, how the suspension takes effect.

**z/OS** On z/OS, controls logging and update activity and how the command runs when the queue manager is a member of a queue sharing group.

You can use the **SUSPEND QMGR FACILITY(Db2)** command to terminate the queue manager connection to Db2. This command might be useful if you want to apply service to Db2. Be aware, if you use this option then there is no access to Db2 resources, for example, large messages which might be offloaded to Db2 from a coupling facility.

**z/OS** You can use the **SUSPEND QMGR FACILITY(IMSBRIDGE)** command to stop sending messages from the IBM MQ IMS bridge to IMS OTMA. **z/OS** For more information about controlling message delivery to shared and non-shared queues, see [Controlling the IMS bridge](#).

### **CLUSTER (clustername)**

The name of the cluster for which availability is to be suspended.

### **CLUSNL (nname)**

The name of the namelist that specifies a list of clusters for which availability is to be suspended.

### **z/OS FACILITY**

Specifies the facility to which connection is to be terminated. The parameter must have one of the following values:

#### **Db2**

Causes the existing connection to Db2 to be terminated. The connection is re-established when the **RESUME QMGR** command is issued. When the Db2 connection is **SUSPENDED**, any API requests which must access Db2 to complete will be suspended until the **RESUME QMGR FACILITY( Db2)** command is issued. API requests include:

- The first MQOPEN of a shared queue since the queue manager started
- MQPUT, MQPUT1 and MQGET to or from a shared queue where the message payload has been offloaded to Db2

z/OS

## IMSBRIDGE

Stops the sending of messages from IMS bridge queues to OTMA. The IMS connection is not affected. When the tasks that transmit messages to IMS have been terminated, no further messages are sent to IMS until one of the following actions happens:

- OTMA or IMS is stopped and restarted
- IBM MQ is stopped and restarted
- A **RESUME QMGR** command is processed

Return messages from IMS OTMA to the queue manager are unaffected.

To monitor progress of the command, issue the following command and ensure that none of the queues are open:

```
DIS Q(*) CMDSCOPE(qmgr) STGCLASS(bridge_stgclass) IPPROCS
```

If any queue is open, use **DISPLAY QSTATUS** to verify that the MQ-IMS bridge does not have it open.

This parameter is valid only on z/OS.

z/OS

## LOG

Suspends logging and update activity for the queue manager until a subsequent **RESUME** request is issued. Any unwritten log buffers are externalized, a system checkpoint is taken (non-data sharing environment only), and the BSDS is updated with the high-written RBA before the update activity is suspended. A highlighted message (CSQJ372I) is issued and remains on the system console until update activity has been resumed. Valid on z/OS only. If **LOG** is specified, the command can be issued only from the z/OS system console.

This option is not permitted when a system quiesce is active by either the **ARCHIVE LOG** or **STOP QMGR** command.

Update activity remains suspended until a **RESUME QMGR LOG** or **STOP QMGR** command is issued.

This command must not be used during periods of high activity, or for long periods of time. Suspending update activity can cause timing-related events such as lock timeouts or IBM MQ diagnostic memory dumps when delays are detected.

z/OS

## CMDSCOPE

This parameter applies to z/OS only and specifies how the command runs when the queue manager is a member of a queue sharing group.

..

The command runs on the queue manager on which it was entered. This is the default value.

### *qmgr-name*

The command runs on the queue manager you specify, providing the queue manager is active within the queue sharing group.

You can specify a queue manager name, other than the queue manager on which the command was entered, only if you are using a queue sharing group environment and if the command server is enabled.

## MODE

Specifies how the suspension of availability is to take effect:

### QUIESCE

Other queue managers in the cluster are advised to avoid sending messages to the local queue manager if possible. It does not mean that the queue manager is disabled.

## **FORCE**

All inbound cluster channels from other queue managers in the cluster are stopped forcibly. This occurs only if the queue manager has also been forcibly suspended from all other clusters to which the cluster receiver channel for this cluster belongs.

The **MODE** keyword is permitted only with **CLUSTER** or **CLUSNL**. It is not permitted with the **LOG** or **FACILITY** parameter.

### **Related reference**

[“RESUME QMGR \(resume a cluster queue manager\)” on page 913](#)

Use the MQSC command RESUME QMGR to inform other queue managers in a cluster that the local queue manager is available again for processing and can be sent messages. It reverses the action of the SUSPEND QMGR command.

[SUSPEND QMGR, RESUME QMGR and clusters](#)

## **Programmable command formats (PCFs) reference**

PCFs define command and reply messages that can be exchanged across a network between a program and any queue manager that supports PCFs. This simplifies queue manager administration and other network administration.

For an introduction to PCFs, see [Introduction to Programmable Command Formats](#).

For the full list of PCFs, see [“Definitions of the Programmable Command Formats” on page 983](#).

PCF commands and responses have a consistent structure including of a header and any number of parameter structures of defined types. For information about these structures, see [“Structures for PCF commands and responses” on page 1514](#).

For an example PCF, see [“PCF example” on page 1541](#).

- [“Definitions of the Programmable Command Formats” on page 983](#)
- [“Structures for PCF commands and responses” on page 1514](#)
- [“PCF example” on page 1541](#)

### **Related concepts**

[“IBM MQ control commands reference” on page 21](#)

Reference information about the IBM MQ control commands.

### **Related reference**

[“CL commands for IBM i reference” on page 1551](#)

A list of CL commands for IBM i, grouped according to command type.


[“MQSC commands reference” on page 251](#)

Use MQSC commands to help you manage queue manager objects, including the queue manager itself, queues, process definitions, channels, client connection channels, listeners, services, namelists, clusters, and authentication information objects.

## **Definitions of the Programmable Command Formats**

All the available Programmable Command Formats (PCFs) are listed including their parameters (required and optional), response data and error codes.

Following is the reference information for the Programmable Command Formats (PCFs) of commands and responses sent between an IBM MQ systems management application program and an IBM MQ queue manager.

 [“MQCMD\\_BACKUP\\_CF\\_STRUC \(Backup CF Structure\) on z/OS” on page 1102](#)

[“Change, Copy, and Create Authentication Information Object” on page 998](#)

 [“Change, Copy, and Create CF Structure on z/OS” on page 1007](#)

[“Change, Copy, and Create Channel” on page 1011](#)

[“Change, Copy, and Create Channel \(MQTT\) on AIX, Linux, and Windows” on page 1047](#)

[“Change, Copy, and Create Channel Listener on Multiplatforms” on page 1052](#)

[“Change, Copy, and Create Namelist” on page 1058](#)

[“Change, Copy, and Create Process” on page 1061](#)

[“Change, Copy, and Create Queue” on page 1064](#)

[“MQCMD\\_CHANGE\\_Q\\_MGR \(Change Queue Manager\)” on page 1105](#)

[“MQCMD\\_CHANGE\\_SECURITY \(Change Security\) on z/OS” on page 1134](#)

[▶ \*\*z/OS\*\* “MQCMD\\_CHANGE\\_SMDS \(Change SMDS\) on z/OS” on page 1134](#)

[“Change, Copy, and Create Service on Multiplatforms” on page 1084](#)

[▶ \*\*z/OS\*\* “Change, Copy, and Create Storage Class on z/OS” on page 1086](#)

[“Change, Copy, and Create Subscription” on page 1089](#)

[“Change, Copy, and Create Topic” on page 1093](#)

[“MQCMD\\_CLEAR\\_Q \(Clear Queue\)” on page 1135](#)

[“MQCMD\\_CLEAR\\_TOPIC\\_STRING \(Clear Topic String\)” on page 1136](#)

[“MQCMD\\_DELETE\\_AUTH\\_INFO \(Delete Authentication Information Object\)” on page 1137](#)

[“MQCMD\\_DELETE\\_AUTH\\_REC \(Delete Authority Record\) on Multiplatforms” on page 1138](#)

[▶ \*\*z/OS\*\* “MQCMD\\_DELETE\\_CF\\_STRUC \(Delete CF Structure\) on z/OS” on page 1140](#)

[“MQCMD\\_DELETE\\_CHANNEL \(Delete Channel\)” on page 1140](#)

[“MQCMD\\_DELETE\\_CHANNEL \(delete channel\) MQTT on AIX, Linux, and Windows” on page 1142](#)

[“MQCMD\\_DELETE\\_LISTENER \(Delete Channel Listener\) on Multiplatforms” on page 1143](#)

[“MQCMD\\_DELETE\\_NAMELIST \(Delete Namelist\)” on page 1143](#)

[“MQCMD\\_DELETE\\_PROCESS \(Delete Process\)” on page 1145](#)

[“MQCMD\\_DELETE\\_Q \(Delete Queue\)” on page 1147](#)

[“MQCMD\\_DELETE\\_SERVICE \(Delete Service\) on Multiplatforms” on page 1149](#)

[▶ \*\*z/OS\*\* “MQCMD\\_DELETE\\_STG\\_CLASS \(Delete Storage Class\) on z/OS” on page 1149](#)

[“MQCMD\\_DELETE\\_SUBSCRIPTION \(Delete Subscription\)” on page 1150](#)

[“MQCMD\\_DELETE\\_TOPIC \(Delete Topic\)” on page 1151](#)

[“MQCMD\\_ESCAPE \(Escape\) on Multiplatforms” on page 1153](#)

[“MQCMD\\_ESCAPE \(Escape\) Response on Multiplatforms” on page 1153](#)

[▶ \*\*z/OS\*\* “MQCMD\\_INQUIRE\\_ARCHIVE \(Inquire Archive\) on z/OS” on page 1161](#)

[▶ \*\*z/OS\*\* “MQCMD\\_INQUIRE\\_ARCHIVE \(Inquire Archive\) Response on z/OS” on page 1161](#)

[“MQCMD\\_INQUIRE\\_AUTH\\_INFO \(Inquire Authentication Information Object\)” on page 1164](#)

[“MQCMD\\_INQUIRE\\_AUTH\\_INFO \(Inquire Authentication Information Object\) Response” on page 1167](#)

[“MQCMD\\_INQUIRE\\_AUTH\\_INFO\\_NAMES \(Inquire Authentication Information Object Names\)” on page 1171](#)

[“MQCMD\\_INQUIRE\\_AUTH\\_INFO\\_NAMES \(Inquire Authentication Information Object Names\) Response” on page 1172](#)

[“MQCMD\\_INQUIRE\\_AUTH\\_RECS \(Inquire Authority Records\) on Multiplatforms” on page 1173](#)

[“MQCMD\\_INQUIRE\\_AUTH\\_RECS \(Inquire Authority Records\) Response on Multiplatforms” on page 1176](#)

[“MQCMD\\_INQUIRE\\_AUTH\\_SERVICE \(Inquire Authority Service\) on Multiplatforms” on page 1179](#)

[“MQCMD\\_INQUIRE\\_AUTH\\_SERVICE \(Inquire Authority Service\) Response on Multiplatforms” on page 1180](#)

[▶ \*\*z/OS\*\* “MQCMD\\_INQUIRE\\_CF\\_STRUC \(Inquire CF Structure\) on z/OS” on page 1180](#)

[▶ \*\*z/OS\*\* “MQCMD\\_INQUIRE\\_CF\\_STRUC \(Inquire CF Structure\) Response on z/OS” on page 1182](#)

[▶ \*\*z/OS\*\* “MQCMD\\_INQUIRE\\_CF\\_STRUC\\_NAMES \(Inquire CF Structure Names\) on z/OS” on page 1185](#)



[z/OS “MQCMD\\_INQUIRE\\_CF\\_STRUC\\_NAMES \(Inquire CF Structure Names\) Response on z/OS” on page 1185](#)

[z/OS “MQCMD\\_INQUIRE\\_CF\\_STRUC\\_STATUS \(Inquire CF Structure Status\) on z/OS” on page 1186](#)

[z/OS “MQCMD\\_INQUIRE\\_CF\\_STRUC\\_STATUS \(Inquire CF Structure Status\) Response on z/OS” on page 1187](#)

[“MQCMD\\_INQUIRE\\_CHANNEL \(Inquire Channel\)” on page 1191](#)

[“MQCMD\\_INQUIRE\\_CHANNEL \(Inquire Channel\) MQTT on AIX, Linux, and Windows” on page 1198](#)

[“MQCMD\\_INQUIRE\\_CHANNEL \(Inquire Channel\) Response” on page 1200](#)

[“MQCMD\\_INQUIRE\\_CHLAUTH\\_RECS \(Inquire Channel Authentication Records\)” on page 1246](#)

[“MQCMD\\_INQUIRE\\_CHLAUTH\\_RECS \(Inquire Channel Authentication Records\) Response” on page 1250](#)

[“MQCMD\\_INQUIRE\\_CHANNEL\\_INIT \(Inquire Channel Initiator\) on z/OS” on page 1211](#)

[“MQCMD\\_INQUIRE\\_CHANNEL\\_INIT \(Inquire Channel Initiator\) Response on z/OS” on page 1212](#)

[“MQCMD\\_INQUIRE\\_LISTENER \(Inquire Channel Listener\) on Multiplatforms” on page 1285](#)

[“MQCMD\\_INQUIRE\\_LISTENER \(Inquire Channel Listener\) Response on Multiplatforms” on page 1287](#)

[“MQCMD\\_INQUIRE\\_LISTENER\\_STATUS \(Inquire Channel Listener Status\) on Multiplatforms” on page 1289](#)

[“MQCMD\\_INQUIRE\\_LISTENER\\_STATUS \(Inquire Channel Listener Status\) Response on Multiplatforms” on page 1291](#)

[“MQCMD\\_INQUIRE\\_CHANNEL\\_NAMES \(Inquire Channel Names\)” on page 1214](#)

[“MQCMD\\_INQUIRE\\_CHANNEL\\_NAMES \(Inquire Channel Names\) Response” on page 1216](#)

[“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\)” on page 1216](#)

[“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\) MQTT on AIX, Linux, and Windows” on page 1229](#)

[“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\) Response” on page 1231](#)

[“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\) Response MQTT on AIX, Linux, and Windows” on page 1244](#)

[“MQCMD\\_INQUIRE\\_CLUSTER\\_Q\\_MGR \(Inquire Cluster Queue Manager\)” on page 1252](#)

[“MQCMD\\_INQUIRE\\_CLUSTER\\_Q\\_MGR \(Inquire Cluster Queue Manager\) Response” on page 1256](#)

[“MQCMD\\_INQUIRE\\_COMM\\_INFO \(Inquire Communication Information Object\) on Multiplatforms” on page 1263](#)

[“MQCMD\\_INQUIRE\\_COMM\\_INFO \(Inquire Communication Information Object\) Response on Multiplatforms” on page 1265](#)

[“MQCMD\\_INQUIRE\\_CONNECTION \(Inquire Connection\)” on page 1267](#)

[“MQCMD\\_INQUIRE\\_CONNECTION \(Inquire Connection\) Response” on page 1271](#)

[“MQCMD\\_INQUIRE\\_ENTITY\\_AUTH \(Inquire Entity Authority\) on Multiplatforms” on page 1278](#)

[“MQCMD\\_INQUIRE\\_ENTITY\\_AUTH \(Inquire Entity Authority\) Response on Multiplatforms” on page 1280](#)

[z/OS “MQCMD\\_INQUIRE\\_QSG \(Inquire Group\) on z/OS” on page 1283](#)

[z/OS “MQCMD\\_INQUIRE\\_QSG \(Inquire Group\) Response on z/OS” on page 1283](#)

[z/OS “MQCMD\\_INQUIRE\\_LOG \(Inquire Log\) on z/OS” on page 1293](#)

[z/OS “MQCMD\\_INQUIRE\\_LOG \(Inquire Log\) Response on z/OS” on page 1293](#)

[“MQCMD\\_INQUIRE\\_NAMELIST \(Inquire Namelist\)” on page 1297](#)

[“MQCMD\\_INQUIRE\\_NAMELIST \(Inquire Namelist\) Response” on page 1300](#)

[“MQCMD\\_INQUIRE\\_NAMELIST\\_NAMES \(Inquire Namelist Names\)” on page 1301](#)

[“MQCMD\\_INQUIRE\\_NAMELIST\\_NAMES \(Inquire Namelist Names\) Response” on page 1302](#)

[“MQCMD\\_INQUIRE\\_PROCESS \(Inquire Process\)” on page 1303](#)

[“MQCMD\\_INQUIRE\\_PROCESS \(Inquire Process\) Response” on page 1305](#)

[“MQCMD\\_INQUIRE\\_PROCESS\\_NAMES \(Inquire Process Names\)” on page 1306](#)

[“MQCMD\\_INQUIRE\\_PROCESS\\_NAMES \(Inquire Process Names\) Response” on page 1307](#)

[“MQCMD\\_INQUIRE\\_PUBSUB\\_STATUS \(Inquire Publish/Subscribe Status\)” on page 1310](#)

[“MQCMD\\_INQUIRE\\_PUBSUB\\_STATUS \(Inquire Publish/Subscribe Status\) Response” on page 1311](#)

[“MQCMD\\_INQUIRE\\_Q \(Inquire Queue\)” on page 1313](#)

[“MQCMD\\_INQUIRE\\_Q \(Inquire Queue\) Response” on page 1323](#)

[“MQCMD\\_INQUIRE\\_Q\\_MGR \(Inquire Queue Manager\)” on page 1334](#)

[“MQCMD\\_INQUIRE\\_Q\\_MGR \(Inquire Queue Manager\) Response” on page 1344](#)

[“MQCMD\\_INQUIRE\\_Q\\_MGR\\_STATUS \(Inquire Queue Manager Status\) on Multiplatforms” on page 1372](#)

[“MQCMD\\_INQUIRE\\_Q\\_MGR\\_STATUS \(Inquire Queue Manager Status\) Response on Multiplatforms” on page 1376](#)

[“MQCMD\\_INQUIRE\\_Q\\_NAMES \(Inquire Queue Names\)” on page 1383](#)

[“MQCMD\\_INQUIRE\\_Q\\_NAMES \(Inquire Queue Names\) Response” on page 1384](#)

[“MQCMD\\_INQUIRE\\_Q\\_STATUS \(Inquire Queue Status\)” on page 1385](#)

[“MQCMD\\_INQUIRE\\_Q\\_STATUS \(Inquire Queue Status\) Response” on page 1390](#)

**z/OS** [“MQCMD\\_INQUIRE\\_SECURITY \(Inquire Security\) on z/OS” on page 1397](#)

**z/OS** [“MQCMD\\_INQUIRE\\_SECURITY \(Inquire Security\) Response on z/OS” on page 1397](#)

[“MQCMD\\_INQUIRE\\_SERVICE \(Inquire Service\) on Multiplatforms” on page 1399](#)

[“MQCMD\\_INQUIRE\\_SERVICE \(Inquire Service\) Response on Multiplatforms” on page 1400](#)

[“MQCMD\\_INQUIRE\\_SERVICE\\_STATUS \(Inquire Service Status\) on Multiplatforms” on page 1402](#)

[“MQCMD\\_INQUIRE\\_SERVICE\\_STATUS \(Inquire Service Status\) Response on Multiplatforms” on page 1403](#)

**z/OS** [“MQCMD\\_INQUIRE\\_SMDS \(Inquire SMDS\) on z/OS” on page 1405](#)

**z/OS** [“MQCMD\\_INQUIRE\\_SMDS \(Inquire SMDS\) Response on z/OS” on page 1405](#)

**z/OS** [“MQCMD\\_INQUIRE\\_SMDSCONN \(Inquire SMDS Connection\) on z/OS” on page 1406](#)

**z/OS** [“MQCMD\\_INQUIRE\\_SMDSCONN \(Inquire SMDS Connection\) Response on z/OS” on page 1407](#)

**z/OS** [“MQCMD\\_INQUIRE\\_STG\\_CLASS \(Inquire Storage Class\) on z/OS” on page 1408](#)

**z/OS** [“MQCMD\\_INQUIRE\\_STG\\_CLASS \(Inquire Storage Class\) Response on z/OS” on page 1410](#)

**z/OS** [“MQCMD\\_INQUIRE\\_STG\\_CLASS\\_NAMES \(Inquire Storage Class Names\) on z/OS” on page 1411](#)

**z/OS** [“MQCMD\\_INQUIRE\\_STG\\_CLASS\\_NAMES \(Inquire Storage Class Names\) Response on z/OS” on page 1413](#)

[“MQCMD\\_INQUIRE\\_SUBSCRIPTION \(Inquire Subscription\)” on page 1413](#)

[“MQCMD\\_INQUIRE\\_SUBSCRIPTION \(Inquire Subscription\) Response” on page 1416](#)

[“MQCMD\\_INQUIRE\\_SUB\\_STATUS \(Inquire Subscription Status\)” on page 1421](#)

[“MQCMD\\_INQUIRE\\_SUB\\_STATUS \(Inquire Subscription Status\) Response” on page 1422](#)

**z/OS** [“MQCMD\\_INQUIRE\\_SYSTEM \(Inquire System\) on z/OS” on page 1424](#)

**z/OS** [“MQCMD\\_INQUIRE\\_SYSTEM \(Inquire System\) Response on z/OS” on page 1424](#)

[“MQCMD\\_INQUIRE\\_TOPIC \(Inquire Topic\)” on page 1428](#)

[“MQCMD\\_INQUIRE\\_TOPIC \(Inquire Topic\) Response” on page 1432](#)

[“MQCMD\\_INQUIRE\\_TOPIC\\_NAMES \(Inquire Topic Names\)” on page 1438](#)

[“MQCMD\\_INQUIRE\\_TOPIC\\_NAMES \(Inquire Topic Names\) Response” on page 1439](#)

[“MQCMD\\_INQUIRE\\_TOPIC\\_STATUS \(Inquire Topic Status\)” on page 1439](#)

[“MQCMD\\_INQUIRE\\_TOPIC\\_STATUS \(Inquire Topic Status\) Response” on page 1441](#)

**z/OS** [“MQCMD\\_INQUIRE\\_USAGE \(Inquire Usage\) on z/OS” on page 1447](#)

- ▶ **z/OS** [“MQCMD\\_INQUIRE\\_USAGE \(Inquire Usage\) Response on z/OS” on page 1447](#)
- ▶ **z/OS** [“MQCMD\\_MOVE\\_Q \(Move Queue\) on z/OS” on page 1452](#)
- [“MQCMD\\_PING\\_CHANNEL \(Ping Channel\)” on page 1453](#)
- [“MQCMD\\_PING\\_Q\\_MGR \(Ping Queue Manager\) on Multiplatforms” on page 1456](#)
- [“MQCMD\\_PURGE\\_CHANNEL \(Purge Channel\) on AIX, Linux, and Windows” on page 1457](#)
- ▶ **z/OS** [“MQCMD\\_RECOVER\\_CF\\_STRUC \(Recover CF Structure\) on z/OS” on page 1457](#)
- [“MQCMD\\_REFRESH\\_CLUSTER \(Refresh Cluster\)” on page 1458](#)
- [“MQCMD\\_REFRESH\\_Q\\_MGR \(Refresh Queue Manager\)” on page 1459](#)
- [“MQCMD\\_REFRESH\\_SECURITY \(Refresh Security\)” on page 1462](#)
- ▶ **z/OS** [“MQCMD\\_RESET\\_CF\\_STRUC \(Reset coupling facility structure\) on z/OS” on page 1464](#)
- [“MQCMD\\_RESET\\_CHANNEL \(Reset Channel\)” on page 1464](#)
- [“MQCMD\\_RESET\\_CLUSTER \(Reset Cluster\)” on page 1466](#)
- [“MQCMD\\_RESET\\_Q\\_MGR \(Reset Queue Manager\)” on page 1467](#)
- [“MQCMD\\_RESET\\_Q\\_STATS \(Reset Queue Statistics\)” on page 1470](#)
- [“MQCMD\\_RESET\\_Q\\_STATS \(Reset Queue Statistics\) Response” on page 1471](#)
- ▶ **z/OS** [“MQCMD\\_RESET\\_SMDS \(Reset shared message data sets\) on z/OS” on page 1472](#)
- [“MQCMD\\_RESOLVE\\_CHANNEL \(Resolve Channel\)” on page 1473](#)
- ▶ **z/OS** [“MQCMD\\_RESUME\\_Q\\_MGR \(Resume Queue Manager\) on z/OS” on page 1475](#)
- [“MQCMD\\_RESUME\\_Q\\_MGR\\_CLUSTER \(Resume Queue Manager Cluster\)” on page 1475](#)
- ▶ **z/OS** [“MQCMD\\_REVERIFY\\_SECURITY \(Reverify Security\) on z/OS” on page 1476](#)
- ▶ **z/OS** [“MQCMD\\_SET\\_ARCHIVE \(Set Archive\) on z/OS” on page 1477](#)
- [“MQCMD\\_SET\\_AUTH\\_REC \(Set Authority Record\) on Multiplatforms” on page 1480](#)
- [“MQCMD\\_SET\\_CHLAUTH\\_REC \(Set Channel Authentication Record\)” on page 1484](#)
- ▶ **z/OS** [“MQCMD\\_SET\\_LOG \(Set Log\) on z/OS” on page 1491](#)
- ▶ **z/OS** [“MQCMD\\_SET\\_SYSTEM \(Set System\) on z/OS” on page 1493](#)
- [“MQCMD\\_START\\_CHANNEL \(Start Channel\)” on page 1495](#)
- [“MQCMD\\_START\\_CHANNEL \(Start Channel\) MQTT on AIX, Linux, and Windows” on page 1498](#)
- [“MQCMD\\_START\\_CHANNEL\\_INIT \(Start Channel Initiator\)” on page 1499](#)
- [“MQCMD\\_START\\_CHANNEL\\_LISTENER \(Start Channel Listener\)” on page 1500](#)
- [“MQCMD\\_START\\_SERVICE \(Start Service\) on Multiplatforms” on page 1502](#)
- ▶ **z/OS** [“MQCMD\\_START\\_SMDSCONN \(Start SMDS Connection\) on z/OS” on page 1502](#)
- [“MQCMD\\_STOP\\_CHANNEL \(Stop Channel\)” on page 1503](#)
- [“MQCMD\\_STOP\\_CHANNEL \(Stop Channel\) MQTT on AIX, Linux, and Windows” on page 1507](#)
- ▶ **z/OS** [“MQCMD\\_STOP\\_CHANNEL\\_INIT \(Stop Channel Initiator\) on z/OS” on page 1508](#)
- [“MQCMD\\_STOP\\_CHANNEL\\_LISTENER \(Stop Channel Listener\)” on page 1509](#)
- [“MQCMD\\_STOP\\_CONNECTION \(Stop Connection\) on Multiplatforms” on page 1510](#)
- [“MQCMD\\_STOP\\_SERVICE \(Stop Service\) on Multiplatforms” on page 1511](#)
- ▶ **z/OS** [“MQCMD\\_STOP\\_SMDSCONN \(stop shared message data sets connection\) on z/OS” on page 1511](#)
- ▶ **z/OS** [“MQCMD\\_SUSPEND\\_Q\\_MGR \(Suspend Queue Manager\) on z/OS” on page 1512](#)
- [“MQCMD\\_SUSPEND\\_Q\\_MGR\\_CLUSTER \(Suspend Queue Manager Cluster\)” on page 1513](#)

## How the PCF definitions are shown

The definitions of the Programmable Command Formats (PCFs) including their commands, responses, parameters, constants, and error codes are shown in a consistent format.

For each PCF command or response, there is a description of what the command or response does, giving the command identifier in parentheses. See [Constants](#) for all values of the command identifier. Each command description starts with a table that identifies the platforms on which the command is valid. For additional, more detailed, usage notes for each command, see the corresponding command description in the [“Definitions of the Programmable Command Formats”](#) on page 983.

IBM MQ products, other than IBM MQ for z/OS, can use the IBM MQ Administration Interface (MQAI), which provides a simplified way for applications written in the C and Visual Basic programming language to build and send PCF commands. For information about the MQAI see the second section of this topic.

## Commands

The *required parameters* and the *optional parameters* are listed.

**Multi** On [Multiplatforms](#), the parameters must occur in this order:

1. All required parameters, in the order stated, followed by
2. Optional parameters as required, in any order, unless noted in the PCF definition.

**z/OS** On z/OS, the parameters can be in any order.

## Responses

The response data attribute is *always returned* whether it is requested or not. This parameter is required to identify, uniquely, the object when there is a possibility of multiple reply messages being returned.

The other attributes shown are *returned if requested* as optional parameters on the command. The response data attributes are not returned in a defined order.

## Parameters and response data

Each parameter name is followed by its structure name in parentheses (details are given in [“Structures for PCF commands and responses”](#) on page 1514 ). The parameter identifier is given at the beginning of the description.

## Constants

For the values of constants used by PCF commands and responses see [Constants](#).

## Informational messages

**z/OS**

On z/OS, a number of command responses return a structure, MQIACF\_COMMAND\_INFO, with values that provide information about the command.

MQIACF_COMMAND_INFO value	Meaning
MQCMDI_CMDSCOPE_ACCEPTED	A command that specified <i>CommandScope</i> was entered. It has been passed to the one or more requested queue managers for processing
MQCMDI_CMDSCOPE_GENERATED	A command that specified <i>CommandScope</i> was generated in response to the command originally entered

Table 188. MQIACF\_COMMAND\_INFO values (continued)


MQIACF_COMMAND_INFO value	Meaning
MQCMDI_CMDScope_COMPLETED	Processing for the command that specified <i>CommandScope</i> - either entered or generated by another command - has completed successfully on all requested queue managers
MQCMDI_QSG_DISP_COMPLETED	Processing for the command that refers to an object with the indicated disposition has completed successfully
MQCMDI_COMMAND_ACCEPTED	Initial processing for the command has completed successfully. The command requires further action by the channel initiator, for which a request has been queued. Messages reporting the success or otherwise of the action are be sent to the command issuer later
MQCMDI_CLUSTER_REQUEST_QUEUED	Initial processing for the command has completed successfully. The command requires further action by the cluster repository manager, for which a request has been queued
MQCMDI_CHANNEL_INIT_STARTED	A Start Channel Initiator command has been issued and the channel initiator address space has been started successfully
MQCMDI_RECOVER_STARTED	The queue manager has successfully started a task to process the Recover CF Structure command for the named structure
MQCMDI_BACKUP_STARTED	The queue manager has successfully started a task to process the Backup CF Structure command for the named structure
MQCMDI_RECOVER_COMPLETED	The named CF structure has been recovered successfully. The structure is available for use again
MQCMDI_SEC_TIMER_ZERO	The Change Security command was entered with the <i>SecurityInterval</i> attribute set to 0. This means that no user timeouts occur
MQCMDI_REFRESH_CONFIGURATION	A Change Queue Manager command has been issued that enables configuration events. Event messages need to be generated to ensure that the configuration information is complete and up to date
MQCMDI_IMS_BRIDGE_SUSPENDED	The MQ-IMS bridge facility is suspended.
MQCMDI_DB2_SUSPENDED	The connection to Db2 is suspended
MQCMDI_DB2_OBSOLETE_MSGS	Obsolete Db2 messages exist in the queue sharing group

## Error codes

 In z/OS, PCF commands can return MQRC reason codes instead of MQRCCF codes

MQRCCF codes are used in AIX, Linux, and Windows. At the end of most command format definitions, there is a list of error codes that might be returned by that command.

## Error codes applicable to all commands

In addition to those error codes listed under each command format, any command might return the following error codes in the response format header (descriptions of the MQR\*\_\* error codes are given in the [Messages and reason codes](#)  and [IBM MQ for z/OS messages, completion, and reason codes](#) documentation):

### Reason (MQLONG)

The value can be any of the following values:

#### **MQR\*\_NONE**

(0, X'000') No reason to report.

#### **MQR\*\_MSG\_TOO\_BIG\_FOR\_Q**

(2030, X'7EE') Message length greater than maximum for queue.

#### **MQR\*\_CONNECTION\_BROKEN**

(2009, X'7D9') Connection to queue manager lost.

#### **MQR\*\_NOT\_AUTHORIZED**

(2035, X'7F3') Not authorized for access.

#### **MQR\*\_SELECTOR\_ERROR**

(2067, X'813') Attribute selector not valid.

#### **MQR\*\_STORAGE\_NOT\_AVAILABLE**

(2071, X'817') Insufficient storage available.

#### **MQR\*\_UNKNOWN\_OBJECT\_NAME**

(2085, X'825') Unknown object name.

#### **MQRCCF\_ATTR\_VALUE\_ERROR**

Attribute value not valid.

#### **MQRCCF\_CFBF\_FILTER\_VAL\_LEN\_ERROR**

Filter value length not valid.

#### **MQRCCF\_CFBF\_LENGTH\_ERROR**

Structure length not valid.

#### **MQRCCF\_CFBF\_OPERATOR\_ERROR**

Operator error.

#### **MQRCCF\_CFBF\_PARM\_ID\_ERROR**

Parameter identifier not valid.

#### **MQRCCF\_CFBS\_DUPLICATE\_PARM**

Duplicate parameter.

#### **MQRCCF\_CFBS\_LENGTH\_ERROR**

Structure length not valid.

#### **MQRCCF\_CFBS\_PARM\_ID\_ERROR**

Parameter identifier not valid.

#### **MQRCCF\_CFBS\_STRING\_LENGTH\_ERROR**

String length not valid.

#### **MQRCCF\_CFGR\_LENGTH\_ERROR**

Structure length not valid.

#### **MQRCCF\_CFGR\_PARM\_COUNT\_ERROR**

Parameter count not valid.

#### **MQRCCF\_CFGR\_PARM\_ID\_ERROR**

Parameter identifier not valid.

**MQRCCF\_CFH\_COMMAND\_ERROR**  
Command identifier not valid.

**MQRCCF\_CFH\_CONTROL\_ERROR**  
Control option not valid.

**MQRCCF\_CFH\_LENGTH\_ERROR**  
Structure length not valid.

**MQRCCF\_CFH\_MSG\_SEQ\_NUMBER\_ERR**  
Message sequence number not valid.

**MQRCCF\_CFH\_PARM\_COUNT\_ERROR**  
Parameter count not valid.

**MQRCCF\_CFH\_TYPE\_ERROR**  
Type not valid.

**MQRCCF\_CFH\_VERSION\_ERROR**  
Structure version number is not valid.

**MQRCCF\_CFIF\_LENGTH\_ERROR**  
Structure length not valid.

**MQRCCF\_CFIF\_OPERATOR\_ERROR**  
Operator error.

**MQRCCF\_CFIF\_PARM\_ID\_ERROR**  
Parameter identifier not valid.

**MQRCCF\_CFIL\_COUNT\_ERROR**  
Count of parameter values not valid.

**MQRCCF\_CFIL\_DUPLICATE\_VALUE**  
Duplicate parameter.

**MQRCCF\_CFIL\_LENGTH\_ERROR**  
Structure length not valid.

**MQRCCF\_CFIL\_PARM\_ID\_ERROR**  
Parameter identifier not valid.

**MQRCCF\_CFIN\_DUPLICATE\_PARM**  
Duplicate parameter.

**MQRCCF\_CFIN\_LENGTH\_ERROR**  
Structure length not valid.

**MQRCCF\_CFIN\_PARM\_ID\_ERROR**  
Parameter identifier not valid.

**MQRCCF\_CFSF\_FILTER\_VAL\_LEN\_ERROR**  
Filter value length not valid.

**MQRCCF\_CFSF\_LENGTH\_ERROR**  
Structure length not valid.

**MQRCCF\_CFSF\_OPERATOR\_ERROR**  
Operator error.

**MQRCCF\_CFSF\_PARM\_ID\_ERROR**  
Parameter identifier not valid.

**MQRCCF\_CFSL\_COUNT\_ERROR**  
Count of parameter values not valid.

**MQRCCF\_CFSL\_DUPLICATE\_PARM**  
Duplicate parameter.

**MQRCCF\_CFSL\_LENGTH\_ERROR**  
Structure length not valid.

**MQRCCF\_CFSL\_PARM\_ID\_ERROR**  
Parameter identifier not valid.

**MQRCCF\_CFSL\_STRING\_LENGTH\_ERROR**

String length value not valid.

**MQRCCF\_CFSL\_TOTAL\_LENGTH\_ERROR**

Total string length error.

**MQRCCF\_CFST\_CONFLICTING\_PARM**

Conflicting parameters.

**MQRCCF\_CFST\_DUPLICATE\_PARM**

Duplicate parameter.

**MQRCCF\_CFST\_LENGTH\_ERROR**

Structure length not valid.

**MQRCCF\_CFST\_PARM\_ID\_ERROR**

Parameter identifier not valid.

**MQRCCF\_CFST\_STRING\_LENGTH\_ERROR**

String length value not valid.

**MQRCCF\_COMMAND\_FAILED**

Command failed.

**MQRCCF\_ENCODING\_ERROR**

Encoding error.

**MQRCCF\_MD\_FORMAT\_ERROR**

Format not valid.

**MQRCCF\_MSG\_SEQ\_NUMBER\_ERROR**

Message sequence number not valid.

**MQRCCF\_MSG\_TRUNCATED**

Message truncated.

**MQRCCF\_MSG\_LENGTH\_ERROR**

Message length not valid.

**MQRCCF\_OBJECT\_NAME\_ERROR**

Object name not valid.

**MQRCCF\_OBJECT\_OPEN**

Object is open.

**MQRCCF\_PARM\_COUNT\_TOO\_BIG**

Parameter count too large.

**MQRCCF\_PARM\_COUNT\_TOO\_SMALL**

Parameter count too small.

**MQRCCF\_PARM\_SEQUENCE\_ERROR**

Parameter sequence not valid.

**MQRCCF\_PARM\_SYNTAX\_ERROR**

Syntax error found in parameter.

**MQRCCF\_STRUCTURE\_TYPE\_ERROR**

Structure type not valid.

**MQRCCF\_UNKNOWN\_OBJECT\_NAME**

Unknown object name.

***PCF commands and responses in groups***

In the main navigation for this product documentation, the PCF commands and data responses are given in alphabetical order. This topic gives an alternative index, by grouping the PCF commands by functional area.

**Authentication Information commands**

- [“Change, Copy, and Create Authentication Information Object” on page 998](#)



- [“MQCMD\\_DELETE\\_AUTH\\_INFO \(Delete Authentication Information Object\)” on page 1137](#)
- [“MQCMD\\_INQUIRE\\_AUTH\\_INFO \(Inquire Authentication Information Object\)” on page 1164](#)
- [“MQCMD\\_INQUIRE\\_AUTH\\_INFO\\_NAMES \(Inquire Authentication Information Object Names\)” on page 1171](#)

## Authority Record commands




- [“MQCMD\\_DELETE\\_AUTH\\_REC \(Delete Authority Record\) on Multiplatforms” on page 1138](#)
- [“MQCMD\\_INQUIRE\\_AUTH\\_RECS \(Inquire Authority Records\) on Multiplatforms” on page 1173](#)
- [“MQCMD\\_INQUIRE\\_AUTH\\_SERVICE \(Inquire Authority Service\) on Multiplatforms” on page 1179](#)
- [“MQCMD\\_INQUIRE\\_ENTITY\\_AUTH \(Inquire Entity Authority\) on Multiplatforms” on page 1278](#)
- [“MQCMD\\_SET\\_AUTH\\_REC \(Set Authority Record\) on Multiplatforms” on page 1480](#)

## CF commands

### z/OS

- [“MQCMD\\_BACKUP\\_CF\\_STRUC \(Backup CF Structure\) on z/OS” on page 1102](#)
- [“Change, Copy, and Create CF Structure on z/OS” on page 1007](#)
- [“MQCMD\\_DELETE\\_CF\\_STRUC \(Delete CF Structure\) on z/OS” on page 1140](#)
- [“MQCMD\\_INQUIRE\\_CF\\_STRUC \(Inquire CF Structure\) on z/OS” on page 1180](#)
- [“MQCMD\\_INQUIRE\\_CF\\_STRUC\\_NAMES \(Inquire CF Structure Names\) on z/OS” on page 1185](#)
- [“MQCMD\\_INQUIRE\\_CF\\_STRUC\\_STATUS \(Inquire CF Structure Status\) on z/OS” on page 1186](#)
- [“MQCMD\\_RECOVER\\_CF\\_STRUC \(Recover CF Structure\) on z/OS” on page 1457](#)

## Channel commands

- [“Change, Copy, and Create Channel” on page 1011](#)
- [“MQCMD\\_DELETE\\_CHANNEL \(Delete Channel\)” on page 1140](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL \(Inquire Channel\)” on page 1191](#)
-  [“MQCMD\\_INQUIRE\\_CHANNEL\\_INIT \(Inquire Channel Initiator\) on z/OS” on page 1211](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL\\_NAMES \(Inquire Channel Names\)” on page 1214](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\)” on page 1216](#)
- [“MQCMD\\_PING\\_CHANNEL \(Ping Channel\)” on page 1453](#)
- [“MQCMD\\_RESET\\_CHANNEL \(Reset Channel\)” on page 1464](#)
- [“MQCMD\\_RESOLVE\\_CHANNEL \(Resolve Channel\)” on page 1473](#)
- [“MQCMD\\_START\\_CHANNEL \(Start Channel\)” on page 1495](#)
-  [“MQCMD\\_START\\_CHANNEL\\_INIT \(Start Channel Initiator\)” on page 1499](#)
- [“MQCMD\\_STOP\\_CHANNEL \(Stop Channel\)” on page 1503](#)
-  [“MQCMD\\_STOP\\_CHANNEL\\_INIT \(Stop Channel Initiator\) on z/OS” on page 1508](#)

## Channel commands (MQTT)

- [“Change, Copy, and Create Channel \(MQTT\) on AIX, Linux, and Windows” on page 1047](#)
- [“MQCMD\\_DELETE\\_CHANNEL \(delete channel\) MQTT on AIX, Linux, and Windows” on page 1142](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL \(Inquire Channel\) MQTT on AIX, Linux, and Windows” on page 1198](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\) MQTT on AIX, Linux, and Windows” on page 1229](#)

- [“MQCMD\\_PURGE\\_CHANNEL \(Purge Channel\) on AIX, Linux, and Windows” on page 1457](#)
- [“MQCMD\\_START\\_CHANNEL \(Start Channel\) MQTT on AIX, Linux, and Windows” on page 1498](#)
- [“MQCMD\\_STOP\\_CHANNEL \(Stop Channel\) MQTT on AIX, Linux, and Windows” on page 1507](#)

### **Channel Authentication commands**

- [“MQCMD\\_INQUIRE\\_CHLAUTH\\_RECS \(Inquire Channel Authentication Records\)” on page 1246](#)
- [“MQCMD\\_SET\\_CHLAUTH\\_REC \(Set Channel Authentication Record\)” on page 1484](#)

### **Channel Listener commands**

- [“Change, Copy, and Create Channel Listener on Multiplatforms” on page 1052](#)
- [“MQCMD\\_DELETE\\_LISTENER \(Delete Channel Listener\) on Multiplatforms” on page 1143](#)
- [“MQCMD\\_INQUIRE\\_LISTENER \(Inquire Channel Listener\) on Multiplatforms” on page 1285](#)
- [“MQCMD\\_INQUIRE\\_LISTENER\\_STATUS \(Inquire Channel Listener Status\) on Multiplatforms” on page 1289](#)
- [“MQCMD\\_START\\_CHANNEL\\_LISTENER \(Start Channel Listener\)” on page 1500](#)
- [“MQCMD\\_STOP\\_CHANNEL\\_LISTENER \(Stop Channel Listener\)” on page 1509](#)

### **Cluster commands**

- [“MQCMD\\_INQUIRE\\_CLUSTER\\_Q\\_MGR \(Inquire Cluster Queue Manager\)” on page 1252](#)
- [“MQCMD\\_REFRESH\\_CLUSTER \(Refresh Cluster\)” on page 1458](#)
- [“MQCMD\\_RESET\\_CLUSTER \(Reset Cluster\)” on page 1466](#)
- [“MQCMD\\_RESUME\\_Q\\_MGR\\_CLUSTER \(Resume Queue Manager Cluster\)” on page 1475](#)
- [“MQCMD\\_SUSPEND\\_Q\\_MGR\\_CLUSTER \(Suspend Queue Manager Cluster\)” on page 1513](#)

### **Communication Information commands**

- [“Change, Copy, and Create Communication Information Object on Multiplatforms” on page 1054](#)
- [“MQCMD\\_DELETE\\_COMM\\_INFO \(Delete Communication Information Object\) on Multiplatforms” on page 1143](#)
- [“MQCMD\\_INQUIRE\\_COMM\\_INFO \(Inquire Communication Information Object\) on Multiplatforms” on page 1263](#)

### **Connection commands**

- [“MQCMD\\_INQUIRE\\_CONNECTION \(Inquire Connection\)” on page 1267](#)
- [“MQCMD\\_STOP\\_CONNECTION \(Stop Connection\) on Multiplatforms” on page 1510](#)

### **Escape command**

- [“MQCMD\\_ESCAPE \(Escape\) on Multiplatforms” on page 1153](#)

### **Namelist commands**

- [“Change, Copy, and Create Namelist” on page 1058](#)
- [“MQCMD\\_DELETE\\_NAMELIST \(Delete Namelist\)” on page 1143](#)
- [“MQCMD\\_INQUIRE\\_NAMELIST \(Inquire Namelist\)” on page 1297](#)
- [“MQCMD\\_INQUIRE\\_NAMELIST\\_NAMES \(Inquire Namelist Names\)” on page 1301](#)


## Process commands

- [“Change, Copy, and Create Process” on page 1061](#)
- [“MQCMD\\_DELETE\\_PROCESS \(Delete Process\)” on page 1145](#)
- [“MQCMD\\_INQUIRE\\_PROCESS \(Inquire Process\)” on page 1303](#)
- [“MQCMD\\_INQUIRE\\_PROCESS\\_NAMES \(Inquire Process Names\)” on page 1306](#)



## Publish/subscribe commands

- [“Change, Copy, and Create Subscription” on page 1089](#)
- [“Change, Copy, and Create Topic” on page 1093](#)
- [“MQCMD\\_CLEAR\\_TOPIC\\_STRING \(Clear Topic String\)” on page 1136](#)
- [“MQCMD\\_DELETE\\_SUBSCRIPTION \(Delete Subscription\)” on page 1150](#)
- [“MQCMD\\_DELETE\\_TOPIC \(Delete Topic\)” on page 1151](#)
- [“MQCMD\\_INQUIRE\\_PUBSUB\\_STATUS \(Inquire Publish/Subscribe Status\)” on page 1310](#)
- [“MQCMD\\_INQUIRE\\_SUBSCRIPTION \(Inquire Subscription\)” on page 1413](#)
- [“MQCMD\\_INQUIRE\\_SUB\\_STATUS \(Inquire Subscription Status\)” on page 1421](#)
- [“MQCMD\\_INQUIRE\\_TOPIC \(Inquire Topic\)” on page 1428](#)
- [“MQCMD\\_INQUIRE\\_TOPIC\\_NAMES \(Inquire Topic Names\)” on page 1438](#)
- [“MQCMD\\_INQUIRE\\_TOPIC\\_STATUS \(Inquire Topic Status\)” on page 1439](#)

## Queue commands


- [“Change, Copy, and Create Queue” on page 1064](#)
- [“MQCMD\\_CLEAR\\_Q \(Clear Queue\)” on page 1135](#)
- [“MQCMD\\_DELETE\\_Q \(Delete Queue\)” on page 1147](#)
- [“MQCMD\\_INQUIRE\\_Q \(Inquire Queue\)” on page 1313](#)
- [“MQCMD\\_INQUIRE\\_Q\\_NAMES \(Inquire Queue Names\)” on page 1383](#)
- [“MQCMD\\_INQUIRE\\_Q\\_STATUS \(Inquire Queue Status\)” on page 1385](#)
-  [“MQCMD\\_MOVE\\_Q \(Move Queue\) on z/OS” on page 1452](#)
- [“MQCMD\\_RESET\\_Q\\_STATS \(Reset Queue Statistics\)” on page 1470](#)

## Queue Manager commands

- [“MQCMD\\_CHANGE\\_Q\\_MGR \(Change Queue Manager\)” on page 1105](#)
- [“MQCMD\\_INQUIRE\\_Q\\_MGR \(Inquire Queue Manager\)” on page 1334](#)
- [“MQCMD\\_INQUIRE\\_Q\\_MGR\\_STATUS \(Inquire Queue Manager Status\) on Multiplatforms” on page 1372](#)
- [“MQCMD\\_PING\\_Q\\_MGR \(Ping Queue Manager\) on Multiplatforms” on page 1456](#)
- [“MQCMD\\_REFRESH\\_Q\\_MGR \(Refresh Queue Manager\)” on page 1459](#)
- [“MQCMD\\_RESET\\_Q\\_MGR \(Reset Queue Manager\)” on page 1467](#)
-  [“MQCMD\\_RESUME\\_Q\\_MGR \(Resume Queue Manager\) on z/OS” on page 1475](#)
-  [“MQCMD\\_SUSPEND\\_Q\\_MGR \(Suspend Queue Manager\) on z/OS” on page 1512](#)

## Security commands

- [“MQCMD\\_CHANGE\\_SECURITY \(Change Security\) on z/OS” on page 1134](#)
- [“MQCMD\\_INQUIRE\\_SECURITY \(Inquire Security\) on z/OS” on page 1397](#)

- [“MQCMD\\_REFRESH\\_SECURITY \(Refresh Security\)” on page 1462](#)
-  [“MQCMD\\_REVERIFY\\_SECURITY \(Reverify Security\) on z/OS” on page 1476](#)

## Service commands

- [“Change, Copy, and Create Service on Multiplatforms” on page 1084](#)
- [“MQCMD\\_DELETE\\_SERVICE \(Delete Service\) on Multiplatforms” on page 1149](#)
- [“MQCMD\\_INQUIRE\\_SERVICE \(Inquire Service\) on Multiplatforms” on page 1399](#)
- [“MQCMD\\_INQUIRE\\_SERVICE\\_STATUS \(Inquire Service Status\) on Multiplatforms” on page 1402](#)
- [“MQCMD\\_START\\_SERVICE \(Start Service\) on Multiplatforms” on page 1502](#)
- [“MQCMD\\_STOP\\_SERVICE \(Stop Service\) on Multiplatforms” on page 1511](#)


## SMDS commands



- [“MQCMD\\_CHANGE\\_SMDS \(Change SMDS\) on z/OS” on page 1134](#)
- [“MQCMD\\_INQUIRE\\_SMDS \(Inquire SMDS\) on z/OS” on page 1405](#)
- [“MQCMD\\_INQUIRE\\_SMDSCONN \(Inquire SMDS Connection\) on z/OS” on page 1406](#)
- [“MQCMD\\_RESET\\_SMDS \(Reset shared message data sets\) on z/OS” on page 1472](#)
- [“MQCMD\\_START\\_SMDSCONN \(Start SMDS Connection\) on z/OS” on page 1502](#)
- [“MQCMD\\_STOP\\_SMDSCONN \(stop shared message data sets connection\) on z/OS” on page 1511](#)

## Storage class commands




-  [“Change, Copy, and Create Storage Class on z/OS” on page 1086](#)
- [“MQCMD\\_DELETE\\_STG\\_CLASS \(Delete Storage Class\) on z/OS” on page 1149](#)
- [“MQCMD\\_INQUIRE\\_STG\\_CLASS \(Inquire Storage Class\) on z/OS” on page 1408](#)
- [“MQCMD\\_INQUIRE\\_STG\\_CLASS\\_NAMES \(Inquire Storage Class Names\) on z/OS” on page 1411](#)

## System commands




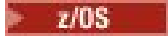




- [“MQCMD\\_INQUIRE\\_ARCHIVE \(Inquire Archive\) on z/OS” on page 1161](#)
- [“MQCMD\\_SET\\_ARCHIVE \(Set Archive\) on z/OS” on page 1477](#)
- [“MQCMD\\_INQUIRE\\_QSG \(Inquire Group\) on z/OS” on page 1283](#)
- [“MQCMD\\_INQUIRE\\_LOG \(Inquire Log\) on z/OS” on page 1293](#)
- [“MQCMD\\_SET\\_LOG \(Set Log\) on z/OS” on page 1491](#)
- [“MQCMD\\_INQUIRE\\_SYSTEM \(Inquire System\) on z/OS” on page 1424](#)
- [“MQCMD\\_SET\\_SYSTEM \(Set System\) on z/OS” on page 1493](#)
- [“MQCMD\\_INQUIRE\\_USAGE \(Inquire Usage\) on z/OS” on page 1447](#)

## Data responses to commands

- [“MQCMD\\_ESCAPE \(Escape\) Response on Multiplatforms” on page 1153](#)
-  [“MQCMD\\_INQUIRE\\_ARCHIVE \(Inquire Archive\) Response on z/OS” on page 1161](#)
- [“MQCMD\\_INQUIRE\\_AUTH\\_INFO \(Inquire Authentication Information Object\) Response” on page 1167](#)

- [“MQCMD\\_INQUIRE\\_AUTH\\_INFO\\_NAMES \(Inquire Authentication Information Object Names\) Response” on page 1172](#)
- [“MQCMD\\_INQUIRE\\_AUTH\\_RECS \(Inquire Authority Records\) Response on Multiplatforms” on page 1176](#)
- [“MQCMD\\_INQUIRE\\_AUTH\\_SERVICE \(Inquire Authority Service\) Response on Multiplatforms” on page 1180](#)
- [▶ z/OS “MQCMD\\_INQUIRE\\_CF\\_STRUC \(Inquire CF Structure\) Response on z/OS” on page 1182](#)
- [▶ z/OS “MQCMD\\_INQUIRE\\_CF\\_STRUC\\_NAMES \(Inquire CF Structure Names\) Response on z/OS” on page 1185](#)
- [▶ z/OS “MQCMD\\_INQUIRE\\_CF\\_STRUC\\_STATUS \(Inquire CF Structure Status\) Response on z/OS” on page 1187](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL \(Inquire Channel\) Response” on page 1200](#)
- [“MQCMD\\_INQUIRE\\_CHLAUTH\\_RECS \(Inquire Channel Authentication Records\) Response” on page 1250](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL\\_INIT \(Inquire Channel Initiator\) Response on z/OS” on page 1212](#)
- [“MQCMD\\_INQUIRE\\_LISTENER \(Inquire Channel Listener\) Response on Multiplatforms” on page 1287](#)
- [“MQCMD\\_INQUIRE\\_LISTENER\\_STATUS \(Inquire Channel Listener Status\) Response on Multiplatforms” on page 1291](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL\\_NAMES \(Inquire Channel Names\) Response” on page 1216](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\) Response” on page 1231](#)
- [“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\) Response MQTT on AIX, Linux, and Windows” on page 1244](#)
- [“MQCMD\\_INQUIRE\\_CLUSTER\\_Q\\_MGR \(Inquire Cluster Queue Manager\) Response” on page 1256](#)
- [“MQCMD\\_INQUIRE\\_COMM\\_INFO \(Inquire Communication Information Object\) Response on Multiplatforms” on page 1265](#)
- [“MQCMD\\_INQUIRE\\_CONNECTION \(Inquire Connection\) Response” on page 1271](#)
- [“MQCMD\\_INQUIRE\\_ENTITY\\_AUTH \(Inquire Entity Authority\) Response on Multiplatforms” on page 1280](#)
- [▶ z/OS “MQCMD\\_INQUIRE\\_QSG \(Inquire Group\) Response on z/OS” on page 1283](#)
- [▶ z/OS “MQCMD\\_INQUIRE\\_LOG \(Inquire Log\) Response on z/OS” on page 1293](#)
- [“MQCMD\\_INQUIRE\\_NAMELIST \(Inquire Namelist\) Response” on page 1300](#)
- [“MQCMD\\_INQUIRE\\_NAMELIST\\_NAMES \(Inquire Namelist Names\) Response” on page 1302](#)
- [“MQCMD\\_INQUIRE\\_PROCESS \(Inquire Process\) Response” on page 1305](#)
- [“MQCMD\\_INQUIRE\\_PROCESS\\_NAMES \(Inquire Process Names\) Response” on page 1307](#)
- [“MQCMD\\_INQUIRE\\_PUBSUB\\_STATUS \(Inquire Publish/Subscribe Status\) Response” on page 1311](#)
- [“MQCMD\\_INQUIRE\\_Q \(Inquire Queue\) Response” on page 1323](#)
- [“MQCMD\\_INQUIRE\\_Q\\_MGR \(Inquire Queue Manager\) Response” on page 1344](#)
- [“MQCMD\\_INQUIRE\\_Q\\_MGR\\_STATUS \(Inquire Queue Manager Status\) Response on Multiplatforms” on page 1376](#)
- [“MQCMD\\_INQUIRE\\_Q\\_NAMES \(Inquire Queue Names\) Response” on page 1384](#)
- [“MQCMD\\_RESET\\_Q\\_STATS \(Reset Queue Statistics\) Response” on page 1471](#)
- [“MQCMD\\_INQUIRE\\_Q\\_STATUS \(Inquire Queue Status\) Response” on page 1390](#)
- [▶ z/OS “MQCMD\\_INQUIRE\\_SECURITY \(Inquire Security\) Response on z/OS” on page 1397](#)
- [“MQCMD\\_INQUIRE\\_SERVICE \(Inquire Service\) Response on Multiplatforms” on page 1400](#)

- [“MQCMD\\_INQUIRE\\_SERVICE\\_STATUS \(Inquire Service Status\) Response on Multiplatforms” on page 1403](#)
-  [“MQCMD\\_INQUIRE\\_STG\\_CLASS \(Inquire Storage Class\) Response on z/OS” on page 1410](#)
-  [“MQCMD\\_INQUIRE\\_STG\\_CLASS\\_NAMES \(Inquire Storage Class Names\) Response on z/OS” on page 1413](#)
-  [“MQCMD\\_INQUIRE\\_SMDS \(Inquire SMDS\) Response on z/OS” on page 1405](#)
-  [“MQCMD\\_INQUIRE\\_SMDSCONN \(Inquire SMDS Connection\) Response on z/OS” on page 1407](#)
- [“MQCMD\\_INQUIRE\\_SUBSCRIPTION \(Inquire Subscription\) Response” on page 1416](#)
- [“MQCMD\\_INQUIRE\\_SUB\\_STATUS \(Inquire Subscription Status\) Response” on page 1422](#)
-  [“MQCMD\\_INQUIRE\\_SYSTEM \(Inquire System\) Response on z/OS” on page 1424](#)
- [“MQCMD\\_INQUIRE\\_TOPIC \(Inquire Topic\) Response” on page 1432](#)
- [“MQCMD\\_INQUIRE\\_TOPIC\\_NAMES \(Inquire Topic Names\) Response” on page 1439](#)
- [“MQCMD\\_INQUIRE\\_TOPIC\\_STATUS \(Inquire Topic Status\) Response” on page 1441](#)
-  [“MQCMD\\_INQUIRE\\_USAGE \(Inquire Usage\) Response on z/OS” on page 1447](#)

### ***Change, Copy, and Create Authentication Information Object***

The Change authentication information PCF command changes attributes of an existing authentication information object. The Create and Copy authentication information commands create new authentication information objects - the Copy command uses attribute values of an existing object.

The Change authentication information (MQCMD\_CHANGE\_AUTH\_INFO) command changes the specified attributes in an authentication information object. For any optional parameters that are omitted, the value does not change.

The Copy authentication information (MQCMD\_COPY\_AUTH\_INFO) command creates new authentication information object using, for attributes not specified in the command, the attribute values of an existing authentication information object.

The Create authentication information (MQCMD\_CREATE\_AUTH\_INFO) command creates an authentication information object. Any attributes that are not defined explicitly are set to the default values on the destination queue manager. A system default authentication information object exists and default values are taken from it.

### **Required parameters (Change authentication information)**

#### **AuthInfoName (MQCFST)**

The authentication information object name (parameter identifier: MQCA\_AUTH\_INFO\_NAME).

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH.

#### **AuthInfoType (MQCFIN)**

The type of authentication information object (parameter identifier: MQIA\_AUTH\_INFO\_TYPE).

The value can be:

##### **MQAIT\_CRL\_LDAP**

This defines this authentication information object as specifying an LDAP server containing Certificate Revocation Lists.

##### **MQAIT\_OCSP**

This value defines this authentication information object as specifying certificate revocation checking using OCSP.

AuthInfoType MQAIT\_OCSP does not apply for use on IBM i or z/OS queue managers, but it can be specified on those platforms to be copied to the client channel definition table for client use.

#### **MQAIT\_IDPW\_OS**

This value defines this authentication information object as specifying certificate revocation checking using user ID and password checking through the operating system.

#### **MQAIT\_IDPW\_LDAP**

This value defines this authentication information object as specifying certificate revocation checking using user ID and password checking through an LDAP server.


**Important:** This option is not valid on z/OS.

See [Securing IBM MQ](#) for more information.

### **Required parameters (Copy authentication information)**

#### **FromAuthInfoName (MQCFST)**

The name of the authentication information object definition to be copied from (parameter identifier: MQCACF\_FROM\_AUTH\_INFO\_NAME).

 On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD\_Q\_MGR or MQQSGD\_COPY to copy from. This parameter is ignored if a value of MQQSGD\_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToAuthInfoName* and the disposition of MQQSGD\_GROUP is searched for to copy from.

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH.

#### **ToAuthInfoName (MQCFST)**

The name of the authentication information object to copy to (parameter identifier: MQCACF\_TO\_AUTH\_INFO\_NAME).

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH.

#### **AuthInfoType (MQCFIN)**

The type of authentication information object (parameter identifier: MQIA\_AUTH\_INFO\_TYPE). The value must match the AuthInfoType of the authentication information object from which you are copying.

The value can be:

#### **MQAIT\_CRL\_LDAP**

This value defines this authentication information object as specifying Certificate Revocation Lists that are held on LDAP.

#### **MQAIT\_OCSP**

This value defines this authentication information object as specifying certificate revocation checking using OCSP.

#### **MQAIT\_IDPW\_OS**

This value defines this authentication information object as specifying certificate revocation checking using user ID and password checking through the operating system.

#### **MQAIT\_IDPW\_LDAP**

This value defines this authentication information object as specifying certificate revocation checking using user ID and password checking through an LDAP server.

**Important:** This option is not valid on z/OS.

See [Securing IBM MQ](#) for more information.

### **Required parameters (Create authentication information)**

#### **AuthInfoName (MQCFST)**

Authentication information object name (parameter identifier: MQCA\_AUTH\_INFO\_NAME).

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH.

### **AuthInfoType (MQCFIN)**

The type of authentication information object (parameter identifier: MQIA\_AUTH\_INFO\_TYPE).

The following values are accepted:

#### **MQAIT\_CRL\_LDAP**

This value defines this authentication information object as specifying an LDAP server containing Certificate Revocation Lists.

#### **MQAIT\_OCSP**

This value defines this authentication information object as specifying certificate revocation checking using OCSP.

An authentication information object with AuthInfoType MQAIT\_OCSP does not apply for use on IBM i or z/OS queue managers, but it can be specified on those platforms to be copied to the client channel definition table for client use.

#### **MQAIT\_IDPW\_OS**

This value defines this authentication information object as specifying certificate revocation checking using user ID and password checking through the operating system.

#### **MQAIT\_IDPW\_LDAP**

This value defines this authentication information object as specifying certificate revocation checking using user ID and password checking through an LDAP server.

**Important:** This option is not valid on z/OS.

See [Securing IBM MQ](#) for more information.

## **Optional parameters (Change, Copy, and Create Authentication Information Object)**

### **AdoptContext (MQCFIN)**

Whether to use the presented credentials as the context for this application (parameter identifier MQIA\_ADOPT\_CONTEXT). This means that they are used for authorization checks, shown on administrative displays, and appear in messages.

#### **MQADPCTX\_YES**

The user ID presented in the MQCSP structure, which has been successfully validated by password, is adopted as the context to use for this application. Therefore, this user ID will be the credentials checked for authorization to use IBM MQ resources.

If the user ID presented is an LDAP user ID, and authorization checks are done using operating system user IDs, the [ShortUser](#) associated with the user entry in LDAP will be adopted as the credentials for authorization checks to be done against.

#### **MQADPCTX\_NO**

Authentication will be performed on the user ID and password presented in the MQCSP structure, but then the credentials will not be adopted for further use. Authorization will be performed using the user ID the application is running under.

This attribute is only valid for **AuthInfoType** of *MQAIT\_IDPW\_OS* and *MQAIT\_IDPW\_LDAP*.

The maximum length is MQIA\_ADOPT\_CONTEXT\_LENGTH.

### **AuthInfoConnName (MQCFST)**

The connection name of the authentication information object (parameter identifier: MQCA\_AUTH\_INFO\_CONN\_NAME).

This parameter is relevant only when AuthInfoType is set to *MQAIT\_CRL\_LDAP* or *MQAIT\_IDPW\_LDAP*, when it is required.

When used with an AuthInfoType of *MQAIT\_IDPW\_LDAP*, this can be a comma separated list of connection names.

 On [Multiplatforms](#), the maximum length is MQ\_AUTH\_INFO\_CONN\_NAME\_LENGTH.



On z/OS, the maximum length is MQ\_LOCAL\_ADDRESS\_LENGTH.

### **AuthInfoDesc (MQCFST)**

The description of the authentication information object (parameter identifier: MQCA\_AUTH\_INFO\_DESC).

The maximum length is MQ\_AUTH\_INFO\_DESC\_LENGTH.

### **AuthenticationMethod (MQCFIN)**

Authentication methods for user passwords (parameter identifier: MQIA\_AUTHENTICATION\_METHOD). Possible values are:

#### **MQAUTHENTICATE\_OS**

Use the traditional UNIX password verification method

This is the default value.

#### **MQAUTHENTICATE\_PAM**

Use the Pluggable Authentication Method to authenticate the user passwords.

You can set the PAM value only on AIX and Linux platforms.

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_OS*, and is not valid on IBM MQ for z/OS.

### **AuthorizationMethod (MQCFIN)**

Authorization methods for the queue manager (parameter identifier: MQIA\_LDAP\_AUTHORMD). Possible values are:

#### **MQLDAP\_AUTHORMD\_OS**

Use operating system groups to determine permissions associated with a user.

This is how IBM MQ has previously worked, and is the default value.

#### **MQLDAP\_AUTHORMD\_SEARCHGRP**

A group entry in the LDAP repository contains an attribute listing the Distinguished Name of all the users belonging to that group. Membership is indicated by the attribute defined in [FindGroup](#). This value is typically *member* or *uniqueMember*.

#### **MQLDAP\_AUTHORMD\_SEARCHUSR**

A user entry in the LDAP repository contains an attribute listing the Distinguished Name of all the groups to which the specified user belongs. The attribute to query is defined by the [FindGroup](#) value, typically *memberOf*.

#### **MQLDAP\_AUTHORMD\_SRCHGRPSN**

A group entry in the LDAP repository contains an attribute listing the short user name of all the users belonging to that group. The attribute in the user record that contains the short user name is specified by [ShortUser](#).

Membership is indicated by the attribute defined in [FindGroup](#). This value is typically *memberUid*.

**Note:** This authorization method should only be used if all user short names are distinct.

Many LDAP servers use an attribute of the group object to determine group membership and you should, therefore, set this value to *MQLDAP\_AUTHORMD\_SEARCHGRP*.

Microsoft Active Directory typically stores group memberships as a user attribute. The IBM Tivoli Directory Server supports both methods.

In general, retrieving memberships through a user attribute will be faster than searching for groups that list the user as a member.

### **BaseDNGroup (MQCFST)**

In order to be able to find group names, this parameter must be set with the base DN to search for groups in the LDAP server (parameter identifier: MQCA\_LDAP\_BASE\_DN\_GROUPS).

The maximum length is MQ\_LDAP\_BASE\_DN\_LENGTH.

### BaseDNUser (MQCFST)

In order to be able to find the short user name attribute (see [ShortUser](#)) this parameter must be set with the base DN to search for users within the LDAP server (parameter identifier: MQCA\_LDAP\_BASE\_DN\_USERS).

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_LDAP* and is mandatory.

The maximum length is MQ\_LDAP\_BASE\_DN\_LENGTH.

### Checkclient (MQCFIN)

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_OS* or *MQAIT\_IDPW\_LDAP* (parameter identifier: MQIA\_CHECK\_CLIENT\_BINDING). The possible values are:

#### MQCHK\_NONE

Switches off checking.


#### MQCHK\_OPTIONAL

Ensures that if a user ID and password are provided by an application, they are a valid pair, but that it is not mandatory to provide them. This option might be useful during migration, for example.

#### MQCHK\_REQUIRED

Requires that all applications provide a valid user ID and password.

#### MQCHK\_REQUIRED\_ADMIN

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the OPTIONAL setting.  (This setting is not allowed on z/OS systems.)

A privileged user is one that has full administrative authorities for IBM MQ. See [Privileged users](#) for more information.

### Checklocal (MQCFIN)

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_OS* or *MQAIT\_IDPW\_LDAP* (parameter identifier: MQIA\_CHECK\_LOCAL\_BINDING). The possible values are:

#### MQCHK\_NONE


Switches off checking.

#### MQCHK\_OPTIONAL


Ensures that if a user ID and password are provided by an application, they are a valid pair, but that it is not mandatory to provide them. This option might be useful during migration, for example.

#### MQCHK\_REQUIRED

Requires that all applications provide a valid user ID and password.

 If your user ID has UPDATE access to the BATCH profile in the MQCONN class, you can treat **MQCHK\_REQUIRED** as if it is **MQCHK\_OPTIONAL**. That is, you do not have to supply a password, but if you do, the password must be the correct one.

#### MQCHK\_REQUIRED\_ADMIN

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the OPTIONAL setting.  (This setting is not allowed on z/OS systems.)

A privileged user is one that has full administrative authorities for IBM MQ. See [Privileged users](#) for more information.

### ClassGroup (MQCFST)

The LDAP object class used for group records in the LDAP repository (parameter identifier: MQCA\_LDAP\_GROUP\_OBJECT\_CLASS).

If the value is blank, **groupOfNames** is used.

Other commonly used values include *groupOfUniqueNames* or *group*.

The maximum length is MQ\_LDAP\_CLASS\_LENGTH.

### **Classuser (MQCFST)**

The LDAP object class used for user records in the LDAP repository (parameter identifier MQCA\_LDAP\_USER\_OBJECT\_CLASS).

If blank, the value defaults to *inetOrgPerson*, which is generally the value needed.

For Microsoft Active Directory, the value you require required is often *user*.

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_LDAP*.

### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **FailureDelay (MQCFIN)**

When a user ID and password are provided for connection authentication, and the authentication fails due to the user ID or password being incorrect, this is the delay, in seconds, before the failure is returned to the application (parameter identifier: MQIA\_AUTHENTICATION\_FAIL\_DELAY).

This can aid in avoiding busy loops from an application that simply retries, continuously, after receiving a failure.

The value must be in the range 0 - 60 seconds. The default value is 1.

This parameter is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_OS* or *MQAIT\_IDPW\_LDAP*.

### **FindGroup (MQCFST)**

Name of the attribute used within an LDAP entry to determine group membership (parameter identifier: MQCA\_LDAP\_FIND\_GROUP\_FIELD).

When AuthorizationMethod = *MQLDAP\_AUTHORMD\_SEARCHGRP*, this attribute is typically set to *member* or *uniqueMember*.

When AuthorizationMethod = *MQLDAP\_AUTHORMD\_SEARCHUSR*, this attribute is typically set to *memberOf*.

When AuthorizationMethod = *MQLDAP\_AUTHORMD\_SRCHGRPSN*, this attribute is typically set to *memberUid*.

When left blank, if:

- AuthorizationMethod = *MQLDAP\_AUTHORMD\_SEARCHGRP*, this attribute defaults to *memberOf*.
- AuthorizationMethod = *MQLDAP\_AUTHORMD\_SEARCHUSR*, this attribute defaults to *member*.
- AuthorizationMethod = *MQLDAP\_AUTHORMD\_SRCHGRPSN*, this attribute defaults to *memberUid*.

The maximum length is MQ\_LDAP\_FIELD\_LENGTH.

### **GroupField (MQCFST)**

LDAP attribute that represents a simple name for the group (parameter identifier: MQCA\_LDAP\_GROUP\_ATTR\_FIELD).

If the value is blank, commands like `setmqaut` must use a qualified name for the group. The value can either be a full DN, or a single attribute.

The maximum length is `MQ_LDAP_FIELD_LENGTH`.

### GroupNesting (MQCFIN)

Whether groups are members of other groups (parameter identifier: `MQIA_LDAP_NESTGRP`). The values can be:

#### MQLDAP\_NESTGRP\_NO

Only the initially discovered groups are considered for authorization.

#### MQLDAP\_NESTGRP\_YES

The group list is searched recursively to enumerate all the groups to which a user belongs.

The group's Distinguished Name is used when searching the group list recursively, regardless of the authorization method selected in [AuthorizationMethod](#).

### LDAPPassword (MQCFST)

The LDAP password (parameter identifier: `MQCA_LDAP_PASSWORD`).

This parameter is relevant only when **AuthInfoType** is set to `MQAIT_CRL_LDAP` or `MQAIT_IDPW_LDAP`.

The maximum length is `MQ_LDAP_PASSWORD_LENGTH`.

### LDAPUserName (MQCFST)

The LDAP user name (parameter identifier: `MQCA_LDAP_USER_NAME`).

This parameter is relevant only when **AuthInfoType** is set to `MQAIT_CRL_LDAP` or `MQAIT_IDPW_LDAP`.

**Multi** On [Multiplatforms](#), the maximum length is `MQ_DISTINGUISHED_NAME_LENGTH`.

**z/OS** On z/OS, the maximum length is `MQ_SHORT_DNAME_LENGTH`.

### OCSPResponderURL (MQCFST)

The URL at which the OCSP responder can be contacted (parameter identifier: `MQCA_AUTH_INFO_OCSP_URL`).

This parameter is relevant only when **AuthInfoType** is set to `MQAIT_OCSP`, when it is required.

This field is case-sensitive. It must start with the string `http://` in lowercase. The rest of the URL might be case sensitive, depending on the OCSP server implementation.

The maximum length is `MQ_AUTH_INFO_OCSP_URL_LENGTH`.

### **z/OS** QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: `MQIA_QSG_DISP`). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

Table 189. QSGDisposition: Where objects are defined and how they behave		
QSGDisposition	Change	Copy, Create
<b>MQQSGD_COPY</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter <code>MQQSGD_COPY</code> . Any object residing in the shared repository, or any object defined using a command that had the parameter <code>MQQSGD_Q_MGR</code> , is not affected by this command.	The object is defined on the page set of the queue manager that executes the command. It uses the <code>MQQSGD_GROUP</code> object of the same name as the <i>ToAuthInfoName</i> object (for Copy) or the <i>AuthInfoName</i> object (for Create).

Table 189. QSGDisposition: Where objects are defined and how they behave (continued)

QSGDisposition	Change	Copy, Create
<b>MQQSGD_GROUP</b>	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group so that they refresh local copies on page set zero:</p> <pre>DEFINE AUTHINFO(name) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This definition is allowed only if the queue manager is in a queue sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group so that they make or refresh local copies on page set zero:</p> <pre>DEFINE AUTHINFO(name) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
<b>MQQSGD_PRIVATE</b>	<p>The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR, or MQQSGD_COPY. Any object residing in the shared repository is unaffected.</p>	<p>Not permitted.</p>
<b>MQQSGD_Q_MGR</b>	<p>The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This value is the default value.</p>	<p>The object is defined on the page set of the queue manager that executes the command. This value is the default value.</p>

**Replace (MQCFIN)**

Replace attributes (parameter identifier: MQIACF\_REPLACE).

If an Authentication Information object with the same name as AuthInfoName or ToAuthInfoName exists, it specifies whether it is to be replaced. The value can be any of the following values:

**MQRP\_YES**

Replace existing definition

**MQRP\_NO**

Do not replace existing definition

**SecureComms (MQCFIN)**

Whether connectivity to the LDAP server should be done securely using TLS (parameter identifier MQIA\_LDAP\_SECURE\_COMM).

**MQSECCOMM\_YES**

Connectivity to the LDAP server is made securely using TLS.

The certificate used is the default certificate for the queue manager, named in CERTLABL on the queue manager object, or if that is blank, the one described in [Digital certificate labels, understanding the requirements](#).

The certificate is located in the key repository specified in SSLKEYR on the queue manager object. A cipherspec will be negotiated that is supported by both IBM MQ and the LDAP server.

If the queue manager is configured to use SSLFIPS(YES) or SUITEB cipher specs, then this is taken account of in the connection to the LDAP server as well.

### **MQSECCOMM\_ANON**

Connectivity to the LDAP server is made securely using TLS just as for MQSECCOMM\_YES with one difference.

No certificate is sent to the LDAP server; the connection will be made anonymously. To use this setting, ensure that the key repository specified in SSLKEYR, on the queue manager object, does not contain a certificate marked as the default.

### **MQSECCOMM\_NO**

Connectivity to the LDAP server does not use TLS.

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_LDAP*.

### **ShortUser (MQCFST)**

A field in the user record to be used as a short user name in IBM MQ (parameter identifier MQCA\_LDAP\_SHORT\_USER\_FIELD).

This field must contain values of 12 characters or less. This short user name is used for the following purposes:

- If LDAP authentication is enabled, but LDAP authorization is not enabled, this is used as an operating system user ID for authorization checks. In this case, the attribute must represent an operating system user ID.
- If LDAP authentication and authorization are both enabled, this is used as the user ID carried with the message in order for the LDAP user name to be rediscovered when the user ID inside the message needs to be used.

For example, on another queue manager, or when writing report messages. In this case, the attribute does not need to represent an operating system user ID, but must be a unique string. An employee serial number is an example of a good attribute for this purpose.

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_LDAP* and is mandatory.

The maximum length is MQ\_LDAP\_FIELD\_LENGTH.

### **UserField (MQCFST)**

If the user ID provided by an application for authentication does not contain a qualifier for the field in the LDAP user record, that is, it does not contain an '=' sign, this attribute identifies the field in the LDAP user record that is used to interpret the provided user ID (parameter identifier MQCA\_LDAP\_USER\_ATTR\_FIELD).

This field can be blank. If this is the case, any unqualified user IDs use the [ShortUser](#) field to interpret the provided user ID.

The contents of this field will be concatenated with an '=' sign, together with the value provided by the application, to form the full user ID to be located in an LDAP user record. For example, the application provides a user of fred and this field has the value cn, then the LDAP repository will be searched for cn=fred.

The maximum length is MQ\_LDAP\_FIELD\_LENGTH.

## **Change, Copy, and Create CF Structure on z/OS**

The Change CF Structure PCF command changes existing CF application structures. The Copy and Create CF Structure commands create new CF application structures - the Copy command uses attribute values of an existing CF application structure.

**Note:** These commands are supported only on z/OS when the queue manager is a member of a queue sharing group.

The Change CF Structure (MQCMD\_CHANGE\_CF\_STRUC) command changes the specified attributes in a CF application structure. For any optional parameters that are omitted, the value does not change.

The Copy CF Structure (MQCMD\_COPY\_CF\_STRUC) command creates new CF application structure using, for attributes not specified in the command, the attribute values of an existing CF application structure.

The Create CF Structure (MQCMD\_CREATE\_CF\_STRUC) command creates a CF application structure. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

### **Required parameters (Change and Create CF Structure)**

#### **CFStrucName (MQCFST)**

The name of the CF application structure with backup and recovery parameters that you want to define (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

### **Required parameters (Copy CF Structure)**

#### **FromCFStrucName (MQCFST)**

The name of the CF application structure to be copied from (parameter identifier: MQCACF\_FROM\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

#### **ToCFStrucName (MQCFST)**

The name of the CF application structure to copy to (parameter identifier: MQCACF\_TO\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

### **Optional parameters (Change, Copy, and Create CF Structure)**

#### **CFConlos (MQCFIN)**

Specifies the action to be taken when a queue manager loses connectivity to the CF structure (parameter identifier: MQIA\_CF\_CONLOS).

The value can be any of the following values:

##### **MQCFCONLOS\_ASQMGR**

The action taken is based on the setting of the CFCONLOS queue manager attribute. This value is the default for newly created CF structure objects with CFLEVEL(5).

##### **MQCFCONLOS\_TERMINATE**

The queue manager terminates when connectivity to the structure is lost. This value is the default if the CF structure object is not at CFLEVEL(5), and for existing CF structure objects that are changed to CFLEVEL(5).

##### **MQCFCONLOS\_TOLERATE**

The queue manager tolerates loss of connectivity to the structure without terminating.

This parameter is only valid from CFLEVEL(5).

#### **CFLevel (MQCFIN)**

The functional capability level for this CF application structure (parameter identifier: MQIA\_CF\_LEVEL).

Specifies the functional capability level for the CF application structure. The value can be any of the following values:

**1**

A CF structure that can be "auto-created" by a queue manager at command level 520.

**2**

A CF structure at command level 520 that can only be created or deleted by a queue manager at command level 530 or greater.

**3**

A CF structure at command level 530. This *CFLevel* is required if you want to use persistent messages on shared queues, or for message grouping, or both. This level is the default *CFLevel* for queue managers at command level 600.

You can only increase the value of *CFLevel* to 3 if all the queue managers in the queue sharing group are at command level 530 or greater - this restriction is to ensure that there are no latent command level 520 connections to queues referencing the CF structure.

You can only decrease the value of *CFLevel* from 3 if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

**4**

This *CFLevel* supports all the *CFLevel* (3) functions. *CFLevel* (4) allows queues defined with CF structures at this level to have messages with a length greater than 63 KB.

Only a queue manager with a command level of 600 can connect to a CF structure at *CFLevel* (4).


You can only increase the value of *CFLevel* to 4 if all the queue managers in the queue sharing group are at command level 600 or greater.

You can only decrease the value of *CFLevel* from 4 if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

**5**

This *CFLevel* supports all the *CFLevel* (4) functions. *CFLevel* (5) allows persistent, and nonpersistent messages to be selectively stored in Db2 or shared message data sets.

Structures are required to be at CFLEVEL(5) to support toleration of loss of connectivity.

 For more information, see [Where are shared queue messages held?](#)

### **CFStrucDesc (MQCFST)**

The description of the CF structure (parameter identifier: MQCA\_CF\_STRUC\_DESC).

The maximum length is MQ\_CF\_STRUC\_DESC\_LENGTH.

### **DSBlock (MQCFIN)**

The logical block size for shared message data sets (parameter identifier: MQIACF\_CF\_SMDS\_BLOCK\_SIZE).

The unit in which shared message data set space is allocated to individual queues. The value can be any of the following values:

#### **MQDSB\_8K**

The logical block size is set to 8 K.

#### **MQDSB\_16K**

The logical block size is set to 16K.

#### **MQDSB\_32K**

The logical block size is set to 32 K.

#### **MQDSB\_64K**

The logical block size is set to 64 K.



**MQDSB\_128K**

The logical block size is set to 128 K.

**MQDSB\_256K**

The logical block size is set to 256 K.

**MQDSB\_512K**

The logical block size is set to 512 K.

**MQDSB\_1024K**

The logical block size is set to 1024 K.

**MQDSB\_1M**

The logical block size is set to 1 M.

Value can not be set unless CFLEVEL(5) is defined.

The default value is 256 K unless CFLEVEL is not 5. In this case a value of 0 is used.

**DSBufs (MQCFIN)**

The shared message data set buffers group (parameter identifier: MQIA\_CF\_SMDS\_BUFFERS).

Specifies the number of buffers to be allocated in each queue manager for accessing shared message data sets. The size of each buffer is equal to the logical block size.

A value in the range 1 - 9999.

Value can not be set unless CFLEVEL(5) is defined.

**DSEXPA ND (MQCFIN)**

The shared message data set expand option (parameter identifier: MQIACF\_CF\_SMDS\_EXPAND).

Specifies whether or not the queue manager should expand a shared message data set when it is nearly full, and further blocks are required in the data set. The value can be any of the following values:

**MQDSE\_YES**

The data set can be expanded.

**MQDSE\_NO**

The data set cannot be expanded.

**MQDSE\_DEFAULT**

Only returned on DISPLAY CFSTRUCT when not explicitly set

Value can not be set unless CFLEVEL(5) is defined.

**DSGroup (MQCFST)**

The shared message data set group name (parameter identifier: MQCACF\_CF\_SMDS\_GENERIC\_NAME).

Specifies a generic data set name to be used for the group of shared message data sets associated with this CF structure.

The string must contain exactly one asterisk (\*), which will be replaced with the queue manager name of up to 4 characters.

The maximum length of this parameter is 44 characters.

Value can not be set unless CFLEVEL(5) is defined.

**Offload (MQCFIN)**

Specifies whether offloaded message data is to be stored in a group of shared message data sets or in Db2 (parameter identifier: MQIA\_CF\_OFFLOAD).

The value can be:

**MQCFOFFLD\_DB2**

Large shared messages can be stored in Db2.

## **MQCFOFFLD\_SMDS**

Large shared messages can be stored in z/OS shared message data sets.

Value can not be set unless CFLEVEL(5) is defined.

For existing CF structure objects that are changed to CFLEVEL(5) the default is MQCFOFFLD\_DB2.

For newly created CF structure objects with CFLEVEL(5) the default is MQCFOFFLD\_SMDS.

For more information about the group of parameters (*OFFLDxSZ* and *OFFLDxTH*), see [Specifying offload options for shared message data sets](#)

### **OFFLD1SZ (MQCFST)**

The offload size property 1 (parameter identifier: MQCACF\_CF\_OFFLOAD\_SIZE1)

Specifies the first offload rule, based on upon message size and the coupling facility structure percentage use threshold. This property indicates the size of the messages to be offloaded. The property is specified as a string with values in the range 0K - 64K.

The default value is 32K. This property is used with *OFFLD1TH*.

Value can not be set unless CFLEVEL(5) is defined.

The value 64K indicates that the rule is not being used.

The maximum length is 3.

### **OFFLD2SZ (MQCFST)**

The offload size property 2 (parameter identifier: MQCACF\_CF\_OFFLOAD\_SIZE2)

Specifies the second offload rule, based on upon message size and the coupling facility structure percentage use threshold. This property indicates the size of the messages to be offloaded. The property is specified as a string with values in the range 0K - 64K.

The default value is 4K. This property is used with *OFFLD2TH*.

Value can not be set unless CFLEVEL(5) is defined.

The value 64K indicates that the rule is not being used.

The maximum length is 3.

### **OFFLD3SZ (MQCFST)**

The offload size property 3 (parameter identifier: MQCACF\_CF\_OFFLOAD\_SIZE3)

Specifies the third offload rule, based on upon message size and the coupling facility structure percentage use threshold. This property indicates the size of the messages to be offloaded. The property is specified as a string with values in the range 0K - 64K.

The default value is 0K. This property is used with *OFFLD3TH*.

Value can not be set unless CFLEVEL(5) is defined.

The value 64K indicates that the rule is not being used.

The maximum length is 3.

### **OFFLD1TH (MQCFIN)**

The offload threshold property 1 (parameter identifier: MQIA\_CF\_OFFLOAD\_THRESHOLD1)

Specifies the first offload rule, based on upon message size and the coupling facility structure percentage use threshold. This property indicates the coupling facility structure percentage full.

The default value is 70. This property is used with *OFFLD1SZ*.

Value can not be set unless CFLEVEL(5) is defined.

### **OFFLD2TH (MQCFIN)**

The offload threshold property 2 (parameter identifier: MQIA\_CF\_OFFLOAD\_THRESHOLD2)

Specifies the second offload rule, based on upon message size and the coupling facility structure percentage use threshold. This property indicates the coupling facility structure percentage full.

The default value is 80. This property is used with *OFFLD2SZ*.

Value can not be set unless CFLEVEL(5) is defined.

### **OFFLD3TH (MQCFIN)**

The offload threshold property 3 (parameter identifier: MQIA\_CF\_OFFLOAD\_THRESHOLD3)

Specifies the third offload rule, based on upon message size and the coupling facility structure percentage use threshold. This property indicates the coupling facility structure percentage full.

The default value is 90. This property is used with *OFFLD3SZ*.

Value can not be set unless CFLEVEL(5) is defined.

### **Recauto (MQCFIN)**

Specifies the automatic recovery action to be taken when a queue manager detects that the structure is failed, or when a queue manager loses connectivity to the structure and no systems in the sysplex have connectivity to the coupling facility that the structure is allocated in (parameter identifier: MQIA\_CF\_RECAUTO).

The value can be:

#### **MQRECAUTO\_YES**

The structure and associated shared message data sets which also need recovery are automatically recovered. This value is the default for newly created CF structure objects with CFLEVEL(5).

#### **MQRECAUTO\_NO**

The structure is not automatically recovered. This value is the default if the CF structure object is not at CFLEVEL(5), and for existing CF structure objects that are changed to CFLEVEL(5).

This parameter is only valid from CFLEVEL(5).

### **Recovery (MQCFIN)**

Specifies whether CF recovery is supported for the application structure (parameter identifier: MQIA\_CF\_RECOVER).

The value can be:

#### **MQCFR\_YES**

Recovery is supported.

#### **MQCFR\_NO**

Recovery is not supported.

### **Replace (MQCFIN)**

Replace attributes (parameter identifier: MQIACF\_REPLACE).

If a CF structure definition with the same name as *ToCFStrucName* exists, this value specifies whether it is to be replaced. The value can be any of the following values:

#### **MQRP\_YES**

Replace existing definition.

#### **MQRP\_NO**

Do not replace existing definition.

## ***Change, Copy, and Create Channel***

The Change Channel PCF command changes existing channel definitions. The Copy and Create Channel commands create new channel definitions - the Copy command uses attribute values of an existing channel definition.

The Change Channel (MQCMD\_CHANGE\_CHANNEL) command changes the specified attributes in a channel definition. For any optional parameters that are omitted, the value does not change.

The Copy Channel (MQCMD\_COPY\_CHANNEL) command creates new channel definition using, for attributes not specified in the command, the attribute values of an existing channel definition.

The Create Channel (MQCMD\_CREATE\_CHANNEL) command creates an IBM MQ channel definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager. If a system default channel exists for the type of channel being created, the default values are taken from there.

The following table shows the parameters that are applicable to each type of channel.

<i>Table 190. Change, Copy, Create Channel parameters</i>									
<b>Parameter</b>	<b>Sender</b>	<b>Server</b>	<b>Receiver</b>	<b>Requester</b>	<b>Client conn</b>	<b>Server conn</b>	<b>Cluster sender</b>	<b>Cluster receiver</b>	<b>AMQP</b>
<u>AMQPKeepAlive</u>									✓
<u>BatchHeartBeat</u>	✓	✓					✓	✓	
<u>BatchInterval</u>	✓	✓					✓	✓	
<u>BatchDataLimit</u>	✓	✓					✓	✓	
<u>BatchSize</u>	✓	✓	✓	✓			✓	✓	
<u>CertificateLabel</u>	✓	✓	✓	✓			✓	✓	✓
<u>ChannelDesc</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>ChannelMonitoring</u>	✓	✓	✓	✓		✓	✓	✓	
<u>ChannelStatistics</u>	✓	✓	✓	✓			✓	✓	
<u>ChannelName</u> (see footnote 1)	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>ChannelType</u> (see footnote 3)	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>ClientChannelWeight</u>					✓				
<u>ClusterName</u>							✓	✓	
<u>ClusterNameList</u>							✓	✓	
<u>CLWLChannelPriority</u>							✓	✓	
<u>CLWLChannelRank</u>							✓	✓	
<u>CLWLChannelWeight</u>							✓	✓	
 <u>CommandScope</u>	✓	✓	✓	✓	✓	✓	✓	✓	

Table 190. Change, Copy, Create Channel parameters (continued)

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	AMQP
<u>ConnectionAffinity</u>					✓				
<u>ConnectionName</u>	✓	✓		✓	✓		✓	✓	
<u>DataConversion</u>	✓	✓		✓	✓		✓	✓	
<u>DefaultChannelDisposition</u>	✓	✓	✓	✓		✓	✓	✓	
<u>DefReconnect</u>					✓				
<u>DiscInterval</u>	✓	✓				✓	✓	✓	
<u>FromChannelName</u> (see footnote 2)	✓	✓	✓	✓	✓	✓	✓	✓	
<u>HeaderCompression</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>HeartBeatInterval</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>KeepAliveInterval</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>LocalAddress</u>	✓	✓		✓	✓		✓	✓	✓
<u>LongRetryCount</u>	✓	✓					✓	✓	
<u>LongRetryInterval</u>	✓	✓					✓	✓	
<u>MaxInstances</u>						✓			✓
<u>MaxInstancesPerClient</u>						✓			
<u>MaxMsgLength</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>MCAName</u>	✓	✓		✓			✓		
<u>MCAType</u>	✓	✓		✓			✓	✓	
<u>MCAUserIdentifier</u>			✓	✓		✓		✓	✓
<u>MessageCompression</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>ModeName</u>	✓	✓		✓	✓		✓	✓	
<u>MsgExit</u>	✓	✓	✓	✓			✓	✓	
<u>MsgRetryCount</u>			✓	✓				✓	

Table 190. Change, Copy, Create Channel parameters (continued)








Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	AMQP
<u>MsgRetryExit</u>			✓	✓				✓	
<u>MsgRetryInterval</u>			✓	✓				✓	
<u>MsgRetryUserData</u>			✓	✓				✓	
<u>MsgUserData</u>	✓	✓	✓	✓			✓	✓	
<u>NetworkPriority</u>								✓	
<u>NonPersistentMsgSpeed</u>	✓	✓	✓	✓			✓	✓	
<u>Password</u>	✓	✓		✓	✓		✓		
<u>Port</u>									✓
<u>PropertyControl</u>	✓	✓					✓	✓	
<u>PutAuthority</u>			✓	✓		✓ "4" on page 1015		✓	
<u>QMgrName</u>					✓				
 <u>z/OS</u>  <u>z/OS</u> <u>QSGDisposition</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>ReceiveExit</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>ReceiveUserData</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>Replace</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SecurityExit</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SecurityUserData</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SendExit</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SendUserData</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>SeqNumberWrap</u>	✓	✓	✓	✓			✓	✓	
<u>SharingConversations</u>					✓	✓			
<u>ShortRetryCount</u>	✓	✓					✓	✓	
<u>ShortRetryInterval</u>	✓	✓					✓	✓	

Table 190. Change, Copy, Create Channel parameters (continued)

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	AMQP
 <u>SPLProtection</u>	✓	✓	✓	✓					
<u>SSLCipherSpec</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<u>SSLClientAuth</u>		✓	✓	✓		✓		✓	✓
<u>SSLPeerName</u>	✓	✓	✓	✓	✓	✓	✓	✓	✓
  <u>TemporaryModelQName</u>									✓
  <u>TemporaryQPrefix</u>									✓
<u>ToChannelName</u> (see footnote 2)	✓	✓	✓	✓	✓	✓	✓	✓	
<u>TpName</u>	✓	✓		✓	✓	✓	✓	✓	
<u>TpRoot</u>									✓
<u>TransportType</u>	✓	✓	✓	✓	✓	✓	✓	✓	
<u>UseCltId</u>									✓
<u>UseDLQ</u>	✓	✓	✓	✓			✓	✓	
<u>UserIdentifier</u>	✓	✓		✓	✓		✓		
<u>XmitQName</u>	✓	✓							

**Note:**

1. Required parameter on Change and Create Channel commands.
2. Required parameter on Copy Channel command.
3. Required parameter on Change, Create, and Copy Channel commands.
4. PUTAUT is valid for a channel type of SVRCONN on z/OS only.
5. Required parameter on Create Channel command if TtpType is TCP.
6. Required parameter on Create Channel command for a channel type of MQTT.

**Required parameters (Change, Create Channel)**

**ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

Specifies the name of the channel definition to be changed, or created

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

This parameter is required on all types of channel; on a CLUSSDR it can be different from on the other channel types. If your convention for naming channels includes the name of the queue manager, you can make a CLUSSDR definition using the +QMNAME+ construction, and IBM MQ substitutes the correct repository queue manager name in place of +QMNAME+. This facility applies only to IBM i, AIX, Linux, and Windows only. See [Configuring a queue manager cluster](#) for more details.

**Multi** On CLUSRCVR channels when using automatic cluster setup, this parameter can use some additional inserts:

- +AUTOCL+ resolves to the automatic cluster name
- +QMNAME+ resolves to the local queue manager name.

When using these inserts, both the unexpanded string and the string with the replaced values must fit inside the maximum size of the field. If there are configured automatic cluster full repositories in the AutoCluster configuration, the channel name must also fit in the maximum channel name length when +QMNAME+ is replaced with each of the configured full repository names.

### ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

Specifies the type of the channel being changed, copied, or created. The value can be any of the following values:

#### **MQCHT\_SENDER**

Sender.

#### **MQCHT\_SERVER**

Server.

#### **MQCHT\_RECEIVER**

Receiver.

#### **MQCHT\_REQUESTER**

Requester.

#### **MQCHT\_SVRCONN**

Server-connection (for use by clients).

#### **MQCHT\_CLNTCONN**

Client connection.

#### **MQCHT\_CLUSRCVR**

Cluster-receiver.

#### **MQCHT\_CLUSSDR**

Cluster-sender.

#### **MQCHT\_AMQP**

AMQP.

### Required parameters (Copy Channel)

#### FromChannelName (MQCFST)

From channel name (parameter identifier: MQCACF\_FROM\_CHANNEL\_NAME).

The name of the existing channel definition that contains values for the attributes that are not specified in this command.

**z/OS** On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD\_Q\_MGR or MQQSGD\_COPY to copy from. This parameter is ignored if a value of MQQSGD\_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToChannelName* and the disposition MQQSGD\_GROUP is searched for to copy from.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.



### **ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

Specifies the type of the channel being changed, copied, or created. The value can be any of the following values:

#### **MQCHT\_SENDER**

Sender.

#### **MQCHT\_SERVER**

Server.

#### **MQCHT\_RECEIVER**

Receiver.

#### **MQCHT\_REQUESTER**

Requester.

#### **MQCHT\_SVRCONN**

Server-connection (for use by clients).

#### **MQCHT\_CLNTCONN**

Client connection.

#### **MQCHT\_CLUSRCVR**

Cluster-receiver.

#### **MQCHT\_CLUSSDR**

Cluster-sender.

#### **MQCHT\_AMQP**

AMQP.

### **ToChannelName (MQCFST)**

To channel name (parameter identifier: MQCACF\_TO\_CHANNEL\_NAME).

The name of the new channel definition.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

Channel names must be unique; if a channel definition with this name exists, the value of *Replace* must be MQRP\_YES. The channel type of the existing channel definition must be the same as the channel type of the new channel definition otherwise it cannot be replaced.

## **Optional parameters (Change, Copy, and Create Channel)**

### **AMQPKeepAlive (MQCFIN)**

The AMQP channel keep alive interval (parameter identifier: MQIACH\_AMQP\_KEEP\_ALIVE).

The keep alive time for an AMQP channel in milliseconds. If the AMQP client has not sent any frames within the keep alive interval, then the connection is closed with a `amqp:resource-limit-exceeded` AMQP error condition.

This parameter is valid only for *ChannelType* values of MQCHT\_AMQP.

### **BatchHeartbeat (MQCFIN)**

The batch heartbeat interval (parameter identifier: MQIACH\_BATCH\_HB).

Batch heartbeating allows sender-type channels to determine whether the remote channel instance is still active, before going in-doubt. The value can be in the range 0 - 999999. A value of 0 indicates that batch heart-beating is not to be used. Batch heartbeat is measured in milliseconds.

This parameter is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR.

### **BatchInterval (MQCFIN)**

Batch interval (parameter identifier: MQIACH\_BATCH\_INTERVAL). The approximate time in milliseconds that a channel keeps a batch open, if fewer than `BatchSize` messages or `BatchDataLimit` bytes have been transmitted in the current batch.

The batch is terminated when one of the following conditions is met:

- `BatchSize` messages have been sent.
- `BatchDataLimit` bytes have been sent.
- The transmission queue is empty and `BatchInterval` milliseconds have elapsed since the start of the batch.

`BatchInterval` must be in the range 0 - 999999999. A value of zero means that the batch is terminated as soon as the transmission queue becomes empty, or the `BatchSize` or `BatchDataLimit` is reached.

This parameter applies only to channels with a `ChannelType` of: `MQCHT_SENDER`, `MQCHT_SERVER`, `MQCHT_CLUSSDR`, or `MQCHT_CLUSRCVR`.

### **BatchDataLimit (MQCFIN)**

Batch data limit (parameter identifier: `MQIACH_BATCH_DATA_LIMIT`).

The limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point. A sync point is taken after the message that caused the limit to be reached has flowed across the channel. A value of zero in this attribute means that no data limit is applied to batches over this channel.

The value must be in the range 0 - 999999. The default value is 5000.

The **BATCHLIM** parameter is supported on all platforms.

This parameter only applies to channels with a `ChannelType` of `MQCHT_SENDER`, `MQCHT_SERVER`, `MQCHT_CLUSRCVR`, or `MQCHT_CLUSSDR`.

### **BatchSize (MQCFIN)**

Batch size (parameter identifier: `MQIACH_BATCH_SIZE`).

The maximum number of messages that must be sent through a channel before a checkpoint is taken.

The batch size which is used is the lowest of the following:

- The *BatchSize* of the sending channel
- The *BatchSize* of the receiving channel
- The maximum number of uncommitted messages at the sending queue manager
- The maximum number of uncommitted messages at the receiving queue manager

The maximum number of uncommitted messages is specified by the **MaxUncommittedMsgs** parameter of the Change Queue Manager command.

Specify a value in the range 1 - 9999.

This parameter is not valid for channels with a `ChannelType` of `MQCHT_SVRCONN` or `MQCHT_CLNTCONN`.

### **CertificateLabel (MQCFST)**

Certificate label (parameter identifier: `MQCA_CERT_LABEL`).

Certificate label for this channel to use.

The label identifies which personal certificate in the key repository is sent to the remote peer. If this attribute is blank, the certificate is determined by the queue manager **CertificateLabel** parameter.

Note that inbound channels (including receiver, requester, cluster-receiver, unqualified server, and server-connection channels) only send the configured certificate if the IBM MQ version of the remote peer fully supports certificate label configuration, and the channel is using a TLS CipherSpec.

An unqualified server channel is one that does not have the **ConnectionName** field set.

In all other cases, the queue manager **CertificateLabel** parameter determines the certificate sent. In particular, the following only ever receive the certificate configured by the

**CertificateLabel** parameter of the queue manager, regardless of the channel-specific label setting:

- All current Java and JMS clients.
- Versions of IBM MQ prior to IBM MQ 8.0.

### **ChannelDesc (MQCFST)**

Channel description (parameter identifier: MQCACH\_DESC).

The maximum length of the string is MQ\_CHANNEL\_DESC\_LENGTH.

Use characters from the character set, identified by the coded character set identifier (CCSID) for the message queue manager on which the command is executing, to ensure that the text is translated correctly.

### **ChannelMonitoring (MQCFIN)**

Online monitoring data collection (parameter identifier: MQIA\_MONITORING\_CHANNEL).

Specifies whether online monitoring data is to be collected and, if so, the rate at which the data is collected. The value can be any of the following values:

#### **MQMON\_OFF**

Online monitoring data collection is turned off for this channel.

#### **MQMON\_Q\_MGR**

The value of the queue manager's **ChannelMonitoring** parameter is inherited by the channel.

#### **MQMON\_LOW**

If the value of the queue manager's *ChannelMonitoring* parameter is not MQMON\_NONE, online monitoring data collection is turned on, with a low rate of data collection, for this channel.

#### **MQMON\_MEDIUM**

If the value of the queue manager's *ChannelMonitoring* parameter is not MQMON\_NONE, online monitoring data collection is turned on, with a moderate rate of data collection, for this channel.

#### **MQMON\_HIGH**

If the value of the queue manager's *ChannelMonitoring* parameter is not MQMON\_NONE, online monitoring data collection is turned on, with a high rate of data collection, for this channel.

### **ChannelStatistics (MQCFIN)**

Statistics data collection (parameter identifier: MQIA\_STATISTICS\_CHANNEL).

Specifies whether statistics data is to be collected and, if so, the rate at which the data is collected. The value can be:

#### **MQMON\_OFF**

Statistics data collection is turned off for this channel.

#### **MQMON\_Q\_MGR**

The value of the queue manager's **ChannelStatistics** parameter is inherited by the channel.

#### **MQMON\_LOW**


If the value of the queue manager's *ChannelStatistics* parameter is not MQMON\_NONE, online monitoring data collection is turned on, with a low rate of data collection, for this channel.

#### **MQMON\_MEDIUM**

If the value of the queue manager's *ChannelStatistics* parameter is not MQMON\_NONE, online monitoring data collection is turned on, with a moderate rate of data collection, for this channel.

#### **MQMON\_HIGH**

If the value of the queue manager's *ChannelStatistics* parameter is not MQMON\_NONE, online monitoring data collection is turned on, with a high rate of data collection, for this channel.

 On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

**ClientChannelWeight (MQCFIN)**

Client Channel Weight (parameter identifier: MQIACH\_CLIENT\_CHANNEL\_WEIGHT).

The client channel weighting attribute is used so client channel definitions can be selected at random, with the larger weightings having a higher probability of selection, when more than one suitable definition is available.

Specify a value in the range 0 - 99. The default is 0.

This parameter is only valid for channels with a *ChannelType* of MQCHT\_CLNTCONN

**ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

The name of the cluster to which the channel belongs.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT\_CLUSSDR
- MQCHT\_CLUSRCVR

Only one of the values of *ClusterName* and *ClusterNameList* can be nonblank; the other must be blank.

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

**ClusterNameList (MQCFST)**

Cluster namelist (parameter identifier: MQCA\_CLUSTER\_NAMELIST).

The name, of the namelist, that specifies a list of clusters to which the channel belongs.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT\_CLUSSDR
- MQCHT\_CLUSRCVR

Only one of the values of *ClusterName* and *ClusterNameList* can be nonblank; the other must be blank.

**CLWLChannelPriority (MQCFIN)**

Channel priority for the purposes of cluster workload distribution (parameter identifier: MQIACH\_CLWL\_CHANNEL\_PRIORITY).

Specify a value in the range 0 - 9 where 0 is the lowest priority and 9 is the highest.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT\_CLUSSDR
- MQCHT\_CLUSRCVR

**CLWLChannelRank (MQCFIN)**

Channel rank for the purposes of cluster workload distribution (parameter identifier: MQIACH\_CLWL\_CHANNEL\_RANK).

Specify a value in the range 0 - 9 where 0 is the lowest priority and 9 is the highest.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT\_CLUSSDR
- MQCHT\_CLUSRCVR

**CLWLChannelWeight (MQCFIN)**

Channel weighting for the purposes of cluster workload distribution (parameter identifier: MQIACH\_CLWL\_CHANNEL\_WEIGHT).

Specify a weighting for the channel for use in workload management. Specify a value in the range 1 - 99 where 1 is the lowest priority and 99 is the highest.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT\_CLUSSDR
- MQCHT\_CLUSRCVR



### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### ConnectionAffinity (MQCFIN)

Channel Affinity (parameter identifier: MQIACH\_CONNECTION\_AFFINITY)

The channel affinity attribute specifies whether client applications that connect multiple times using the same queue manager name, use the same client channel. The value can be any of the following values:

#### MQCAFTY\_PREFERRED

The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions based on the weighting with any zero ClientChannelWeight definitions first in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful nonzero ClientChannelWeight definitions are moved to the end of the list. Zero ClientChannelWeight definitions remain at the start of the list and are selected first for each connection. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT has been modified since the list was created. Each client process with the same host name creates the same list.

This value is the default value.

#### MQCAFTY\_NONE

The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process independently select an applicable definition based on the weighting with any applicable zero ClientChannelWeight definitions selected first in alphabetical order. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT has been modified since the list was created.

This parameter is only valid for channels with a ChannelType of MQCHT\_CLNTCONN.

### ConnectionName (MQCFST)

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).



On CLUSRCVR channels when using automatic cluster setup, this parameter can use some additional inserts:

- +AUTOCL+ resolves to the automatic cluster name
- +QMNAME+ resolves to the local queue manager name

In addition, any variable configured at queue manager create time, see the `crtmqm -iv` option, can be used surrounded by '+', for example +CONNNAME+. When using these inserts, both the unexpanded inserts and the expanded values must fit inside the field maximum size.

**Multi** On Multiplatforms, the maximum length of the string is 264.

**z/OS** On z/OS, the maximum length of the string is 48.

Specify *ConnectionName* as a comma-separated list of names of machines for the stated *TransportType*. Typically, only one machine name is required. You can provide multiple machine names to configure multiple connections with the same properties. The connections are tried in the order they are specified in the connection list until a connection is successfully established. If no connection is successful, the channel starts to try processing again. Connection lists are an alternative to queue manager groups to configure connections for reconnectable clients, and also to configure channel connections to multi-instance queue managers.

Specify the name of the machine as required for the stated *TransportType*:

- For MQXPT\_LU62 on IBM i, AIX and Linux, specify the name of the CPI-C communications side object. On Windows specify the CPI-C symbolic destination name.

**z/OS** On z/OS, there are two forms in which to specify the value:

#### Logical unit name

The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. This name can be specified in one of three forms:

Form	Example
luname	IGY12355
luname/TPname	IGY12345/APING
luname/TPname/modename	IGY12345/APINGD/#INTER

For the first form, the TP name and mode name must be specified for the *TpName* and *ModeName* parameters; otherwise these parameters must be blank.

**Note:** For client-connection channels, only the first form is allowed.

#### Symbolic name

The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The **TpName** and **ModeName** parameters must be blank.

**Note:** For cluster-receiver channels, the side information is on the other queue managers in the cluster. Alternatively, in this case it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

The specified or implied LU name can be that of a VTAM generic resources group.

- For MQXPT\_TCP, you can specify a connection name, or a connection list, containing the host name or the network address of the remote machine. Separate connection names in a connection list with commas.

**z/OS** On z/OS, the connection name can include the IP\_name of a z/OS dynamic DNS group or a network dispatcher input port. Do not include this parameter for channels with a *ChannelType* value of MQCHT\_CLUSSDR.

**Multi** On Multiplatforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

(1415)

The generated **CONNNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

- For MQXPT\_NETBIOS specify the NetBIOS station name.
- For MQXPT\_SPX specify the 4 byte network address, the 6 byte node address, and the 2 byte socket number. These values must be entered in hexadecimal, with a period separating the network and node addresses. The socket number must be enclosed in brackets, for example:

```
0a0b0c0d.804abcde23a1(5e86)
```

If the socket number is omitted, the IBM MQ default value (5e86 hex) is assumed.

This parameter is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_REQUESTER, MQCHT\_CLNTCONN, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR.

**Note:** If you are using clustering between IPv6 -only and IPv4 -only queue managers, do not specify an IPv6 network address as the *ConnectionName* for cluster-receiver channels. A queue manager that is capable only of IPv4 communication is unable to start a cluster sender channel definition that specifies the *ConnectionName* in IPv6 hexadecimal form. Consider, instead, using host names in a heterogeneous IP environment.

### DataConversion (MQCFIN)

Whether sender must convert application data (parameter identifier: MQIACH\_DATA\_CONVERSION).

This parameter is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR.

The value can be any of the following values:

#### MQCDC\_NO\_SENDER\_CONVERSION

No conversion by sender.

#### MQCDC\_SENDER\_CONVERSION

Conversion by sender.

### DefaultChannelDisposition (MQCFIN)

Intended disposition of the channel when activated or started (parameter identifier: MQIACH\_DEF\_CHANNEL\_DISP).

This parameter applies to z/OS only.

The value can be any of the following values:

#### MQCHLD\_PRIVATE

The intended use of the object is as a private channel.

This value is the default value.

#### MQCHLD\_FIXSHARED

The intended use of the object is as a fixshared channel.

#### MQCHLD\_SHARED

The intended use of the object is as a shared channel.

### DefReconnect (MQCFIN)

Client channel default reconnection option (parameter identifier: MQIACH\_DEF\_RECONNECT).

The default automatic client reconnection option. You can configure an IBM MQ MQI client to automatically reconnect a client application. The IBM MQ MQI client tries to reconnect to a queue manager after a connection failure. It tries to reconnect without the application client issuing an MQCONN or MQCONNX MQI call.

#### MQRCN\_NO

MQRCN\_NO is the default value.

Unless overridden by **MQCONNX**, the client is not reconnected automatically.

**MQRCN\_YES**

Unless overridden by **MQCONN**, the client reconnects automatically.

**MQRCN\_Q\_MGR**

Unless overridden by **MQCONN**, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as MQCNO\_RECONNECT\_Q\_MGR.

**MQRCN\_DISABLED**

Reconnection is disabled, even if requested by the client program using the **MQCONN** MQI call.

*Table 192. Automatic reconnection depends on the values set in the application and in the channel definition*

DefReconnect	Reconnection options set in the application			
	MQCNO_RECONNE CT	MQCNO_RECONNE CT_Q_MGR	MQCNO_RECONNE CT_AS_DEF	MQCNO_RECONNE CT_DISABLED
MQRCN_NO	YES	QMGR	NO	NO
MQRCN_YES	YES	QMGR	YES	NO
MQRCN_Q_MGR	YES	QMGR	QMGR	NO
MQRCN_DISABLED	NO	NO	NO	NO

This parameter is valid only for a *ChannelType* value of MQCHT\_CLNTCONN.

**DiscInterval (MQCFIN)**

Disconnection interval (parameter identifier: MQIACH\_DISC\_INTERVAL).

This interval defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before terminating the channel. A value of zero causes the message channel agent to wait indefinitely.

Specify a value in the range 0 - 999 999.

This parameter is valid only for *ChannelType* values of MQCHT\_SENDER MQCHT\_SERVER, MQCHT\_SVRCONN, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR.

For server-connection channels using the TCP protocol, this interval is the minimum time in seconds for which the server-connection channel instance remains active without any communication from its partner client. A value of zero disables this disconnect processing. The server-connection inactivity interval only applies between MQ API calls from a client, so no client is disconnected during an extended MQGET with wait call. This attribute is ignored for server-connection channels using protocols other than TCP.

**HeaderCompression (MQCFIL)**

Header data compression techniques supported by the channel (parameter identifier: MQIACH\_HDR\_COMPRESSION).

The list of header data compression techniques supported by the channel. For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The mutually supported compression techniques of the channel are passed to the message exit of the sending channel where the compression technique used can be altered on a per message basis. Compression alters the data passed to send and receive exits.

Specify one or more of:

**MQCOMPRESS\_NONE**

No header data compression is performed. This value is the default value.

**MQCOMPRESS\_SYSTEM**

Header data compression is performed.



### HeartbeatInterval (MQCFIN)

Heartbeat interval (parameter identifier: MQIACH\_HB\_INTERVAL).

The interpretation of this parameter depends on the channel type, as follows:

- For a channel type of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_RECEIVER, MQCHT\_REQUESTER, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR, this interval is the time in seconds between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. This interval gives the receiving MCA the opportunity to quiesce the channel. To be useful, *HeartbeatInterval* must be less than *DiscInterval*. However, the only check is that the value is within the permitted range.

This type of heartbeat is supported on the following platforms: IBM i, z/OS, AIX, Linux, and Windows.

- For a channel type of MQCHT\_CLNTCONN or MQCHT\_SVRCONN, this interval is the time in seconds between heartbeat flows passed from the server MCA when that MCA has issued an MQGET call with the MQGMO\_WAIT option on behalf of a client application. This interval allows the server MCA to handle situations where the client connection fails during an MQGET with MQGMO\_WAIT.

This type of heartbeat is supported on all platforms.

The value must be in the range 0 - 999 999. A value of 0 means that no heartbeat exchange occurs. The value that is used is the larger of the values specified at the sending side and receiving side.

### KeepAliveInterval (MQCFIN)

KeepAlive interval (parameter identifier: MQIACH\_KEEP\_ALIVE\_INTERVAL).

Specifies the value passed to the communications stack for KeepAlive timing for the channel.

For this attribute to be effective, TCP/IP keepalive must be enabled. On z/OS, you enable TCP/IP keepalive by issuing the Change Queue Manager command with a value of MQTCPKEEP in the *TCPKeepAlive* parameter; if the *TCPKeepAlive* queue manager parameter has a value of MQTCPKEEP\_NO, the value is ignored, and the KeepAlive facility is not used. On other platforms, TCP/IP keepalive is enabled when the KEEPALIVE=YES parameter is specified in the TCP stanza in the distributed queuing configuration file, qm.ini, or through the IBM MQ Explorer. Keepalive must also be enabled within TCP/IP itself, using the TCP profile configuration data set.

Although this parameter is available on all platforms, its setting is implemented only on z/OS. On platforms other than z/OS, you can access and modify the parameter, but it is only stored and forwarded; there is no functional implementation of the parameter. This parameter is useful in a clustered environment where a value set in a cluster-receiver channel definition on AIX, for example, flows to (and is implemented by) z/OS queue managers that are in, or join, the cluster.

Specify either:


#### *integer*

The KeepAlive interval to be used, in seconds, in the range 0 - 99 999. If you specify a value of 0, the value used is that specified by the INTERVAL statement in the TCP profile configuration data set.

#### MQKAI\_AUTO

The KeepAlive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated *HeartbeatInterval* is greater than zero, KeepAlive interval is set to that value plus 60 seconds.
- If the negotiated *HeartbeatInterval* is zero, the value used is that specified by the INTERVAL statement in the TCP profile configuration data set.

 On [Multiplatforms](#), if you need the functionality provided by the **KeepAliveInterval** parameter, use the **HeartBeatInterval** parameter.

### LocalAddress (MQCFST)

Local communications address for the channel (parameter identifier: MQCACH\_LOCAL\_ADDRESS).

The maximum length of the string is MQ\_LOCAL\_ADDRESS\_LENGTH.

The value that you specify depends on the transport type (*TransportType*) to be used:

### TCP/IP

The value is the optional IP address and optional port or port range to be used for outbound TCP/IP communications. The format for this information is as follows:

```
LOCLADDR([ip-addr][(low-port[,high-port])][, [ip-addr][(low-port[,high-port])]])
```

where *ip-addr* is specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric form, and *low-port* and *high-port* are port numbers enclosed in parentheses. All are optional.

Specify `[, [ip-addr][(low-port[,high-port])]]` multiple times for each additional local address. Use multiple local addresses if you want to specify a specific subset of local network adapters. You can also use `[, [ip-addr][(low-port[,high-port])]]` to represent a particular local network address on different servers that are part of a multi-instance queue manager configuration.

### All Others

The value is ignored; no error is diagnosed.

Use this parameter if you want a channel to use a particular IP address, port, or port range for outbound communications. This parameter is useful when a machine is connected to multiple networks with different IP addresses.

Examples of use

Value	Meaning
9.20.4.98	Channel binds to this address locally
9.20.4.98 (1000)	Channel binds to this address and port 1000 locally
9.20.4.98 (1000,2000)	Channel binds to this address and uses a port in the range 1000 - 2000 locally
(1000)	Channel binds to port 1000 locally
(1000,2000)	Channel binds to a port in the range 1000 - 2000 locally

This parameter is valid for the following channel types:

- MQCHT\_SENDER
- MQCHT\_SERVER
- MQCHT\_REQUESTER
- MQCHT\_CLNTCONN
- MQCHT\_CLUSRCVR
- MQCHT\_CLUSSDR

### Note:

- Do not confuse this parameter with *ConnectionName*. The *LocalAddress* parameter specifies the characteristics of the local communications; the *ConnectionName* parameter specifies how to reach a remote queue manager.

### LongRetryCount (MQCFIN)

Long retry count (parameter identifier: MQIACH\_LONG\_RETRY).

When a sender or server channel is attempting to connect to the remote machine, and the count specified by *ShortRetryCount* has been exhausted, this count specifies the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by *LongRetryInterval*.

If this count is also exhausted without success, an error is logged to the operator, and the channel is stopped. The channel must later be restarted with a command (it is not started automatically by the channel initiator), and it then makes only one attempt to connect, as it is assumed that the problem has now been cleared by the administrator. The retry sequence is not carried out again until after the channel has successfully connected.

Specify a value in the range 0 - 999 999 999.

This parameter is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR.

### **LongRetryInterval (MQCFIN)**

Long timer (parameter identifier: MQIACH\_LONG\_TIMER).

Specifies the long retry wait interval for a sender or server channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by *ShortRetryCount* has been exhausted.

The time is approximate; zero means that another connection attempt is made as soon as possible.

Specify a value in the range 0 - 999 999. Values exceeding this value are treated as 999 999.

This parameter is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR.

### **MaxInstances (MQCFIN)**

Maximum number of simultaneous instances of a server-connection channel or an AMQP channel (parameter identifier: MQIACH\_MAX\_INSTANCES).

Specify a value in the range 0 - 999 999 999.

The default value is 999 999 999.

A value of zero indicates that no client connections are allowed on the channel.

If the value is reduced below the number of instances of the server-connection channel that are currently running, the running channels are not affected. This parameter applies even if the value is zero. However, if the value is reduced below the number of instances of the server-connection channel that are currently running, then new instances cannot be started until sufficient existing instances have ceased to run.

If an AMQP client attempts to connect to an AMQP channel, and the number of connected clients has reached MaxInstances, the channel closes the connection with a close frame. The close frame contains the following message: `amqp:resource-limit-exceeded`. If a client connects with an ID that is already connected (that is, it performs a client-takeover), and the client is permitted to take over the connection, the takeover will succeed regardless of whether the number of connected clients has reached MaxInstances.

This parameter is valid only for channels with a *ChannelType* value of MQCHT\_SVRCONN or MQCHT\_AMQP.

### **MaxInstancesPerClient (MQCFIN)**

Maximum number of simultaneous instances of a server-connection channel that can be started from a single client (parameter identifier: MQIACH\_MAX\_INSTS\_PER\_CLIENT). In this context, connections that originate from the same remote network address are regarded as coming from the same client.

Specify a value in the range 0 - 999 999 999.

The default value is 999 999 999.

A value of zero indicates that no client connections are allowed on the channel.

If the value is reduced below the number of instances of the server-connection channel that are currently running from individual clients, the running channels are not affected. This parameter applies even if the value is zero. However, if the value is reduced below the number of instances of the server-connection channel that are currently running from individual clients, new instances from those clients cannot start until sufficient existing instances have ceased to run.

This parameter is valid only for channels with a *ChannelType* value of MQCHT\_SVRCONN.

### MaxMsgLength (MQCFIN)

Maximum message length (parameter identifier: MQIACH\_MAX\_MSG\_LENGTH).

Specifies the maximum message length that can be transmitted on the channel. This value is compared with the value for the remote channel and the actual maximum is the lower of the two values.

The value zero means the maximum message length for the queue manager.

The lower limit for this parameter is 0. The maximum message length is 100 MB (104 857 600 bytes).

### MCAName (MQCFST)

Message channel agent name (parameter identifier: MQCACH\_MCA\_NAME).

**Note:** An alternative way of providing a user ID for a channel to run under is to use channel authentication records. With channel authentication records, different connections can use the same channel while using different credentials. If both MCAUSER on the channel is set and channel authentication records are used to apply to the same channel, the channel authentication records take precedence. The MCAUSER on the channel definition is only used if the channel authentication record uses USERSRC(CHANNEL). For more details, see [Channel authentication records](#)

This parameter is reserved, and if specified can be set only to blanks.


The maximum length of the string is MQ\_MCA\_NAME\_LENGTH.


This parameter is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_REQUESTER, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR.

### MCAType (MQCFIN)

Message channel agent type (parameter identifier: MQIACH\_MCA\_TYPE).

Specifies the type of the message channel agent program.

 On Multiplatforms, this parameter is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_REQUESTER, or MQCHT\_CLUSSDR.

 On z/OS, this parameter is valid only for a *ChannelType* value of MQCHT\_CLURCVR.

The value can be any of the following values:

#### MQMCAT\_PROCESS

Process.

#### MQMCAT\_THREAD

Thread.

### MCAUserIdentifier (MQCFST)

Message channel agent user identifier (parameter identifier: MQCACH\_MCA\_USER\_ID).

If this parameter is nonblank, it is the user identifier which is to be used by the message channel agent for authorization to access IBM MQ resources, including (if *PutAuthority* is MQPA\_DEFAULT) authorization to put the message to the destination queue for receiver or requester channels.

If it is blank, the message channel agent uses its default user identifier.

This user identifier can be overridden by one supplied by a channel security exit.

This parameter is not valid for channels with a *ChannelType* of MQCHT\_SDR, MQCHT\_SVR, MQCHT\_CLNTCONN, MQCHT\_CLUSSDR.

The maximum length of the MCA user identifier depends on the environment in which the MCA is running. MQ\_MCA\_USER\_ID\_LENGTH gives the maximum length for the environment for which your application is running. MQ\_MAX\_MCA\_USER\_ID\_LENGTH gives the maximum for all supported environments.

On Windows, you can optionally qualify a user identifier with the domain name in the following format:

```
user@domain
```

### **MessageCompression (MQCFIL)**

The list of message data compression techniques supported by the channel (parameter identifier: MQIACH\_MSG\_COMPRESSION). For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The mutually supported compression techniques of the channel are passed to the message exit of the sending channel where the compression technique used can be altered on a per message basis. Compression alters the data passed to send and receive exits.

Specify one or more of:

#### **MQCOMPRESS\_NONE**

No message data compression is performed. This value is the default value.

#### **MQCOMPRESS\_RLE**

Message data compression is performed using run-length encoding.

#### **MQCOMPRESS\_ZLIBFAST**

Message data compression is performed using ZLIB encoding with speed prioritized.

#### **MQCOMPRESS\_ZLIBHIGH**

Message data compression is performed using ZLIB encoding with compression prioritized.

#### **MQCOMPRESS\_ANY**

Any compression technique supported by the queue manager can be used. This value is only valid for receiver, requester, and server-connection channels.

### **ModeName (MQCFST)**

Mode name (parameter identifier: MQCACH\_MODE\_NAME).

This parameter is the LU 6.2 mode name.

The maximum length of the string is MQ\_MODE\_NAME\_LENGTH.

- On IBM i, AIX, Linux, and Windows, this parameter can be set only to blanks. The actual name is taken instead from the CPI-C Communications Side Object or (on Windows ) from the CPI-C symbolic destination name properties.

This parameter is valid only for channels with a *TransportType* of MQXPT\_LU62. It is not valid for receiver or server-connection channels.

### **MsgExit (MQCFSL)**

Message exit name (parameter identifier: MQCACH\_MSG\_EXIT\_NAME).

If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

For channels with a channel type (*ChannelType*) of MQCHT\_SVRCONN or MQCHT\_CLNTCONN, this parameter is accepted but ignored, since message exits are not invoked for such channels.

The format of the string is the same as for *SecurityExit*.

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

You can specify a list of exit names by using an MQCFSL structure instead of an MQCFST structure.

- The exits are invoked in the order specified in the list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.

- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all the exit names in the list (excluding trailing blanks in each name) must not exceed MQ\_TOTAL\_EXIT\_NAME\_LENGTH. An individual string must not exceed MQ\_EXIT\_NAME\_LENGTH.
- On z/OS, you can specify the names of up to eight exit programs.

#### **MsgRetryCount (MQCFIN)**

Message retry count (parameter identifier: MQIACH\_MR\_COUNT).

Specifies the number of times that a failing message must be retried.

Specify a value in the range 0 - 999 999 999.

This parameter is valid only for *ChannelType* values of MQCHT\_RECEIVER, MQCHT\_REQUESTER, or MQCHT\_CLUSRCVR.

#### **MsgRetryExit (MQCFST)**

Message retry exit name (parameter identifier: MQCACH\_MR\_EXIT\_NAME).

If a nonblank name is defined, the exit is invoked before performing a wait before retrying a failing message.

The format of the string is the same as for *SecurityExit*.

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

This parameter is valid only for *ChannelType* values of MQCHT\_RECEIVER, MQCHT\_REQUESTER, or MQCHT\_CLUSRCVR.

#### **MsgRetryInterval (MQCFIN)**

Message retry interval (parameter identifier: MQIACH\_MR\_INTERVAL).

Specifies the minimum time interval in milliseconds between retries of failing messages.

Specify a value in the range 0 - 999 999 999.

This parameter is valid only for *ChannelType* values of MQCHT\_RECEIVER, MQCHT\_REQUESTER, or MQCHT\_CLUSRCVR.

#### **MsgRetryUserData (MQCFST)**

Message retry exit user data (parameter identifier: MQCACH\_MR\_EXIT\_USER\_DATA).

Specifies user data that is passed to the message retry exit.

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

This parameter is valid only for *ChannelType* values of MQCHT\_RECEIVER, MQCHT\_REQUESTER, or MQCHT\_CLUSRCVR.

#### **MsgUserData (MQCFSL)**

Message exit user data (parameter identifier: MQCACH\_MSG\_EXIT\_USER\_DATA).

Specifies user data that is passed to the message exit.

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

For channels with a channel type (*ChannelType*) of MQCHT\_SVRCONN or MQCHT\_CLNTCONN, this parameter is accepted but ignored, since message exits are not invoked for such channels.

You can specify a list of exit user data strings by using an MQCFSL structure instead of an MQCFST structure.

- Each exit user data string is passed to the exit at the same ordinal position in the *MsgExit* list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.

- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all the exit user data in the list (excluding trailing blanks in each string) must not exceed MQ\_TOTAL\_EXIT\_DATA\_LENGTH. An individual string must not exceed MQ\_EXIT\_DATA\_LENGTH.
- On z/OS, you can specify up to eight strings.

### **NetworkPriority (MQCFIN)**

Network priority (parameter identifier: MQIACH\_NETWORK\_PRIORITY).

The priority for the network connection. If there are multiple paths available, distributed queuing selects the path with the highest priority.

The value must be in the range 0 (lowest) - 9 (highest).

This parameter applies only to channels with a *ChannelType* of MQCHT\_CLUSRCVR

### **NonPersistentMsgSpeed (MQCFIN)**

Speed at which nonpersistent messages are to be sent (parameter identifier: MQIACH\_NPM\_SPEED).

This parameter is supported in the following environments: IBM i, AIX, Linux, and Windows.

Specifying MQNPMS\_FAST means that nonpersistent messages on a channel need not wait for a syncpoint before being made available for retrieval. The advantage of this is that nonpersistent messages become available for retrieval far more quickly. The disadvantage is that because they do not wait for a syncpoint, they might be lost if there is a transmission failure.

This parameter is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_RECEIVER, MQCHT\_REQUESTER, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR. The value can be any of the following values:

#### **MQNPMS\_NORMAL**

Normal speed.

#### **MQNPMS\_FAST**

Fast speed.

### **Password (MQCFST)**

Password (parameter identifier: MQCACH\_PASSWORD).

This parameter is used by the message channel agent when attempting to initiate a secure SNA session with a remote message channel agent. On IBM i, AIX and Linux, it is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_REQUESTER, MQCHT\_CLNTCONN, or MQCHT\_CLUSSDR. On z/OS, it is valid only for a *ChannelType* value of MQCHT\_CLNTCONN.

The maximum length of the string is MQ\_PASSWORD\_LENGTH. However, only the first 10 characters are used.

### **Port (MQCFIN)**

Port number (parameter identifier MQIACH\_PORT).

The port number used to connect an AMQP channel. The default port for AMQP 1.0 connections is 5672. If you are already using port 5672, you can specify a different port.

This attribute is applicable to AMQP channels.

### **PropertyControl (MQCFIN)**

Property control attribute (parameter identifier MQIA\_PROPERTY\_CONTROL).

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor). The value can be any of the following values:

#### **MQPROP\_COMPATIBILITY**

If the message contains a property with a prefix of **mcd.**, **jms.**, **usr.** or **mqext.**, all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of

the message, except those properties contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

This value is the default value; it allows applications which expect JMS-related properties to be in an MQRFH2 header in the message data to continue to work unmodified.

#### **MQPROP\_NONE**

All properties of the message, except those properties in the message descriptor (or extension), are removed from the message before the message is sent to the remote queue manager.

#### **MQPROP\_ALL**

All properties of the message are included with the message when it is sent to the remote queue manager. The properties, except those properties in the message descriptor (or extension), are placed in one or more MQRFH2 headers in the message data.

This attribute is applicable to Sender, Server, Cluster Sender, and Cluster Receiver channels.

#### **PutAuthority (MQCFIN)**

Put authority (parameter identifier: MQIACH\_PUT\_AUTHORITY).


Specifies which user identifiers are used to establish authority to put messages to the destination queue (for message channels) or to execute an MQI call (for MQI channels).

This parameter is valid only for channels with a *ChannelType* value of MQCHT\_RECEIVER, MQCHT\_REQUESTER, MQCHT\_CLUSRCVR, or MQCHT\_SVRCONN.

The value can be any of the following values:


#### **MQPA\_DEFAULT**

Default user identifier is used.

 On z/OS, MQPA\_DEFAULT might involve using both the user ID received from the network and that derived from MCAUSER.

#### **MQPA\_CONTEXT**

The user ID from the *UserIdentifier* field of the message descriptor is used.

 On z/OS, MQPA\_CONTEXT might involve also using the user ID received from the network or that derived from MCAUSER, or both.

#### **MQPA\_ALTERNATE\_OR\_MCA**

The user ID from the *UserIdentifier* field of the message descriptor is used. Any user ID received from the network is not used. This value is supported only on z/OS.

#### **MQPA\_ONLY\_MCA**

The user ID derived from MCAUSER is used. Any user ID received from the network is not used. This value is supported only on z/OS.

#### **QMgrName (MQCFST)**

Queue manager name (parameter identifier: MQCA\_Q\_MGR\_NAME).

For channels with a *ChannelType* of MQCHT\_CLNTCONN, this name is the name of a queue manager to which a client application can request connection.

For channels of other types, this parameter is not valid. The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.



#### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:



Table 194. QSGDisposition: Where objects are defined and how they behave

QSGDisposition	Change	Copy, Create
<b>MQQSGD_COPY</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command. It uses the MQQSGD_GROUP object of the same name as the <i>ToChannelName</i> object (for Copy) or <i>ChannelName</i> object (for Create).
<b>MQQSGD_GROUP</b>	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE CHANNEL (channel -name) CHLTYPE (type) REPLACE QSGDISP (COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This definition is allowed only if the queue manager is in a queue sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to attempt to make or refresh local copies on page set zero:</p> <pre>DEFINE CHANNEL (channel -name) CHLTYPE (type) REPLACE QSGDISP (COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
<b>MQQSGD_PRIVATE</b>	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.
<b>MQQSGD_Q_MGR</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This value is the default value.	The object is defined on the page set of the queue manager that executes the command. This value is the default value.

**ReceiveExit (MQCFSL)**

Receive exit name (parameter identifier: MQCACH\_RCV\_EXIT\_NAME).

If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The format of the string is the same as for *SecurityExit*.

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

You can specify a list of exit names by using an MQCFSL structure instead of an MQCFST structure.

- The exits are invoked in the order specified in the list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all the exit names in the list (excluding trailing blanks in each name) must not exceed MQ\_TOTAL\_EXIT\_NAME\_LENGTH. An individual string must not exceed MQ\_EXIT\_NAME\_LENGTH.
- On z/OS, you can specify the names of up to eight exit programs.

### **ReceiveUserData (MQCFSL)**

Receive exit user data (parameter identifier: MQCACH\_RCV\_EXIT\_USER\_DATA).

Specifies user data that is passed to the receive exit.

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

You can specify a list of exit user data strings by using an MQCFSL structure instead of an MQCFST structure.

- Each exit user data string is passed to the exit at the same ordinal position in the *ReceiveExit* list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all the exit user data in the list (excluding trailing blanks in each string) must not exceed MQ\_TOTAL\_EXIT\_DATA\_LENGTH. An individual string must not exceed MQ\_EXIT\_DATA\_LENGTH.
- On z/OS, you can specify up to eight strings.

### **Replace (MQCFIN)**

Replace channel definition (parameter identifier: MQIACF\_REPLACE).

The value can be any of the following values:

#### **MQRP\_YES**

Replace existing definition.

If *ChannelType* is MQCHT\_CLUSSDR, MQRP\_YES can be specified only if the channel was created manually.

#### **MQRP\_NO**

Do not replace existing definition.

### **SecurityExit (MQCFST)**

Security exit name (parameter identifier: MQCACH\_SEC\_EXIT\_NAME).

If a nonblank name is defined, the security exit is invoked at the following times:

- Immediately after establishing a channel.  
Before any messages are transferred, the exit is enabled to instigate security flows to validate connection authorization.
- Upon receipt of a response to a security message flow.  
Any security message flows received from the remote processor on the remote machine are passed to the exit.

The exit is given the entire application message and message descriptor for modification.

The format of the string depends on the platform, as follows:

- On IBM i, AIX and Linux, it is of the form

```
libraryname(functionname)
```

**Note:** On IBM i systems, the following form is also supported for compatibility with older releases:

```
progrname libname
```

where *progrname* occupies the first 10 characters, and *libname* the second 10 characters (both blank-padded to the right if necessary).

- On Windows, it is of the form

```
dllname(functionname)
```

where *dllname* is specified without the suffix .DLL.

- On z/OS, it is a load module name, maximum length 8 characters (128 characters are allowed for exit names for client-connection channels, subject to a maximum total length of 999).

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

### **SecurityUserData (MQCFST)**

Security exit user data (parameter identifier: MQCACH\_SEC\_EXIT\_USER\_DATA).

Specifies user data that is passed to the security exit.

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

### **SendExit (MQCFSL)**

Send exit name (parameter identifier: MQCACH\_SEND\_EXIT\_NAME).

If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The format of the string is the same as for *SecurityExit*.

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

You can specify a list of exit names by using an MQCFSL structure instead of an MQCFST structure.

- The exits are invoked in the order specified in the list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all the exit names in the list (excluding trailing blanks in each name) must not exceed MQ\_TOTAL\_EXIT\_NAME\_LENGTH. An individual string must not exceed MQ\_EXIT\_NAME\_LENGTH.
- On z/OS, you can specify the names of up to eight exit programs.

### **SendUserData (MQCFSL)**

Send exit user data (parameter identifier: MQCACH\_SEND\_EXIT\_USER\_DATA).

Specifies user data that is passed to the send exit.

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

You can specify a list of exit user data strings by using an MQCFSL structure instead of an MQCFST structure.

- Each exit user data string is passed to the exit at the same ordinal position in the *SendExit* list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all the exit user data in the list (excluding trailing blanks in each string) must not exceed MQ\_TOTAL\_EXIT\_DATA\_LENGTH. An individual string must not exceed MQ\_EXIT\_DATA\_LENGTH.
- On z/OS, you can specify up to eight strings.

### **SeqNumberWrap (MQCFIN)**

Sequence wrap number (parameter identifier: MQIACH\_SEQUENCE\_NUMBER\_WRAP).

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

Specify a value in the range 100 - 999 999 999.

This parameter is not valid for channels with a *ChannelType* of MQCHT\_SVRCONN or MQCHT\_CLNTCONN.

### **SharingConversations (MQCFIN)**

Maximum number of sharing conversations (parameter identifier: MQIACH\_SHARING\_CONVERSATIONS).

Specifies the maximum number of conversations that can share a particular TCP/IP MQI channel instance (socket).

Specify a value in the range 0 - 999 999 999. The default value is 10 and the migrated value is 10.

This parameter is valid only for channels with a *ChannelType* of MQCHT\_CLNTCONN or MQCHT\_SVRCONN. It is ignored for channels with a *TransportType* other than MQXPT\_TCP.

The number of shared conversations does not contribute to the *MaxInstances* or *MaxInstancesPerClient* totals.

A value of:

**1**

Means that there is no sharing of conversations over a TCP/IP channel instance, but client heartbeating is available whether in an MQGET call or not, read ahead and client asynchronous consumption are available, and channel quiescing is more controllable.

**0**

Specifies no sharing of conversations over a TCP/IP channel instance. The channel instance runs in a mode before that of IBM WebSphere MQ 7.0, regarding:

- Administrator stop-quiesce
- Heartbeating
- Read ahead
- Client asynchronous consumption

### **ShortRetryCount (MQCFIN)**

Short retry count (parameter identifier: MQIACH\_SHORT\_RETRY).

The maximum number of attempts that are made by a sender or server channel to establish a connection to the remote machine, at intervals specified by *ShortRetryInterval* before the (normally longer) *LongRetryCount* and *LongRetryInterval* are used.

Retry attempts are made if the channel fails to connect initially (whether it is started automatically by the channel initiator or by an explicit command), and also if the connection fails after the channel

has successfully connected. However, if the cause of the failure is such that retry is unlikely to be successful, retries are not attempted.

Specify a value in the range 0 - 999 999 999.

This parameter is valid only for *ChannelType* values of MQCMT\_SENDER, MQCMT\_SERVER, MQCMT\_CLUSSDR, or MQCMT\_CLUSRCVR.

### **ShortRetryInterval (MQCFIN)**

Short timer (parameter identifier: MQIACH\_SHORT\_TIMER).

Specifies the short retry wait interval for a sender or server channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine.

The time is approximate. From IBM MQ 8.0, zero means that another connection attempt is made as soon as possible.

Specify a value in the range 0 - 999 999. Values exceeding this value are treated as 999 999.

This parameter is valid only for *ChannelType* values of MQCMT\_SENDER, MQCMT\_SERVER, MQCMT\_CLUSSDR, or MQCMT\_CLUSRCVR.

### **z/OS SPLProtection (MQCFIN)**

SPLProtection (parameter identifier: MQIACH\_SPL\_PROTECTION). This parameter applies to z/OS only, from IBM MQ 9.1.3 onwards.

Security policy protection parameter. Specifies what happens to messages across the channel when Advanced Message Security is active and an applicable policy exists.

This parameter is valid for channel types MQCMT\_SENDER, MQCMT\_SERVER, MQCMT\_RECEIVER, and MQCMT\_REQUESTER only.

Possible values are:

#### **MQSPL\_PASSTHRU**

Pass through, unchanged, any messages sent or received by the message channel agent for this channel.

This value is valid only for *ChannelType* values of MQCMT\_SENDER, MQCMT\_SERVER, MQCMT\_RECEIVER, or MQCMT\_REQUESTER, and is the default value.

#### **MQSPL\_REMOVE**

Remove any AMS protection from messages retrieved from the transmission queue by the message channel agent, and send the messages to the partner.

When the MCA gets a message from the transmission queue, if an AMS policy is defined for the transmission queue, it is applied to remove any AMS protection from the message prior to sending the message across the channel. If an AMS policy is not defined for the transmission queue, the message is sent as is.

This value is valid only for *ChannelType* values of MQCMT\_SENDER or MQCMT\_SERVER.

#### **MQSPL\_AS\_POLICY**

Based on the policy defined for the target queue, apply AMS protection to inbound messages prior to putting them on to the target queue.

When the message channel agent receives an inbound message, if an AMS policy is defined for the target queue, AMS protection is applied to the message prior to the message being put to the target queue. If an AMS policy is not defined for the target queue, the message is put to the target queue as is.

This value is valid only for *ChannelType* values of MQCMT\_RECEIVER or MQCMT\_REQUESTER.

### **SSLCipherSpec (MQCFST)**

CipherSpec (parameter identifier: MQCACH\_SSL\_CIPHER\_SPEC). Specifies the CipherSpec that is used on the channel. The length of the string is MQ\_SSL\_CIPHER\_SPEC\_LENGTH.



**Attention:** **IBM i** **z/OS** On IBM MQ for z/OS, you can also specify the four digit hexadecimal code of a CipherSpec, whether or not it appears in the following table. On IBM i, you can also specify the two digit hexadecimal code of a CipherSpec, whether or not it appears in the following table. Also, on IBM i, installation of AC3 is a prerequisite for the use of TLS. You should not specify hexadecimal cipher values in SSLCIPH, because it is unclear from the value which cipher will be used, and the choice of which protocol to be used is indeterminate. Using hexadecimal cipher values can lead to CipherSpec mismatch errors.

If a specific named CipherSpec is being used, the **SSLCIPH** values at the two ends of a channel must specify the same named CipherSpec.

This parameter is valid on all channel types that use transport type **TRPTYPE (TCP)**. If the parameter is blank, no attempt is made to use TLS on the channel. If the TRPTYPE is not TCP, the data is ignored and no error message is issued.

The value for this parameter is also used to set the value of SecurityProtocol, which is an output field on the [Inquire Channel Status \(Response\)](#) command.

**Note:** When SSLCipherSpec is used with a telemetry channel, it means TLS Cipher Suite.

**ALW** From IBM MQ 9.1.1, you can specify a value of ANY\_TLS12, which represents a subset of acceptable CipherSpecs that use the TLS 1.2 protocol; these CipherSpecs are listed in the following table.

**ALW** From IBM MQ 9.1.4, on AIX, Linux, and Windows, IBM MQ provides an expanded set of alias CipherSpecs that includes ANY\_TLS12\_OR\_HIGHER, and ANY\_TLS13\_OR\_HIGHER. These alias CipherSpecs are listed in the following table.



**Attention:** If your enterprise needs to guarantee that a certain CipherSpec is negotiated and used, you must not use an alias CipherSpec value such as ANY\_TLS12.

For information on changing your existing security configurations to use the ANY\_TLS12\_OR\_HIGHER CipherSpec, see [Migrating existing security configurations to use the ANY\\_TLS12\\_OR\\_HIGHER CipherSpec](#).

Table 195. CipherSpecs you can use with IBM MQ TLS support							
Platform support "1" on page 1040	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 1040	Suite B
<b>Alias CipherSpecs</b>							
All	ANY_TLS13_OR_HIGHER <a href="#">"3" on page 1040</a> <a href="#">"4" on page 1040</a>	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS13 <a href="#">"4" on page 1040</a> <a href="#">"5" on page 1040</a>	N/A	TLS 1.3	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS12_OR_HIGHER <a href="#">"4" on page 1040</a> <a href="#">"6" on page 1040</a>	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY_TLS12 <a href="#">"7" on page 1040</a>	N/A	TLS 1.2	Negotiated	Negotiated	Negotiated	Negotiated
All	ANY <a href="#">"8" on page 1040</a>	N/A	Negotiated	Negotiated	Negotiated	Negotiated	Negotiated
<b>CipherSpecs for TLS 1.3</b>							

Table 195. CipherSpecs you can use with IBM MQ TLS support (continued)













Platform support "1" on page 1040	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 1040	Suite B
All	TLS_AES_128_GCM_SHA256	1301	TLS 1.3	GCM	AES-128 with GCM (128)	Yes	No
All	TLS_AES_256_GCM_SHA384	1302	TLS 1.3	GCM	AES-256 with GCM (256)	Yes	No
All	TLS_CHACHA20_POLY1305_SHA256	1303	TLS 1.3	POLY1305	CHACHA20 (256)	No	No
 ALW	TLS_AES_128_CCM_SHA256	1304	TLS 1.3	CBC-MAC	AES-128 with CTR (128)	Yes	No
 ALW	TLS_AES_128_CCM_8_SHA256 <a href="#">"10" on page 1040</a>	1305	TLS 1.3	CBC-MAC	AES-128 with CTR (128)	Yes	No
<b>CipherSpecs for TLS 1.2</b>							
All	TLS_RSA_WITH_AES_128_CBC_SHA256 <a href="#">"9" on page 1040</a>	003C	TLS 1.2	SHA-256	AES (128)	Yes	No
All	TLS_RSA_WITH_AES_256_CBC_SHA256 <a href="#">"9" on page 1040</a> <a href="#">"11" on page 1040</a>	003D	TLS 1.2	SHA-256	AES (256)	Yes	No
All	TLS_RSA_WITH_AES_128_GCM_SHA256 <a href="#">"9" on page 1040</a> <a href="#">"12" on page 1040</a>	009C	TLS 1.2	SHA-256 and AEAD GCM	AES (128)	Yes	No
All	TLS_RSA_WITH_AES_256_GCM_SHA384 <a href="#">"9" on page 1040</a> <a href="#">"11" on page 1040</a> <a href="#">"12" on page 1040</a>	009D	TLS 1.2	SHA-384 and AEAD GCM	AES (256)	Yes	No
All	ECDHE_ECDSA_AES_128_CBC_SHA256 <a href="#">"9" on page 1040</a>	C023	TLS 1.2	SHA-256	AES (128)	Yes	No
All	ECDHE_ECDSA_AES_256_CBC_SHA384 <a href="#">"9" on page 1040</a> <a href="#">"11" on page 1040</a>	C024	TLS 1.2	SHA-384	AES (256)	Yes	No
All	ECDHE_RSA_AES_128_CBC_SHA256 <a href="#">"9" on page 1040</a>	C027	TLS 1.2	SHA-256	AES (128)	Yes	No
All	ECDHE_RSA_AES_256_CBC_SHA384 <a href="#">"9" on page 1040</a> <a href="#">"11" on page 1040</a>	C028	TLS 1.2	SHA-384	AES (256)	Yes	No
 Multi	ECDHE_ECDSA_AES_128_GCM_SHA256 <a href="#">"11" on page 1040</a> <a href="#">"12" on page 1040</a>	C02B	TLS 1.2	SHA-256 and AEAD GCM	AES (SHA384)	Yes	128 bit
 Multi	ECDHE_ECDSA_AES_256_GCM_SHA384 <a href="#">"11" on page 1040</a> <a href="#">"12" on page 1040</a>	C02C	TLS 1.2	SHA-384 and AEAD GCM	AES (SHA384)	Yes	192 bit

Table 195. CipherSpecs you can use with IBM MQ TLS support (continued)

Platform support "1" on page 1040	CipherSpec name	Hex code	Protocol used	MAC algorithm	Encryption algorithm (encryption bits)	FIPS "2" on page 1040	Suite B
All	ECDHE_RSA_AES_128_GCM_SHA256 "12" on page 1040	C02F	TLS 1.2	SHA-256 and AEAD GCM	AES (128)	Yes	No
All	ECDHE_RSA_AES_256_GCM_SHA384 "11" on page 1040 "12" on page 1040	C030	TLS 1.2	AEAD AES-128 GCM	AES (SHA384)	Yes	No

**Notes:**

- For a list of platforms covered by each platform icon, see [Icons used in the product documentation](#).
- Specifies whether the CipherSpec is FIPS-certified on a FIPS-certified platform. See [Federal Information Processing Standards \(FIPS\)](#) for an explanation of FIPS.
-  The ANY\_TLS13\_OR\_HIGHER alias CipherSpec negotiates the highest level of security that the remote end will allow but will only connect using a TLS 1.3 or higher protocol.
-  To use TLS 1.3, or the ANY CipherSpec, on IBM i the underlying operating system version must support TLS 1.3. See [System TLS support for TLSv1.3](#) for more information.
-  The ANY\_TLS13 alias CipherSpec represents a subset of acceptable CipherSpecs that use the TLS 1.3 protocol, as listed in this table for each platform.
-  The ANY\_TLS12\_OR\_HIGHER alias CipherSpec negotiates the highest level of security that the remote end will allow but will only connect using a TLS 1.2 or higher protocol.
- The ANY\_TLS12 CipherSpec represents a subset of acceptable CipherSpecs that use the TLS 1.2 protocol, as listed in this table for each platform.
-  The ANY alias CipherSpec negotiates the highest level of security that the remote end will allow.
-  These CipherSpecs are not enabled on IBM i 7.4 systems that have System Value QSSLCSLCTL set to \*OPSSYS.
-  These CipherSpecs use an 8-octet Integrity Check Value (ICV) instead of a 16-octet ICV.
- This CipherSpec cannot be used to secure a connection from the IBM MQ Explorer to a queue manager unless the appropriate unrestricted policy files are applied to the JRE used by the Explorer.
-  Following a recommendation by GSKit, TLS 1.2 GCM CipherSpecs have a restriction which means that after 2<sup>24.5</sup> TLS records are sent, using the same session key, the connection is terminated with message [AMQ9288E](#). This GCM restriction is active, regardless of the FIPS mode being used.

To prevent this error from happening, avoid using TLS 1.2 GCM Ciphers, enable secret key reset, or start your IBM MQ queue manager or client with the environment variable GSK\_ENFORCE\_GCM\_RESTRICTION=GSK\_FALSE set. For GSKit libraries, you must set this environment variable on both sides of the connection, and apply it to both client to queue manager connections and queue manager to queue manager connections. Note that this setting affects unmanaged .NET clients, but not Java or managed .NET clients. For more information, see [AES-GCM cipher restriction](#).

This restriction does not apply to IBM MQ for z/OS.

For more information about CipherSpecs, see [Enabling CipherSpecs](#).



When you request a personal certificate, you specify a key size for the public and private key pair. The key size that is used during the SSL handshake can depend on the size stored in the certificate and on the CipherSpec:

- **ALW** **z/OS** On z/OS, AIX, Linux, and Windows, when a CipherSpec name includes `_EXPORT`, the maximum handshake key size is 512 bits. If either of the certificates exchanged during the SSL handshake has a key size greater than 512 bits, a temporary 512-bit key is generated for use during the handshake.
- **z/OS** For z/OS, System SSL state that if a TLS V1.3 connection is being negotiated:
  - The minimum key size for an RSA peer certificate is the larger of the following two values: 2048, or the value specified in the `GSK_PEER_RSA_MIN_KEY_SIZE` attribute.
  - The minimum key size for an ECC peer certificate is the larger of the following two values: 256, or the value specified in the `GSK_PEER_ECC_MIN_KEY_SIZE` attribute.
- **ALW** On AIX, Linux, and Windows, when a CipherSpec name includes `_EXPORT1024`, the handshake key size is 1024 bits.
- Otherwise the handshake key size is the size stored in the certificate.

### SSLClientAuth (MQCFIN)

Client authentication (parameter identifier: MQIACH\_SSL\_CLIENT\_AUTH).

The value can be any of the following values:

#### MQSCA\_REQUIRED

Client authentication required.

#### MQSCA\_OPTIONAL

Client authentication optional.

Defines whether IBM MQ requires a certificate from the TLS client.

The TLS client is the end of the message channel that initiates the connection. The TLS Server is the end of the message channel that receives the initiation flow.

The parameter is used only for channels with SSLCIPH specified. If SSLCIPH is blank, the data is ignored and no error message is issued.

### SSLPeerName (MQCFST)

Peer name (parameter identifier: MQCACH\_SSL\_PEER\_NAME).

**Note:** An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both SSLPEER on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns in order to connect. For more information, see [Channel authentication records](#).

**Multi** On Multiplatforms, the length of the string is `MQ_SSL_PEER_NAME_LENGTH`.

**z/OS** On z/OS, the length of the string is `MQ_SSL_SHORT_PEER_NAME_LENGTH`.

Specifies the filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel. (A Distinguished Name is the identifier of the TLS certificate.) If the Distinguished Name in the certificate received from the peer does not match the SSLPEER filter, the channel does not start.

This parameter is optional; if it is not specified, the Distinguished Name of the peer is not checked when the channel is started. (The Distinguished Name from the certificate is still written into the SSLPEER definition held in memory, and passed to the security exit). If SSLCIPH is blank, the data is ignored and no error message is issued.


This parameter is valid for all channel types.

The SSLPEER value is specified in the standard form used to specify a Distinguished Name. For example:  
 SSLPEER( ' SERIALNUMBER=4C:D0:49:D5:02:5F:38,CN="H1\_C\_FR1",O=IBM,C=GB' )

You can use a semi-colon as a separator instead of a comma.

The possible attribute types supported are as follows:

*Table 196. Attribute types and descriptions*

<b>Attribute</b>	<b>Description</b>
SERIALNUMBER	Certificate serial number
MAIL	Email address
 E	Email address (Deprecated in preference to MAIL)
UID or USERID	User identifier
CN	Common Name
T	Title
OU	Organizational Unit name
DC	Domain component
O	Organization name
STREET	Street / First line of address
L	Locality name
ST (or SP or S)	State or Province name
PC	Postal code / zip code
C	Country
UNSTRUCTUREDNAME	Host name
UNSTRUCTUREDADDRESS	IP address
DNQ	Distinguished name qualifier

IBM MQ only accepts uppercase letters for the attribute types.

If any of the unsupported attribute types are specified in the SSLPEER string, an error is output either when the attribute is defined or at run time (depending on which platform you are running on), and the string is deemed not to have matched the Distinguished Name of the flowed certificate.

If the Distinguished Name of the flowed certificate contains multiple OU (organizational unit) attributes, and SSLPEER specifies these attributes to be compared, they must be defined in descending hierarchical order. For example, if the Distinguished Name of the flowed certificate contains the OUs OU=Large Unit,OU=Medium Unit,OU=Small Unit, specifying the following SSLPEER values work:

```
('OU=Large Unit,OU=Medium Unit') ('OU=*,OU=Medium Unit,OU=Small Unit') ('OU=*,OU=Medium Unit')
```

but specifying the following SSLPEER values fail:

```
('OU=Medium Unit,OU=Small Unit') ('OU=Large Unit,OU=Small Unit') ('OU=Medium Unit')
```

Any or all the attribute values can be generic, either an asterisk (\*) on its own, or a stem with initiating or trailing asterisks. This value allows the SSLPEER to match any Distinguished Name value, or any value starting with the stem for that attribute.

If an asterisk is specified at the beginning or end of any attribute value in the Distinguished Name on the certificate, you can specify `\*` to check for an exact match in SSLPEER. For example, if you have an attribute of `CN=Test*` in the Distinguished Name of the certificate, you can use the following command:

```
SSLPEER('CN=Test\*')
```

#### **Multi** **V 9.3.0** **TemporaryModelQName (MQCFST)**

The name of the model queue to be used while creating a temporary queue (parameter identifier MQCACH\_TEMPORARY\_MODEL\_Q).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

#### **Multi** **V 9.3.0** **TemporaryQPrefix (MQCFST)**

The temporary queue name prefix to add to the beginning of the model queue, when deriving a temporary queue name (parameter identifier MQCACH\_TEMPORARY\_Q\_PREFIX).

The maximum length of the string is MQ\_TEMPORARY\_Q\_PREFIX\_LENGTH.

#### **TpName (MQCFST)**

Transaction program name (parameter identifier: MQCACH\_TP\_NAME).

This name is the LU 6.2 transaction program name.

The maximum length of the string is MQ\_TP\_NAME\_LENGTH.

- On IBM i, AIX, Linux, and Windows platforms, this parameter can be set only to blanks. The actual name is taken instead from the CPI-C Communications Side Object or (on Windows ) from the CPI-C symbolic destination name properties.

This parameter is valid only for channels with a *TransportType* of MQXPT\_LU62. It is not valid for receiver channels.

#### **TPRoot (MQCFST)**

Topic root for an AMQP channel. (parameter identifier: MQCACH\_TOPIC\_ROOT).

The default value for TPRoot is SYSTEM.BASE.TOPIC. With this value, the topic string an AMQP client uses to publish or subscribe has no prefix, and the client can exchange messages with other MQ pub/sub applications. To have AMQP clients publish and subscribe under a topic prefix, first create an MQ topic object with a topic string set to the prefix you want, then set TPRoot to the name of the MQ topic object you created.

This parameter is valid only for AMQP channels.

#### **TransportType (MQCFIN)**

Transmission protocol type (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

No check is made that the correct transport type has been specified if the channel is initiated from the other end. The value can be any of the following values:

##### **MQXPT\_LU62**

LU 6.2.

##### **MQXPT\_TCP**

TCP.

##### **MQXPT\_NETBIOS**

NetBIOS.

This value is supported in Windows. It also applies to z/OS for defining client-connection channels that connect to servers on the platforms supporting NetBIOS.

##### **MQXPT\_SPX**

SPX.

This value is supported in Windows. It also applies to z/OS for defining client-connection channels that connect to servers on the platforms supporting SPX.

**UseClId (MQCFIN)**

Determines how authorization checks are done for AMQP channels. (parameter identifier: MQIACH\_USE\_CLIENT\_ID).

The value can be any of the following values:

**MQUCI\_NO**

The MCA user ID should be used for authorization checks.

**MQUCI\_YES**

The client ID should be used for authorization checks.

This parameter is valid only for AMQP channels.

**UseDLQ (MQCFIN)**

Determines whether the dead-letter queue is used when messages cannot be delivered by channels. (parameter identifier: MQIA\_USE\_DEAD\_LETTER\_Q).

The value can be any of the following values:

**MQUSEDLQ\_NO**

Messages that cannot be delivered by a channel are treated as a failure. The channel either discards the message, or the channel ends, in accordance with the NonPersistentMsgSpeed setting.

**MQUSEDLQ\_YES**

When the DEADQ queue manager attribute provides the name of a dead-letter queue, then it is used, else the behavior is as for MQUSEDLQ\_NO.

**UserIdentifier (MQCFST)**

Task user identifier (parameter identifier: MQCACH\_USER\_ID).

This parameter is used by the message channel agent when attempting to initiate a secure SNA session with a remote message channel agent. On IBM i, AIX and Linux, it is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_REQUESTER, MQCHT\_CLNTCONN, MQCHT\_CLUSSDR, or MQCHT\_CLUSRCVR. On z/OS, it is valid only for a *ChannelType* value of MQCHT\_CLNTCONN.

The maximum length of the string is MQ\_USER\_ID\_LENGTH. However, only the first 10 characters are used.

**XmitQName (MQCFST)**

Transmission queue name (parameter identifier: MQCACH\_XMIT\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

A transmission queue name is required (either previously defined or specified here) if *ChannelType* is MQCHT\_SENDER or MQCHT\_SERVER. It is not valid for other channel types.

**Error codes (Change, Copy, and Create Channel)**

This command might return the following error codes in the response format header, in addition to those codes listed in [“Error codes applicable to all commands”](#) on page 990.

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_BATCH\_INT\_ERROR**

Batch interval not valid.

**MQRCCF\_BATCH\_INT\_WRONG\_TYPE**

Batch interval parameter not allowed for this channel type.

**MQRCCF\_BATCH\_SIZE\_ERROR**

Batch size not valid.

**MQRCCF\_CHANNEL\_NAME\_ERROR**

Channel name error.

**MQRCCF\_CHANNEL\_NOT\_FOUND**  
Channel not found.

**MQRCCF\_CHANNEL\_TYPE\_ERROR**  
Channel type not valid.

**MQRCCF\_CLUSTER\_NAME\_CONFLICT**  
Cluster name conflict.

**MQRCCF\_DISC\_INT\_ERROR**  
Disconnection interval not valid.

**MQRCCF\_DISC\_INT\_WRONG\_TYPE**  
Disconnection interval not allowed for this channel type.

**MQRCCF\_HB\_INTERVAL\_ERROR**  
Heartbeat interval not valid.

**MQRCCF\_HB\_INTERVAL\_WRONG\_TYPE**  
Heartbeat interval parameter not allowed for this channel type.

**MQRCCF\_KWD\_VALUE\_WRONG\_TYPE**  
An attribute keyword and value combination are not valid for this channel type.

**MQRCCF\_LONG\_RETRY\_ERROR**  
Long retry count not valid.

**MQRCCF\_LONG\_RETRY\_WRONG\_TYPE**  
Long retry parameter not allowed for this channel type.

**MQRCCF\_LONG\_TIMER\_ERROR**  
Long timer not valid.

**MQRCCF\_LONG\_TIMER\_WRONG\_TYPE**  
Long timer parameter not allowed for this channel type.

**MQRCCF\_MAX\_INSTANCES\_ERROR**  
Maximum instances value not valid.

**MQRCCF\_MAX\_INSTS\_PER\_CLNT\_ERR**  
Maximum instances per client value not valid.

**MQRCCF\_MAX\_MSG\_LENGTH\_ERROR**  
Maximum message length not valid.

**MQRCCF\_MCA\_NAME\_ERROR**  
Message channel agent name error.

**MQRCCF\_MCA\_NAME\_WRONG\_TYPE**  
Message channel agent name not allowed for this channel type.

**MQRCCF\_MCA\_TYPE\_ERROR**  
Message channel agent type not valid.

**MQRCCF\_MISSING\_CONN\_NAME**  
Connection name parameter required but missing.

**MQRCCF\_MR\_COUNT\_ERROR**  
Message retry count not valid.

**MQRCCF\_MR\_COUNT\_WRONG\_TYPE**  
Message-retry count parameter not allowed for this channel type.

**MQRCCF\_MR\_EXIT\_NAME\_ERROR**  
Channel message-retry exit name error.

**MQRCCF\_MR\_EXIT\_NAME\_WRONG\_TYPE**  
Message-retry exit parameter not allowed for this channel type.

**MQRCCF\_MR\_INTERVAL\_ERROR**  
Message retry interval not valid.

**MQRCCF\_MR\_INTERVAL\_WRONG\_TYPE**  
Message-retry interval parameter not allowed for this channel type.

**MQRCCF\_MSG\_EXIT\_NAME\_ERROR**  
Channel message exit name error.

**MQRCCF\_NET\_PRIORITY\_ERROR**  
Network priority value error.

**MQRCCF\_NET\_PRIORITY\_WRONG\_TYPE**  
Network priority attribute not allowed for this channel type.

**MQRCCF\_NPM\_SPEED\_ERROR**  
Nonpersistent message speed not valid.

**MQRCCF\_NPM\_SPEED\_WRONG\_TYPE**  
Nonpersistent message speed parameter not allowed for this channel type.

**MQRCCF\_PARM\_SEQUENCE\_ERROR**  
Parameter sequence not valid.

**MQRCCF\_PUT\_AUTH\_ERROR**  
Put authority value not valid.

**MQRCCF\_PUT\_AUTH\_WRONG\_TYPE**  
Put authority parameter not allowed for this channel type.

**MQRCCF\_RCV\_EXIT\_NAME\_ERROR**  
Channel receive exit name error.

**MQRCCF\_SEC\_EXIT\_NAME\_ERROR**  
Channel security exit name error.

**MQRCCF\_SEND\_EXIT\_NAME\_ERROR**  
Channel send exit name error.

**MQRCCF\_SEQ\_NUMBER\_WRAP\_ERROR**  
Sequence wrap number not valid.

**MQRCCF\_SHARING\_CONVS\_ERROR**  
Value given for Sharing Conversations not valid.

**MQRCCF\_SHARING\_CONVS\_TYPE**  
Sharing Conversations parameter not valid for this channel type.

**MQRCCF\_SHORT\_RETRY\_ERROR**  
Short retry count not valid.

**MQRCCF\_SHORT\_RETRY\_WRONG\_TYPE**  
Short retry parameter not allowed for this channel type.

**MQRCCF\_SHORT\_TIMER\_ERROR**  
Short timer value not valid.

**MQRCCF\_SHORT\_TIMER\_WRONG\_TYPE**  
Short timer parameter not allowed for this channel type.

**MQRCCF\_SSL\_CIPHER\_SPEC\_ERROR**  
TLS CipherSpec not valid.

**MQRCCF\_SSL\_CLIENT\_AUTH\_ERROR**  
TLS client authentication not valid.

**MQRCCF\_SSL\_PEER\_NAME\_ERROR**  
TLS peer name not valid.

**MQRCCF\_WRONG\_CHANNEL\_TYPE**  
Parameter not allowed for this channel type.

**MQRCCF\_XMIT\_PROTOCOL\_TYPE\_ERR**  
Transmission protocol type not valid.

**MQRCCF\_XMIT\_Q\_NAME\_ERROR**  
Transmission queue name error.

**MQRCCF\_XMIT\_Q\_NAME\_WRONG\_TYPE**  
Transmission queue name not allowed for this channel type.

## Change, Copy, and Create Channel (MQTT) on AIX, Linux, and Windows

The Change Channel PCF command changes existing Telemetry channel definitions. The Copy and Create Channel commands create new Telemetry channel definitions - the Copy command uses attribute values of an existing channel definition.

The Change Channel (MQCMD\_CHANGE\_CHANNEL) command changes the specified attributes in a channel definition. For any optional parameters that are omitted, the value does not change.

The Copy Channel (MQCMD\_COPY\_CHANNEL) command creates new channel definition using, for attributes not specified in the command, the attribute values of an existing channel definition.

The Create Channel (MQCMD\_CREATE\_CHANNEL) command creates an IBM MQ channel definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager. If a system default channel exists for the type of channel being created, the default values are taken from there.

### Required parameters (Change, Create Channel)

#### ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

Specifies the name of the channel definition to be changed, or created

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

#### ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

Specifies the type of the channel being changed, copied, or created. The value can be any of the following values:

##### **MQCHT\_MQTT**

Telemetry.

#### TrpType (MQCFIN)

Transmission protocol type of the channel (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

This parameter is required for a create command in telemetry.

No check is made that the correct transport type has been specified if the channel is initiated from the other end. The value is:

##### **MQXPT\_TCP**

TCP.

#### Port (MQCFIN)

The port number to use if *TrpType* is set to MQXPT\_TCP. This parameter is required for a create command in telemetry, if *TrpType* is set to MQXPT\_TCP.

The value is in the range 1 - 65335.

### Required parameters (Copy Channel)

#### ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

Specifies the type of the channel being changed, copied, or created. The value can be any of the following values:

##### **MQCHT\_MQTT**

Telemetry.

## Optional parameters (Change, Copy, and Create Channel)

### Backlog (MQCFIN)

The number of concurrent connection requests that the telemetry channel supports at any one time (parameter identifier: MQIACH\_BACKLOG).

The value is in the range 0 - 999999999.

### JAASConfig (MQCFST)

The file path of the JAAS configuration (parameter identifier: MQCACH\_JAAS\_CONFIG).

The maximum length of this value is MQ\_JAAS\_CONFIG\_LENGTH.

Only one of JAASCONFIG, MCAUSER, and USECLIENTID can be specified for a telemetry channel; if none is specified, no authentication is performed. If JAASConfig is specified, the client flows a user name and password. In all other cases, the flowed user name is ignored.

### LocalAddress (MQCFST)

Local communications address for the channel (parameter identifier: MQCACH\_LOCAL\_ADDRESS).

The maximum length of the string is MQ\_LOCAL\_ADDRESS\_LENGTH.

The value that you specify depends on the transport type (*TrpType*) to be used:

### TCP/IP

The value is the optional IP address and optional port or port range to be used for outbound TCP/IP communications. The format for this information is as follows:

```
[ip-addr][(low-port[,high-port])]
```

where *ip-addr* is specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric form, and *low-port* and *high-port* are port numbers enclosed in parentheses. All are optional.

### All Others

The value is ignored; no error is diagnosed.

Use this parameter if you want a channel to use a particular IP address, port, or port range for outbound communications. This parameter is useful when a machine is connected to multiple networks with different IP addresses.

Examples of use

Value	Meaning
9.20.4.98	Channel binds to this address locally
9.20.4.98 (1000)	Channel binds to this address and port 1000 locally
9.20.4.98 (1000,2000)	Channel binds to this address and uses a port in the range 1000 - 2000 locally
(1000)	Channel binds to port 1000 locally
(1000,2000)	Channel binds to a port in the range 1000 - 2000 locally

### Note:

- Do not confuse this parameter with *ConnectionName*. The *LocalAddress* parameter specifies the characteristics of the local communications; the *ConnectionName* parameter specifies how to reach a remote queue manager.

### Protocol (MQCFIL)

Client protocols supported by the MQTT channel (parameter identifier: MQIACH\_PROTOCOL).

The value can be one or more of the following values:



**MQPROTO\_MQTTV311**

The channel accepts connections from clients using the protocol defined by the [MQTT 3.1.1 Oasis standard](#). The functionality provided by this protocol is almost identical to that provided by the pre-existing MQTTV3 protocol.

**MQPROTO\_MQTTV3**

The channel accepts connections from clients using the [MQTT V3.1 Protocol Specification](#) from [mqtt.org](#).

**MQPROTO\_HTTP**

The channel accepts HTTP requests for pages, or WebSockets connections to MQ Telemetry.

If you specify no client protocols, the channel accepts connections from clients using any of the supported protocols.

If you are using IBM MQ 8.0.0 Fix Pack 3 or later, and your configuration includes an MQTT channel that was last modified in an earlier version of the product, you must explicitly change the protocol setting to prompt the channel to use the MQTTV311 option. This is so even if the channel does not specify any client protocols, because the specific protocols to use with the channel are stored at the time the channel is configured, and previous versions of the product have no awareness of the MQTTV311 option. To prompt a channel in this state to use the MQTTV311 option, explicitly add the option then save your changes. The channel definition is now aware of the option. If you subsequently change the settings again, and specify no client protocols, the MQTTV311 option is still included in the stored list of supported protocols.

**SSLCipherSuite (MQCFST)**

CipherSuite (parameter identifier: MQCACH\_SSL\_CIPHER\_SUITE).

The length of the string is MQ\_SSL\_CIPHER\_SUITE\_LENGTH.

SSL CIPHER SUITE character channel parameter type.

**SSLClientAuth (MQCFIN)**

Client authentication (parameter identifier: MQIACH\_SSL\_CLIENT\_AUTH).

The value can be any of the following values:

**MQSCA\_REQUIRED**

Client authentication required

**MQSCA\_OPTIONAL**

Client authentication is optional.

**MQSCA\_NEVER\_REQUIRED**

Client authentication is never required, and must not be provided.

Defines whether IBM MQ requires a certificate from the TLS client.

The TLS client is the end of the message channel that initiates the connection. The TLS Server is the end of the message channel that receives the initiation flow.

The parameter is used only for channels with SSLCIPH specified. If SSLCIPH is blank, the data is ignored and no error message is issued.

**SSLKeyFile (MQCFST)**

The store for digital certificates and their associated private keys (parameter identifier: MQCA\_SSL\_KEY\_REPOSITORY).

If you do not specify a key file, TLS is not used.

The maximum length of this parameter is MQ\_SSL\_KEY\_REPOSITORY\_LENGTH.

**SSLPassPhrase (MQCFST)**

The password for the key repository (parameter identifier: MQCACH\_SSL\_KEY\_PASSPHRASE).

If no pass phrase is entered, then unencrypted connections must be used.

The maximum length of this parameter is MQ\_SSL\_KEY\_PASSPHRASE\_LENGTH.

**V 9.3.0** If the MQXR service is configured for encryption of passphrases by specifying the **-sf** option in STARTARG for the service, then the pass phrase will be encrypted. For more information on encryption of passphrases see [Encryption of passphrases for MQTT TLS channels](#).

### **UseClientIdentifier (MQCFIN)**

Determines whether to use the client ID of a new connection as the user ID for that connection (parameter identifier: MQIACH\_USE\_CLIENT\_ID).

The value is either:

#### **MQUCI\_YES**

Yes.

#### **MQUCI\_NO**

No.

Only one of JAASCONFIG, MCAUSER, and USECLIENTID can be specified for a telemetry channel; if none is specified, no authentication is performed. If USECLIENTID is specified, the flowed user name of the client is ignored.

### **Error codes (Change, Copy, and Create Channel)**

This command might return the following error codes in the response format header, in addition to those codes listed in [“Error codes applicable to all commands”](#) on page 990.

#### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRCCF\_BATCH\_INT\_ERROR**

Batch interval not valid.

#### **MQRCCF\_BATCH\_INT\_WRONG\_TYPE**

Batch interval parameter not allowed for this channel type.

#### **MQRCCF\_BATCH\_SIZE\_ERROR**

Batch size not valid.

#### **MQRCCF\_CHANNEL\_NAME\_ERROR**

Channel name error.

#### **MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

#### **MQRCCF\_CHANNEL\_TYPE\_ERROR**

Channel type not valid.

#### **MQRCCF\_CLUSTER\_NAME\_CONFLICT**

Cluster name conflict.

#### **MQRCCF\_DISC\_INT\_ERROR**

Disconnection interval not valid.

#### **MQRCCF\_DISC\_INT\_WRONG\_TYPE**

Disconnection interval not allowed for this channel type.

#### **MQRCCF\_HB\_INTERVAL\_ERROR**

Heartbeat interval not valid.

#### **MQRCCF\_HB\_INTERVAL\_WRONG\_TYPE**

Heartbeat interval parameter not allowed for this channel type.

#### **MQRCCF\_LONG\_RETRY\_ERROR**

Long retry count not valid.

#### **MQRCCF\_LONG\_RETRY\_WRONG\_TYPE**

Long retry parameter not allowed for this channel type.

#### **MQRCCF\_LONG\_TIMER\_ERROR**

Long timer not valid.

**MQRCCF\_LONG\_TIMER\_WRONG\_TYPE**  
Long timer parameter not allowed for this channel type.

**MQRCCF\_MAX\_INSTANCES\_ERROR**  
Maximum instances value not valid.

**MQRCCF\_MAX\_INSTS\_PER\_CLNT\_ERR**  
Maximum instances per client value not valid.

**MQRCCF\_MAX\_MSG\_LENGTH\_ERROR**  
Maximum message length not valid.

**MQRCCF\_MCA\_NAME\_ERROR**  
Message channel agent name error.

**MQRCCF\_MCA\_NAME\_WRONG\_TYPE**  
Message channel agent name not allowed for this channel type.

**MQRCCF\_MCA\_TYPE\_ERROR**  
Message channel agent type not valid.

**MQRCCF\_MISSING\_CONN\_NAME**  
Connection name parameter required but missing.

**MQRCCF\_MR\_COUNT\_ERROR**  
Message retry count not valid.

**MQRCCF\_MR\_COUNT\_WRONG\_TYPE**  
Message-retry count parameter not allowed for this channel type.

**MQRCCF\_MR\_EXIT\_NAME\_ERROR**  
Channel message-retry exit name error.

**MQRCCF\_MR\_EXIT\_NAME\_WRONG\_TYPE**  
Message-retry exit parameter not allowed for this channel type.

**MQRCCF\_MR\_INTERVAL\_ERROR**  
Message retry interval not valid.

**MQRCCF\_MR\_INTERVAL\_WRONG\_TYPE**  
Message-retry interval parameter not allowed for this channel type.

**MQRCCF\_MSG\_EXIT\_NAME\_ERROR**  
Channel message exit name error.

**MQRCCF\_NET\_PRIORITY\_ERROR**  
Network priority value error.

**MQRCCF\_NET\_PRIORITY\_WRONG\_TYPE**  
Network priority attribute not allowed for this channel type.

**MQRCCF\_NPM\_SPEED\_ERROR**  
Nonpersistent message speed not valid.

**MQRCCF\_NPM\_SPEED\_WRONG\_TYPE**  
Nonpersistent message speed parameter not allowed for this channel type.

**MQRCCF\_PARM\_SEQUENCE\_ERROR**  
Parameter sequence not valid.

**MQRCCF\_PUT\_AUTH\_ERROR**  
Put authority value not valid.

**MQRCCF\_PUT\_AUTH\_WRONG\_TYPE**  
Put authority parameter not allowed for this channel type.

**MQRCCF\_RCV\_EXIT\_NAME\_ERROR**  
Channel receive exit name error.

**MQRCCF\_SEC\_EXIT\_NAME\_ERROR**  
Channel security exit name error.

**MQRCCF\_SEND\_EXIT\_NAME\_ERROR**  
Channel send exit name error.

**MQRCCF\_SEQ\_NUMBER\_WRAP\_ERROR**

Sequence wrap number not valid.

**MQRCCF\_SHARING\_CONVS\_ERROR**

Value given for Sharing Conversations not valid.

**MQRCCF\_SHARING\_CONVS\_TYPE**

Sharing Conversations parameter not valid for this channel type.

**MQRCCF\_SHORT\_RETRY\_ERROR**

Short retry count not valid.

**MQRCCF\_SHORT\_RETRY\_WRONG\_TYPE**

Short retry parameter not allowed for this channel type.

**MQRCCF\_SHORT\_TIMER\_ERROR**

Short timer value not valid.

**MQRCCF\_SHORT\_TIMER\_WRONG\_TYPE**

Short timer parameter not allowed for this channel type.

**MQRCCF\_SSL\_CIPHER\_SPEC\_ERROR**

TLS CipherSpec not valid.

**MQRCCF\_SSL\_CLIENT\_AUTH\_ERROR**

TLS client authentication not valid.

**MQRCCF\_SSL\_PEER\_NAME\_ERROR**

TLS peer name not valid.

**MQRCCF\_WRONG\_CHANNEL\_TYPE**

Parameter not allowed for this channel type.

**MQRCCF\_XMIT\_PROTOCOL\_TYPE\_ERR**

Transmission protocol type not valid.

**MQRCCF\_XMIT\_Q\_NAME\_ERROR**

Transmission queue name error.

**MQRCCF\_XMIT\_Q\_NAME\_WRONG\_TYPE**

Transmission queue name not allowed for this channel type.

**Multi *Change, Copy, and Create Channel Listener on Multiplatforms***

The Change Channel Listener PCF command changes existing channel listener definitions. The Copy and Create Channel Listener commands create new channel listener definitions - the Copy command uses attribute values of an existing channel listener definition.

The Change Channel Listener (MQCMD\_CHANGE\_LISTENER) command changes the specified attributes of an existing IBM MQ listener definition. For any optional parameters that are omitted, the value does not change.

The Copy Channel Listener (MQCMD\_COPY\_LISTENER) command creates an IBM MQ listener definition, using, for attributes not specified in the command, the attribute values of an existing listener definition.

The Create Channel Listener (MQCMD\_CREATE\_LISTENER) command creates an IBM MQ listener definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

**Required parameters (Change and Create Channel Listener)****ListenerName (MQCFST)**

The name of the listener definition to be changed or created (parameter identifier: MQCACH\_LISTENER\_NAME).

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

**TransportType (MQCFIN)**

Transmission protocol (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

The value can be:

**MQXPT\_TCP**

TCP.

**MQXPT\_LU62**

LU 6.2. This value is valid only on Windows.

**MQXPT\_NETBIOS**

NetBIOS. This value is valid only on Windows.

**MQXPT\_SPX**

SPX. This value is valid only on Windows.

## Required parameters (Copy Channel Listener)

### FromListenerName (MQCFST)

The name of the listener definition to be copied from (parameter identifier: MQCACF\_FROM\_LISTENER\_NAME).

This parameter specifies the name of the existing listener definition that contains values for the attributes not specified in this command.

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

### ToListenerName (MQCFST)

To listener name (parameter identifier: MQCACF\_TO\_LISTENER\_NAME).

This parameter specifies the name of the new listener definition. If a listener definition with this name exists, *Replace* must be specified as MQRP\_YES.

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

## Optional parameters (Change, Copy, and Create Channel Listener)

### Adapter (MQCFIN)

Adapter number (parameter identifier: MQIACH\_ADAPTER).

The adapter number on which NetBIOS listens. This parameter is valid only on Windows.

### Backlog (MQCFIN)

Backlog (parameter identifier: MQIACH\_BACKLOG).

The number of concurrent connection requests that the listener supports.

### Commands (MQCFIN)

Adapter number (parameter identifier: MQIACH\_COMMAND\_COUNT).

The number of commands that the listener can use. This parameter is valid only on Windows.

### IPAddress (MQCFST)

IP address (parameter identifier: MQCACH\_IP\_ADDRESS).

IP address for the listener specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric host name form. If you do not specify a value for this parameter, the listener listens on all configured IPv4 and IPv6 stacks.

The maximum length of the string is MQ\_LOCAL\_ADDRESS\_LENGTH

### ListenerDesc (MQCFST)

Description of listener definition (parameter identifier: MQCACH\_LISTENER\_DESC).

This parameter is a plain-text comment that provides descriptive information about the listener definition. It must contain only displayable characters.

If characters are used that are not in the coded character set identifier (CCSID) for the queue manager on which the command is executing, they might be translated incorrectly.

The maximum length of the string is MQ\_LISTENER\_DESC\_LENGTH.

**LocalName (MQCFST)**

NetBIOS local name (parameter identifier: MQCACH\_LOCAL\_NAME).

The NetBIOS local name that the listener uses. This parameter is valid only on Windows.

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH

**NetbiosNames (MQCFIN)**

NetBIOS names (parameter identifier: MQIACH\_NAME\_COUNT).

The number of names that the listener supports. This parameter is valid only on Windows.

**Port (MQCFIN)**

Port number (parameter identifier: MQIACH\_PORT).

The port number for TCP/IP. This parameter is valid only if the value of *TransportType* is MQXPT\_TCP.

**Replace (MQCFIN)**

Replace attributes (parameter identifier: MQIACF\_REPLACE).

If a namelist definition with the same name as *ToListenerName* exists, this definition specifies whether it is to be replaced. The value can be:

**MQRP\_YES**

Replace existing definition.

**MQRP\_NO**

Do not replace existing definition.

**Sessions (MQCFIN)**

NetBIOS sessions (parameter identifier: MQIACH\_SESSION\_COUNT).

The number of sessions that the listener can use. This parameter is valid only on Windows.

**Socket (MQCFIN)**

SPX socket number (parameter identifier: MQIACH\_SOCKET).

The SPX socket on which to listen. This parameter is valid only if the value of *TransportType* is MQXPT\_SPX.

**StartMode (MQCFIN)**

Service mode (parameter identifier: MQIACH\_LISTENER\_CONTROL).

Specifies how the listener is to be started and stopped. The value can be any of the following values:

**MQSVC\_CONTROL\_MANUAL**

The listener is not to be started automatically or stopped automatically. It is to be controlled by user command. This value is the default value.

**MQSVC\_CONTROL\_Q\_MGR**

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

**MQSVC\_CONTROL\_Q\_MGR\_START**

The listener is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**TPName (MQCFST)**

Transaction program name (parameter identifier: MQCACH\_TP\_NAME).

The LU 6.2 transaction program name. This parameter is valid only on Windows.

The maximum length of the string is MQ\_TP\_NAME\_LENGTH

**Multi *Change, Copy, and Create Communication Information Object on Multiplatforms***

The Change Communication Information Object PCF command changes existing communication information object definitions. The Copy and Create Communication Information Object commands create

new communication information object definitions - the Copy command uses attribute values of an existing communication information object definition.

The Change communication information (MQCMD\_CHANGE\_COMM\_INFO) command changes the specified attributes of an existing IBM MQ communication information object definition. For any optional parameters that are omitted, the value does not change.

The Copy communication information (MQCMD\_COPY\_COMM\_INFO) command creates an IBM MQ communication information object definition, using, for attributes not specified in the command, the attribute values of an existing communication information definition.

The Create communication information (MQCMD\_CREATE\_COMM\_INFO) command creates an IBM MQ communication information object definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

## Required parameter (Change communication information)

### ComminfoName (MQCFST)

The name of the communication information definition to be changed (parameter identifier: MQCA\_COMM\_INFO\_NAME).

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

## Required parameters (Copy communication information)

### FromComminfoName (MQCFST)

The name of the communication information object definition to be copied from (parameter identifier: MQCACF\_FROM\_COMM\_INFO\_NAME).

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

### ToComminfoName (MQCFST)

The name of the communication information definition to copy to (parameter identifier: MQCACF\_TO\_COMM\_INFO\_NAME).

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

## Required parameters (Create communication information)

### ComminfoName (MQCFST)

The name of the communication information definition to be created (parameter identifier: MQCA\_COMM\_INFO\_NAME).

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

## Optional parameters (Change, Copy, and Create communication information)

### Bridge (MQCFIN)

Controls whether publications from applications not using Multicast are bridged to applications using multicast (parameter identifier: MQIA\_MCAST\_BRIDGE).

Bridging does not apply to topics that are marked as **MCAST (ONLY)**. As these topics can only have multicast traffic, it is not applicable to bridge to the non-multicast publish/subscribe domain.

### MQMCB\_DISABLED

Publications from applications not using multicast are not bridged to applications that do use Multicast. This is the default for IBM i.

### MQMCB\_ENABLED

Publications from applications not using multicast are bridged to applications that do use Multicast. This is the default for platforms other than IBM i. This value is not valid on IBM i.

**CCSID (MQCFIN)**

The coded character set identifier that messages are transmitted on (parameter identifier: MQIA\_CODED\_CHAR\_SET\_ID).

Specify a value in the range 1 to 65535.

The CCSID must specify a value that is defined for use on your platform, and use a character set that is appropriate to the platform. If you use this parameter to change the CCSID, applications that are running when the change is applied continue to use the original CCSID. Because of this, you must stop and restart all running applications before you continue.

This includes the command server and channel programs. To do this, stop and restart the queue manager after making the change. The default value is ASPUB which means that the coded character set is taken from the one that is supplied in the published message.

**CommEvent (MQCFIN)**

Controls whether event messages are generated for multicast handles that are created using this COMMINFO object (parameter identifier: MQIA\_COMM\_EVENT).

Events are only generated if monitoring is also enabled using the **MonitorInterval** parameter.

**MQEVR\_DISABLED**

Publications from applications not using multicast are not bridged to applications that do use multicast. This is the default value.

**MQEVR\_ENABLED**

Publications from applications not using multicast are bridged to applications that do use multicast.

**MQEVR\_EXCEPTION**

Event messages are written if the message reliability is below the reliability threshold. The reliability threshold is set to 90 by default.

**Description (MQCFST)**

Plain-text comment that provides descriptive information about the communication information object (parameter identifier: MQCA\_COMM\_INFO\_DESC).

It must contain only displayable characters. The maximum length is 64 characters. In a DBCS installation, it can contain DBCS characters (subject to a maximum length of 64 bytes).

If characters are used that are not in the coded character set identifier (CCSID) for this queue manager, they might be translated incorrectly if the information is sent to another queue manager.

The maximum length is MQ\_COMM\_INFO\_DESC\_LENGTH.

**Encoding (MQCFIN)**

The encoding that the messages are transmitted in (parameter identifier: MQIACF\_ENCODING).

**MQENC\_AS\_PUBLISHED**

The encoding of the message is taken from the one that is supplied in the published message. This is the default value.

**MQENC\_NORMAL****MQENC\_REVERSED****MQENC\_S390****MQENC\_TNS****GrpAddress (MQCFST)**

The group IP address or DNS name (parameter identifier: MQCACH\_GROUP\_ADDRESS).

It is the administrator's responsibility to manage the group addresses. It is possible for all multicast clients to use the same group address for every topic; only the messages that match outstanding subscriptions on the client are delivered. Using the same group address can be inefficient because every client must examine and process every multicast packet in the network. It is more efficient to allocate different IP group addresses to different topics or sets of topics, but this requires careful



management, especially if other non-MQ multicast applications are in use on the network. The default value is 239.0.0.0.

The maximum length is MQ\_GROUP\_ADDRESS\_LENGTH.

#### **MonitorInterval (MQCFIN)**

How frequently monitoring information is updated and event messages are generated (parameter identifier: MQIA\_MONITOR\_INTERVAL).

The value is specified as a number of seconds in the range 0 to 999 999. A value of 0 indicates that no monitoring is required.

If a non-zero value is specified, monitoring is enabled. Monitoring information is updated and event messages (if enabled using *CommEvent*, are generated about the status of the multicast handles created using this communication information object.

#### **MsgHistory (MQCFIN)**

This value is the amount of message history in kilobytes that is kept by the system to handle retransmissions in the case of NACKs (parameter identifier: MQIACH\_MSG\_HISTORY).

The value is in the range 0 to 999 999 999. A value of 0 gives the least level of reliability. The default value is 100.

#### **MulticastHeartbeat (MQCFIN)**

The heartbeat interval is measured in milliseconds, and specifies the frequency at which the transmitter notifies any receivers that there is no further data available (parameter identifier: MQIACH\_MC\_HB\_INTERVAL).

The value is in the range 0 to 999 999. The default value is 2000 milliseconds.

#### **MulticastPropControl (MQCFIN)**

The multicast properties control how many of the MQMD properties and user properties flow with the message (parameter identifier: MQIACH\_MULTICAST\_PROPERTIES).

##### **MQMCP\_ALL**

All user properties and all the fields of the MQMD are transported. This is the default value.

##### **MQMCP\_REPLY**

Only user properties, and MQMD fields that deal with replying to the messages, are transmitted. These properties are:

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

##### **MQMCP\_USER**

Only the user properties are transmitted.

##### **MQMCP\_NONE**

No user properties or MQMD fields are transmitted.

##### **MQMCP\_COMPAT**

Properties are transmitted in a format compatible with previous MQ multicast clients.

#### **NewSubHistory (MQCFIN)**

The new subscriber history controls whether a subscriber joining a publication stream receives as much data as is currently available, or receives only publications made from the time of the subscription (parameter identifier: MQIACH\_NEW\_SUBSCRIBER\_HISTORY).

##### **MQNSH\_NONE**

A value of NONE causes the transmitter to transmit only publication made from the time of the subscription. This is the default value.

## **MQNSH\_ALL**

A value of ALL causes the transmitter to retransmit as much history of the topic as is known. In some circumstances, this can give a similar behavior to retained publications.

Using the value of MQNSH\_ALL might have a detrimental effect on performance if there is a large topic history because all the topic history is retransmitted.

## **PortNumber (MQCFIN)**

The port number to transmit on (parameter identifier: MQIACH\_PORT).

The default port number is 1414.

## **Type (MQCFIN)**

The type of the communications information object (parameter identifier: MQIA\_COMM\_INFO\_TYPE).

The only type supported is MQCIT\_MULTICAST.

## **Change, Copy, and Create Namelist**

The Change Namelist PCF command changes existing namelist definitions. The Copy and Create Namelist commands create new namelist definitions - the Copy command uses attribute values of an existing namelist definition.

The Change Namelist (MQCMD\_CHANGE\_NAMELIST) command changes the specified attributes of an existing IBM MQ namelist definition. For any optional parameters that are omitted, the value does not change.

The Copy Namelist (MQCMD\_COPY\_NAMELIST) command creates an IBM MQ namelist definition, using, for attributes not specified in the command, the attribute values of an existing namelist definition.

The Create Namelist (MQCMD\_CREATE\_NAMELIST) command creates an IBM MQ namelist definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

## **Required parameter (Change and Create Namelist)**

### **NamelistName (MQCFST)**

The name of the namelist definition to be changed (parameter identifier: MQCA\_NAMELIST\_NAME).


The maximum length of the string is MQ\_NAMELIST\_NAME\_LENGTH.

## **Required parameters (Copy Namelist)**

### **FromNamelistName (MQCFST)**

The name of the namelist definition to be copied from (parameter identifier: MQCACF\_FROM\_NAMELIST\_NAME).

This parameter specifies the name of the existing namelist definition that contains values for the attributes not specified in this command.

 On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD\_Q\_MGR or MQQSGD\_COPY to copy from. This parameter is ignored if a value of MQQSGD\_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToNamelistName* and the disposition MQQSGD\_GROUP is searched for to copy from.

The maximum length of the string is MQ\_NAMELIST\_NAME\_LENGTH.

### **ToNamelistName (MQCFST)**

To namelist name (parameter identifier: MQCACF\_TO\_NAMELIST\_NAME).

This parameter specifies the name of the new namelist definition. If a namelist definition with this name exists, *Replace* must be specified as MQRP\_YES.

The maximum length of the string is MQ\_NAMELIST\_NAME\_LENGTH.

## Optional parameters (Change, Copy, and Create Namelist)

z/OS

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### NamelistDesc (MQCFST)

Description of namelist definition (parameter identifier: MQCA\_NAMELIST\_DESC).

This parameter is a plain-text comment that provides descriptive information about the namelist definition. It must contain only displayable characters.

If characters are used that are not in the coded character set identifier (CCSID) for the queue manager on which the command is executing, they might be translated incorrectly.

The maximum length of the string is MQ\_NAMELIST\_DESC\_LENGTH.

### NamelistType (MQCFIN)

Type of names in the namelist (parameter identifier: MQIA\_NAMELIST\_TYPE). This parameter applies to z/OS only.

Specifies the type of names in the namelist. The value can be any of the following values:

#### **MQNT\_NONE**

The names are of no particular type.

#### **MQNT\_Q**

A namelist that holds a list of queue names.

#### **MQNT\_CLUSTER**

A namelist that is associated with clustering, containing a list of the cluster names.

#### **MQNT\_AUTH\_INFO**

The namelist is associated with TLS, and contains a list of authentication information object names.

### Names (MQCFSL)

The names to be placed in the namelist (parameter identifier: MQCA\_NAMES).

The number of names in the list is given by the *Count* field in the MQCFSL structure. The length of each name is given by the *StringLength* field in that structure. The maximum length of a name is MQ\_OBJECT\_NAME\_LENGTH.

### QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

Table 198. QSGDisposition: Where objects are defined and how they behave

QSGDisposition	Change	Copy, Create
<b>MQQSGD_COPY</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command. It uses the MQQSGD_GROUP object of the same name as the <i>ToNameListName</i> object (for Copy) or <i>NameListName</i> object (for Create).
<b>MQQSGD_GROUP</b>	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group so that they refresh local copies on page set zero:</p> <pre>DEFINE NAMELIST(name) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This is allowed only if the queue manager is in a queue sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group so that they make or refresh local copies on page set zero:</p> <pre>DEFINE NAMELIST(name) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
<b>MQQSGD_PRIVATE</b>	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.
<b>MQQSGD_Q_MGR</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This value is the default value.	The object is defined on the page set of the queue manager that executes the command. This value is the default value.

**Replace (MQCFIN)**

Replace attributes (parameter identifier: MQIACF\_REPLACE).

If a namelist definition with the same name as *ToNameListName* exists, this definition specifies whether it is to be replaced. The value can be:

**MQRP\_YES**

Replace existing definition.

## **MQRP\_NO**

Do not replace existing definition.

### ***Change, Copy, and Create Process***

The Change Process PCF command changes existing process definitions. The Copy and Create Process commands create new process definitions - the Copy command uses attribute values of an existing process definition.

The Change Process (MQCMD\_CHANGE\_PROCESS) command changes the specified attributes of an existing IBM MQ process definition. For any optional parameters that are omitted, the value does not change.

The Copy Process (MQCMD\_COPY\_PROCESS) command creates an IBM MQ process definition, using, for attributes not specified in the command, the attribute values of an existing process definition.

The Create Process (MQCMD\_CREATE\_PROCESS) command creates an IBM MQ process definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

### **Required parameters (Change and Create Process)**

#### **ProcessName (MQCFST)**

The name of the process definition to be changed or created (parameter identifier: MQCA\_PROCESS\_NAME).


The maximum length of the string is MQ\_PROCESS\_NAME\_LENGTH.

### **Required parameters (Copy Process)**

#### **FromProcessName (MQCFST)**

The name of the process definition to be copied from (parameter identifier: MQCACF\_FROM\_PROCESS\_NAME).

Specifies the name of the existing process definition that contains values for the attributes not specified in this command.

 On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD\_Q\_MGR or MQQSGD\_COPY to copy from. This parameter is ignored if a value of MQQSGD\_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToProcessName* and the disposition MQQSGD\_GROUP is searched for to copy from.

The maximum length of the string is MQ\_PROCESS\_NAME\_LENGTH.

#### **ToProcessName (MQCFST)**

To process name (parameter identifier: MQCACF\_TO\_PROCESS\_NAME).

The name of the new process definition. If a process definition with this name exists, *Replace* must be specified as MQRP\_YES.

The maximum length of the string is MQ\_PROCESS\_NAME\_LENGTH.

### **Optional parameters (Change, Copy, and Create Process)**

#### **ApplId (MQCFST)**

Application identifier (parameter identifier: MQCA\_APPL\_ID).

*ApplId* is the name of the application to be started. The application must be on the platform for which the command is executing. The name might typically be a fully qualified file name of an executable object. Qualifying the file name is particularly important if you have multiple IBM MQ installations, to ensure the correct version of the application is run.

The maximum length of the string is MQ\_PROCESS\_APPL\_ID\_LENGTH.

#### **ApplType (MQCFIN)**

Application type (parameter identifier: MQIA\_APPL\_TYPE).

Valid application types are:

**MQAT\_OS400**

IBM i application.

**MQAT\_DOS**

DOS client application.

**MQAT\_WINDOWS**

IBM MQ MQI client application.

**MQAT\_AIX**

AIX application (same value as MQAT\_UNIX).

**MQAT\_CICS**

CICS transaction.

 **MQAT\_ZOS**






z/OS application.

**MQAT\_DEFAULT**

Default application type.

*integer*: System-defined application type in the range zero through 65 535 or a user-defined application type in the range 65 536 through 999 999 999 (not checked).

Only specify application types (other than user-defined types) that are supported on the platform at which the command is executed:

-  On IBM i: MQAT\_OS400, MQAT\_CICS, and MQAT\_DEFAULT are supported.
-   On AIX and Linux: MQAT\_UNIX, MQAT\_OS2, MQAT\_DOS, MQAT\_WINDOWS, MQAT\_CICS, and MQAT\_DEFAULT are supported.
-  On Windows: MQAT\_WINDOWS\_NT, MQAT\_OS2, MQAT\_DOS, MQAT\_WINDOWS, MQAT\_CICS, and MQAT\_DEFAULT are supported.
-  On z/OS: MQAT\_DOS, MQAT\_IMS, MQAT\_MVS, MQAT\_UNIX, MQAT\_CICS, and MQAT\_DEFAULT are supported.



**CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- Blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- A queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. In a shared queue environment, you can provide a different queue manager name from the one you are using to enter the command. The command server must be enabled.
- An asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

**EnvData (MQCFST)**

Environment data (parameter identifier: MQCA\_ENV\_DATA).

A character string that contains environment information pertaining to the application to be started.

The maximum length of the string is MQ\_PROCESS\_ENV\_DATA\_LENGTH.

### ProcessDesc (MQCFST)

Description of process definition (parameter identifier: MQCA\_PROCESS\_DESC).

A plain-text comment that provides descriptive information about the process definition. It must contain only displayable characters.

The maximum length of the string is MQ\_PROCESS\_DESC\_LENGTH.

Use characters from the coded character set identifier (CCSID) for this queue manager. Other characters might be translated incorrectly if the information is sent to another queue manager.

### QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

QSGDisposition	Change	Copy, Create
<b>MQQSGD_COPY</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command. It uses the MQQSGD_GROUP object of the same name as the <i>ToProcessName</i> object (for Copy) or <i>ProcessName</i> object (for Create).
<b>MQQSGD_GROUP</b>	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(GROUP). On the page set of the queue manager that executes the command, only a local copy of the object is altered by this command. If the command is successful, the following command is generated.</p> <pre>DEFINE PROCESS(process-name) REPLACE QSGDISP(COPY)</pre> <p>The command is sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero. The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. GROUP is allowed only if the queue manager is in a queue sharing group. If the definition is successful, the following command is generated.</p> <pre>DEFINE PROCESS(process-name) REPLACE QSGDISP(COPY)</pre> <p>The command is sent to all active queue managers in the queue sharing group to attempt to make or refresh local copies on page set zero. The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
<b>MQQSGD_PRIVATE</b>	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.

Table 199. QSGDisposition: Where objects are defined and how they behave (continued)

QSGDisposition	Change	Copy, Create
<b>MQQSGD_Q_MGR</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. MQQSGD_Q_MGR is the default value.	The object is defined on the page set of the queue manager that executes the command. MQQSGD_Q_MGR is the default value.

### Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF\_REPLACE).

If a process definition with the same name as *ToProcessName* exists, specify whether to replace it.

The value can be any of the following values:

#### MQRP\_YES

Replace existing definition.

#### MQRP\_NO

Do not replace existing definition.

### UserData (MQCFST)

User data (parameter identifier: MQCA\_USER\_DATA).

A character string that contains user information pertaining to the application (defined by *AppId*) that is to be started.

For Microsoft Windows, the character string must not contain double quotation marks if the process definition is going to be passed to **runmqtrm**.

The maximum length of the string is MQ\_PROCESS\_USER\_DATA\_LENGTH.

### Change, Copy, and Create Queue

The Change Queue PCF command changes existing queue definitions. The Copy and Create Queue commands create new queue definitions - the Copy command uses attribute values of an existing queue definition.

The Change Queue command MQCMD\_CHANGE\_Q changes the specified attributes of an existing IBM MQ queue. For any optional parameters that are omitted, the value does not change.

The Copy Queue command MQCMD\_COPY\_Q creates a queue definition of the same type. For attributes not specified in the command, it uses the attribute values of an existing queue definition.

The Create Queue command MQCMD\_CREATE\_Q creates a queue definition with the specified attributes. All attributes that are not specified are set to the default value for the type of queue that is created.

### Required parameters (Change and Create Queue)

#### QName (MQCFST)

Queue name (parameter identifier: MQCA\_Q\_NAME).

The name of the queue to be changed. The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### Required parameters (Copy Queue)

#### FromQName (MQCFST)

From queue name (parameter identifier: MQCACF\_FROM\_Q\_NAME).



Specifies the name of the existing queue definition.

**z/OS** On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD\_Q\_MGR, MQQSGD\_COPY, or MQQSGD\_SHARED to copy from. This parameter is ignored if a value of MQQSGD\_COPY is specified for *QSGDDisposition*. In this case, an object with the name specified by *ToQName* and the disposition MQQSGD\_GROUP is searched for to copy from.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **ToQName (MQCFST)**

To queue name (parameter identifier: MQCACF\_TO\_Q\_NAME).

Specifies the name of the new queue definition.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

Queue names must be unique; if a queue definition exists with the name and type of the new queue, *Replace* must be specified as MQRP\_YES. If a queue definition exists with the same name as and a different type from the new queue, the command fails.

## **Required parameters (all commands)**

### **QType (MQCFIN)**

Queue type (parameter identifier: MQIA\_Q\_TYPE).

The value specified must match the type of the queue being changed.

The value can be any of the following values:

#### **MQQT\_ALIAS**

Alias queue definition.

#### **MQQT\_LOCAL**

Local queue.

#### **MQQT\_REMOTE**

Local definition of a remote queue.

#### **MQQT\_MODEL**

Model queue definition.

## **Optional parameters (Change, Copy, and Create Queue)**

### **BackoutRequeueName (MQCFST) - see MQSC BOQNAME**

Excessive backout requeue name (parameter identifier: MQCA\_BACKOUT\_REQ\_Q\_NAME).

Specifies the name of the queue to which a message is transferred if it is backed out more times than the value of *BackoutThreshold*. The queue does not have to be a local queue.

The backout queue does not need to exist at this time but it must exist when the *BackoutThreshold* value is exceeded.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **BackoutThreshold (MQCFIN)**

Backout threshold (parameter identifier: MQIA\_BACKOUT\_THRESHOLD).

The number of times a message can be backed out before it is transferred to the backout queue specified by *BackoutRequeueName*.

If the value is later reduced, messages that are already on the queue that were backed out at least as many times as the new value remain on the queue. Those messages are transferred if they are backed out again.

Specify a value in the range 0 - 999,999,999.

### **BaseObjectName (MQCFST)**

Name of the object to which the alias resolves (parameter identifier: MQCA\_BASE\_OBJECT\_NAME).

This parameter is the name of a queue or topic that is defined to the local queue manager.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

### BaseQName (MQCFST)

Queue name to which the alias resolves (parameter identifier: MQCA\_BASE\_Q\_NAME).

This parameter is the name of a local or remote queue that is defined to the local queue manager.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

V 9.3.1

Multi

### CapExpiry (MQCFIN)

Capped message expiry processing (parameter identifier MQIA\_CAP\_EXPIRY) which can be an integer value or can take the value of MQCEX\_NOLIMIT.

Specifies a lifetime limit for messages put using the object, expressed in 10ths of a second. A value of -1, displayed as NOLIMIT, has no effect on processing.

Note that in addition to the CapExpiry attribute itself, you can use an attribute of the **CUSTOM** parameter, which is a string. Therefore passing a string PCF parameter MQCA\_CUSTOM, has a string value of CAPEXPY (integer). See [Making CAPEXPY a first-class MQSC attribute in MQ 9.3.1](#) for more information.

CapExpiry provides, or limits, the value in the MQMD [Expiry](#) field of any message put.

An application provided MQMD **Expiry** that is less than any resolved CapExpiry value is passed. This value is not replaced by that resolved CapExpiry value.

This process allows an IBM MQ administrator to limit the life of messages put by an application that overlooked (or was unable to provide, in the case of MQTT) message expiration criteria.

However, this option does not allow an administrator to override application behavior where the required lifetime of messages was under-estimated.

If more than one object is used on the resolution path, for example, alias queue-> remote queue-> transmission queue, then the smallest of all their non-zero CapExpiry values is used as the upper limit for expiry.

The new capped value for expiry is used during the put processing as if it had been provided by the application in the MQMD structure.

The *capped* value is evaluated for each put being performed, and so is sensitive to the resolution of the put operation. For example, in a cluster, where the put operation is performed with BIND NOT FIXED, messages might pick up different expiry values depending on the CapExpiry value set for the transmission queue used by the channel.

z/OS

### CFStructure (MQCFST)

Coupling facility structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME). This parameter applies to z/OS only.

Specifies the name of the coupling facility structure where you want to store messages when you use shared queues. The name:

- Cannot have more than 12 characters
- Must start with an uppercase letter (A - Z)
- Can include only the characters A - Z and 0 - 9

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

The name of the queue sharing group to which the queue manager is connected is prefixed to the name you supply. The name of the queue sharing group is always four characters, padded with @ symbols if necessary. For example, if you use a queue sharing group named NY03 and you supply the name PRODUCT7, the resultant coupling facility structure name is NY03PRODUCT7. Note the administrative structure for the queue sharing group (in this case NY03CSQ\_ADMIN) cannot be used for storing messages.

For local and model queues, the following rules apply. The rules apply if you use the Create Queue command with a value of MQRP\_YES in the **Replace** parameter. The rules also apply if you use the Change Queue command.

- On a local queue with a value of MQQSGD\_SHARED in the **QSGDisposition** parameter, *CFStructure* cannot change.

If you need to change either the *CFStructure* or *QSGDisposition* value, you must delete and redefine the queue. To preserve any of the messages on the queue you must offload the messages before you delete the queue. Reload the messages after you redefine the queue, or move the messages to another queue.

- On a model queue with a value of MQQDT\_SHARED\_DYNAMIC in the **DefinitionType** parameter, *CFStructure* cannot be blank.
- On a local queue with a value other than MQQSGD\_SHARED in the **QSGDisposition** parameter, the value of *CFStructure* does not matter. The value *CFStructure* also does not matter for a model queue with a value other than MQQDT\_SHARED\_DYNAMIC in the **DefinitionType** parameter.

For local and model queues, when you use the Create Queue command with a value of MQRP\_NO in the **Replace** parameter, the coupling facility structure:

- On a local queue with a value of MQQSGD\_SHARED in the **QSGDisposition** parameter, or a model queue with a value of MQQDT\_SHARED\_DYNAMIC in the **DefinitionType** parameter, *CFStructure* cannot be blank.
- On a local queue with a value other than MQQSGD\_SHARED in the **QSGDisposition** parameter, the value of *CFStructure* does not matter. The value *CFStructure* also does not matter for a model queue with a value other than MQQDT\_SHARED\_DYNAMIC in the **DefinitionType** parameter.

**Note:** Before you can use the queue, the structure must be defined in the coupling facility Resource Management (CFRM) policy data set.

### ClusterChannelName (MQCFST)

This parameter is supported only on transmission queues.

`ClusterChannelName` is the generic name of the cluster-sender channels that use this queue as a transmission queue. The attribute specifies which cluster-sender channels send messages to a cluster-receiver channel from this cluster transmission queue. (Parameter identifier: MQCA\_CLUS\_CHL\_NAME.)

You can also set the transmission queue attribute `ClusterChannelName` to a cluster-sender channel manually. Messages that are destined for the queue manager connected by the cluster-sender channel are stored in the transmission queue that identifies the cluster-sender channel. They are not stored in the default cluster transmission queue. If you set the `ClusterChannelName` attribute to blanks, the channel switches to the default cluster transmission queue when the channel restarts. The default queue is either `SYSTEM.CLUSTER.TRANSMIT.ChannelName` or `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, depending on the value of the queue manager `DefClusterXmitQueueType` attribute.

By specifying asterisks, "\*", in **ClusterChannelName**, you can associate a transmission queue with a set of cluster-sender channels. The asterisks can be at the beginning, end, or any number of places in the middle of the channel name string. **ClusterChannelName** is limited to a length of 20 characters: MQ\_CHANNEL\_NAME\_LENGTH.

The default queue manager configuration is for all cluster-sender channels to send messages from a single transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. The default configuration can be changed by modified by changing the queue manager attribute, **DefClusterXmitQueueType**. The default value of the attribute is `SCTQ`. You can change the value to `CHANNEL`. If you set the **DefClusterXmitQueueType** attribute to `CHANNEL`, each cluster-sender channel defaults to using a specific cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.ChannelName`.

### ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

The name of the cluster to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are open.

Only one of the resultant values of **ClusterName** and **ClusterNameList** can be nonblank; you cannot specify a value for both.

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

### **ClusterNameList (MQCFST)**

Cluster namelist (parameter identifier: MQCA\_CLUSTER\_NAMELIST).

The name of the namelist, that specifies a list of clusters to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are open.

Only one of the resultant values of **ClusterName** and **ClusterNameList** can be nonblank; you cannot specify a value for both.

### **CLWLQueuePriority (MQCFIN)**

Cluster workload queue priority (parameter identifier: MQIA\_CLWL\_Q\_PRIORITY).

Specifies the priority of the queue in cluster workload management; see [Configuring a queue manager cluster](#). The value must be in the range 0 - 9, where 0 is the lowest priority and 9 is the highest.

### **CLWLQueueRank (MQCFIN)**

Cluster workload queue rank (parameter identifier: MQIA\_CLWL\_Q\_RANK).

Specifies the rank of the queue in cluster workload management. The value must be in the range 0 - 9, where 0 is the lowest priority and 9 is the highest.

### **CLWLUseQ (MQCFIN)**

Cluster workload use remote queue (parameter identifier: MQIA\_CLWL\_USEQ).

Specifies whether remote and local queues are to be used in cluster workload distribution. The value can be any of the following values:

#### **MQCLWL\_USEQ\_AS\_Q\_MGR**

Use the value of the **CLWLUseQ** parameter on the definition of the queue manager.

#### **MQCLWL\_USEQ\_ANY**

Use remote and local queues.

#### **MQCLWL\_USEQ\_LOCAL**

Do not use remote queues.

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is run when the queue manager is a member of a queue sharing group. You can specify one of the following values:

- Blank, or omit the parameter altogether. The command is run on the queue manager on which it was entered.
- A queue manager name. The command is run on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment. The command server must be enabled.
- An asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **Custom (MQCFST)**

Custom attribute for new features (parameter identifier: MQCA\_CUSTOM).

This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE) . Single quotation marks must be escaped with another single quotation mark.

**CAEXPRY (integer )**

The maximum time, expressed in tenths of a second, until a message put using an object handle, opened using this object on the resolution path, remains in the system until it becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

The value can be one of the following:

**integer**

The value must be in the range one through to 999 999 999.

**NOLIMIT**

There is no limit on the expiry time of messages put using this object. This is the default value.

Specifying a value for CAEXPRY that is not valid, does not cause the command to fail. Instead, the default value is used.

**DefaultPutResponse (MQCFIN)**

Default put response type definition (parameter identifier: MQIA\_DEF\_PUT\_RESPONSE\_TYPE).

The parameter specifies the type of response to be used for put operations to the queue when an application specifies MQPMO\_RESPONSE\_AS\_Q\_DEF. The value can be any of the following values:

**MQPRT\_SYNC\_RESPONSE**

The put operation is issued synchronously, returning a response.

**MQPRT\_ASYNC\_RESPONSE**

The put operation is issued asynchronously, returning a subset of MQMD fields.

**DefBind (MQCFIN)**

Bind definition (parameter identifier: MQIA\_DEF\_BIND).

The parameter specifies the binding to be used when MQOO\_BIND\_AS\_Q\_DEF is specified on the MQOPEN call. The value can be any of the following values:

**MQBND\_BIND\_ON\_OPEN**

The binding is fixed by the MQOPEN call.

**MQBND\_BIND\_NOT\_FIXED**

The binding is not fixed.

**MQBND\_BIND\_ON\_GROUP**

Allows an application to request that a group of messages are all allocated to the same destination instance.

Changes to this parameter do not affect instances of the queue that are open.

**DefinitionType (MQCFIN)**

Queue definition type (parameter identifier: MQIA\_DEFINITION\_TYPE).

The value can be any of the following values:

**MQQDT\_PERMANENT\_DYNAMIC**

Dynamically defined permanent queue.

**MQQDT\_SHARED\_DYNAMIC**

Dynamically defined shared queue. This option is available on z/OS only.

**MQQDT\_TEMPORARY\_DYNAMIC**

Dynamically defined temporary queue.

**DefInputOpenOption (MQCFIN)**

Default input open option (parameter identifier: MQIA\_DEF\_INPUT\_OPEN\_OPTION).

Specifies the default share option for applications opening this queue for input.

The value can be any of the following values:

**MQOO\_INPUT\_EXCLUSIVE**

Open queue to get messages with exclusive access.

**MQOO\_INPUT\_SHARED**

Open queue to get messages with shared access.

**DefPersistence (MQCFIN)**

Default persistence (parameter identifier: MQIA\_DEF\_PERSISTENCE).

Specifies the default for message-persistence on the queue. Message persistence determines whether messages are preserved across restarts of the queue manager.

The value can be any of the following values:

**MQPER\_PERSISTENT**

Message is persistent.

**MQPER\_NOT\_PERSISTENT**

Message is not persistent.

**DefPriority (MQCFIN)**

Default priority (parameter identifier: MQIA\_DEF\_PRIORITY).

Specifies the default priority of messages put on the queue. The value must be in the range zero through to the maximum priority value that is supported (9).

**DefReadAhead (MQCFIN)**

Default read ahead (parameter identifier: MQIA\_DEF\_READ\_AHEAD).

Specifies the default read ahead behavior for non-persistent messages delivered to the client.

The value can be any of the following values:

**MQREADA\_NO**

Non-persistent messages are not read ahead unless the client application is configured to request read ahead.

**MQREADA\_YES**

Non-persistent messages are sent ahead to the client before an application requests them. Non-persistent messages can be lost if the client ends abnormally or if the client does not consume all the messages it is sent.

**MQREADA\_DISABLED**

Read ahead of non-persistent messages is not enabled for this queue. Messages are not sent ahead to the client regardless of whether read ahead is requested by the client application.

**Multi DistLists (MQCFIN)**

Distribution list support (parameter identifier: MQIA\_DIST\_LISTS).

Specifies whether distribution-list messages can be placed on the queue.

**Note:** This attribute is set by the sending message channel agent (MCA). The sending MCA removes messages from the queue each time it establishes a connection to a receiving MCA on a partner queue manager. The attribute is not normally set by administrators, although it can be set if the need arises.

This parameter is supported on Multiplatforms.

The value can be any of the following values:

**MQDL\_SUPPORTED**

Distribution lists supported.

**MQDL\_NOT\_SUPPORTED**

Distribution lists not supported.

**Force (MQCFIN)**

Force changes (parameter identifier: MQIACF\_FORCE).

Specifies whether the command must be forced to complete when conditions are such that completing the command would affect an open queue. The conditions depend upon the type of the queue that is being changed:

#### **QALIAS**

*BaseQName* is specified with a queue name and an application has the alias queue open.

#### **QLOCAL**

Either of the following conditions indicates that a local queue would be affected:

- *Shareability* is specified as MQQA\_NOT\_SHAREABLE and more than one application has the local queue open for input.
- The *Usage* value is changed and one or more applications has the local queue open, or there are one or more messages on the queue. (The *Usage* value must not normally be changed while there are messages on the queue. The format of messages changes when they are put on a transmission queue.)

#### **QREMOTE**

Either of the following conditions indicates that a remote queue would be affected:

- If *XmitQName* is specified with a transmission-queue name, or blank, and an application has a remote queue open that would be affected by this change.
- If any of the following parameters are specified with a queue or queue manager name, and one or more applications has a queue open that resolved through this definition as a queue manager alias. The parameters are:

1. *RemoteQName*
2. *RemoteQMGrName*
3. *XmitQName*

#### **QMODEL**

This parameter is not valid for model queues.

**Note:** A value of MQFC\_YES is not required if this definition is in use as a reply-to queue definition only.

The value can be any of the following values:

#### **MQFC\_YES**

Force the change.

#### **MQFC\_NO**

Do not force the change.

#### **HardenGetBackout (MQCFIN)**

Harden the backout count, or not (parameter identifier: MQIA\_HARDEN\_GET\_BACKOUT).

Specifies whether the count of the number of times that a message was backed out is hardened. When the count is hardened, the value of the **BackoutCount** field of the message descriptor is written to the log before the message is returned by an MQGET operation. Writing the value to the log ensures that the value is accurate across restarts of the queue manger.

**Note:** IBM MQ for IBM i always hardens the count, regardless of the setting of this attribute.

When the backout count is hardened, the performance of MQGET operations for persistent messages on this queue is impacted.

The value can be any of the following values:

#### **MQQA\_BACKOUT\_HARDENED**

The message backout count for messages on this queue is hardened to ensure that the count is accurate.

#### **MQQA\_BACKOUT\_NOT\_HARDENED**

The message backout count for messages on this queue is not hardened and might not be accurate over queue manager restarts.

### ImageRecoverQueue (MQCFST)

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used (parameter identifier: MQIA\_MEDIA\_IMAGE\_RECOVER\_Q).

This parameter is not valid on z/OS. Possible values are:

#### MQIMGRCOV\_YES

These queue objects are recoverable.

#### MQIMGRCOV\_NO

The “rcdmqimg (record media image)” on page 139 and “rcrmqobj (re-create object)” on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

#### MQIMGRCOV\_AS\_Q\_MGR

If you specify MQIMGRCOV\_AS\_Q\_MGR , and the **ImageRecoverQueue** attribute for the queue manager specifies MQIMGRCOV\_YES , these queue objects are recoverable.

If you specify MQIMGRCOV\_AS\_Q\_MGR and the **ImageRecoverQueue** attribute for the queue manager specifies MQIMGRCOV\_NO, the “rcdmqimg (record media image)” on page 139 and “rcrmqobj (re-create object)” on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

MQIMGRCOV\_AS\_Q\_MGR is the default value.

### IndexType (MQCFIN)

Index type (parameter identifier: MQIA\_INDEX\_TYPE). This parameter applies to z/OS only.

Specifies the type of index maintained by the queue manager to expedite MQGET operations on the queue. For shared queues, the type of index determines what type of MQGET calls can be used. The value can be any of the following values:

#### MQIT\_NONE

No index.

#### MQIT\_MSG\_ID

The queue is indexed using message identifiers.

#### MQIT\_CORREL\_ID

The queue is indexed using correlation identifiers.

#### MQIT\_MSG\_TOKEN

**Important:** This index type should only be used for queues used with the IBM MQ Workflow for z/OS product.

The queue is indexed using message tokens.

#### MQIT\_GROUP\_ID

The queue is indexed using group identifiers.

Messages can be retrieved using a selection criterion only if an appropriate index type is maintained, as the following table shows:

Retrieval selection criterion	IndexType required	
	Shared queue	Other queue
None (sequential retrieval)	Any	Any
Message identifier	MQIT_MSG_ID or MQIT_NONE	Any
Correlation identifier	MQIT_CORREL_ID	Any
Message and correlation identifiers	MQIT_MSG_ID or MQIT_CORREL_ID	Any



Table 200. Retrieval selection criteria and index types (continued)

Retrieval selection criterion	<i>IndexType</i> required	
Group identifier	MQIT_GROUP_ID	Any
Grouping	MQIT_GROUP_ID	MQIT_GROUP_ID
Message token	Not allowed	MQIT_MSG_TOKEN

**InhibitGet (MQCFIN)**

Get operations are allowed or inhibited (parameter identifier: MQIA\_INHIBIT\_GET).

The value can be:

**MQQA\_GET\_ALLOWED**

Get operations are allowed.

**MQQA\_GET\_INHIBITED**

Get operations are inhibited.

**InhibitPut (MQCFIN)**

Put operations are allowed or inhibited (parameter identifier: MQIA\_INHIBIT\_PUT).

Specifies whether messages can be put on the queue.

The value can be any of the following values:

**MQQA\_PUT\_ALLOWED**

Put operations are allowed.

**MQQA\_PUT\_INHIBITED**

Put operations are inhibited.

**InitiationQName (MQCFST)**

Initiation queue name (parameter identifier: MQCA\_INITIATION\_Q\_NAME).

The local queue for trigger messages relating to this queue. The initiation queue must be on the same queue manager.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**MaxMsgLength (MQCFIN)**

Maximum message length (parameter identifier: MQIA\_MAX\_MSG\_LENGTH).

The maximum length for messages on the queue. Applications might use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue. If you change this value it might cause an application to operate incorrectly.

Do not set a value that is greater than the *MaxMsgLength* attribute of a queue manager.

The lower limit for this parameter is 0. The upper limit depends on the environment:

- On AIX, Linux, Windows, IBM i, and z/OS, the maximum message length is 100 MB (104,857,600 bytes).
- On other UNIX systems, the maximum message length is 4 MB (4,194,304 bytes).

**MaxQDepth (MQCFIN)**

Maximum queue depth (parameter identifier: MQIA\_MAX\_Q\_DEPTH).

The maximum number of messages allowed on the queue.

**Note:** Other factors might cause the queue to be treated as full. For example, it appears to be full if there is no storage available for a message.

Specify a value greater than or equal to 0, and less than or equal to 999,999,999.

**Multi MaxQFileSize (MQCFIN)**

Maximum queue depth (parameter identifier: MQIA\_MAX\_Q\_FILE\_SIZE).

The maximum size, in megabytes, that a queue file can grow to.

It is possible for a queue file to exceed the maximum size, if it is configured to a value lower than the current queue file size. If that happens the queue file no longer accepts new messages, but allows existing messages to be consumed. When the queue file size has dropped below the configured value, new messages are allowed to be put to the queue.

When displayed in queue status, this attribute indicates the current maximum size the queue file can grow to.

**Note:** This figure can differ from the value of the attribute configured on the queue because internally the queue manager might need to use a larger block size to reach the chosen size. See [Modifying IBM MQ queue files](#) for more information on changing the size of queue files and block size and granularity.

When the granularity needs changing because this attribute has been increased, warning message AMQ7493W Granularity changed is written to the AMQERR logs. This gives you an indication that you need to plan for the queue to be emptied, in order for IBM MQ to adopt the new granularity.

Specify a value greater than or equal to 20, and less than or equal to 267,386,880.

### **MsgDeliverySequence (MQCFIN)**

Messages are delivered in priority order or sequence (parameter identifier: MQIA\_MSG\_DELIVERY\_SEQUENCE).

The value can be any of the following values:

#### **MQMDS\_PRIORITY**

Messages are returned in priority order.

#### **MQMDS\_FIFO**

Messages are returned in FIFO order (first in, first out).

### **NonPersistentMessageClass (MQCFIN)**

The level of reliability to be assigned to non-persistent messages that are put to the queue (parameter identifier: MQIA\_NPM\_CLASS).

The value can be:

#### **MQNPM\_CLASS\_NORMAL**

Non-persistent messages persist as long as the lifetime of the queue manager session. They are discarded in the event of a queue manager restart. This value is the default value.

#### **MQNPM\_CLASS\_HIGH**

The queue manager attempts to retain non-persistent messages for the lifetime of the queue. Non-persistent messages might still be lost in the event of a failure.

This parameter is valid only on local and model queues. It is not valid on z/OS.

### **ProcessName (MQCFST)**

Name of process definition for the queue (parameter identifier: MQCA\_PROCESS\_NAME).

Specifies the local name of the IBM MQ process that identifies the application to be started when a trigger event occurs.

- If the queue is a transmission queue, the process definition contains the name of the channel to be started. This parameter is optional for transmission queues. If you do not specify it, the channel name is taken from the value specified for the **TriggerData** parameter.
- In other environments, the process name must be nonblank for a trigger event to occur, although it can be set after creating the queue.

The maximum length of the string is MQ\_PROCESS\_NAME\_LENGTH.

### **PropertyControl (MQCFIN)**

Property control attribute (parameter identifier: MQIA\_PROPERTY\_CONTROL).

Specifies how message properties are handled when messages are retrieved from queues using the MQGET call with the MQGMO\_PROPERTIES\_AS\_Q\_DEF option. The value can be any of the following values:

### **MQPROP\_COMPATIBILITY**

If the message contains a property with a prefix of **mcd.**, **jms.**, **usr.** or **mqext.**, all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except those properties contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

This value is the default value. It allows applications which expect JMS-related properties to be in an MQRFH2 header in the message data to continue to work unmodified.

### **MQPROP\_NONE**

All properties of the message are removed from the message before the message is sent to the remote queue manager. Properties in the message descriptor, or extension, are not removed.

### **MQPROP\_ALL**

All properties of the message are included with the message when it is sent to the remote queue manager. The properties, except those properties in the message descriptor (or extension), are placed in one or more MQRFH2 headers in the message data.

### **MQPROP\_FORCE\_MQRFH2**

Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

A valid message handle supplied in the `MsgHandle` field of the `MQGMO` structure on the `MQGET` call is ignored. Properties of the message are not accessible using the message handle.

### **MQPROP\_V6COMPAT**

Any application MQRFH2 header is received as it was sent. Any properties set using `MQSETMP` must be retrieved using `MQINQMP`. They are not added to the MQRFH2 created by the application. Properties that were set in the MQRFH2 header by the sending application cannot be retrieved using `MQINQMP`.

This parameter is applicable to Local, Alias, and Model queues.

### **QDepthHighEvent (MQCFIN)**

Controls whether Queue Depth High events are generated (parameter identifier: `MQIA_Q_DEPTH_HIGH_EVENT`).

A Queue Depth High event indicates that an application put a message on a queue. This event caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the **QDepthHighLimit** parameter.

**Note:** The value of this attribute can change implicitly; see [“Definitions of the Programmable Command Formats”](#) on page 983.

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

### **QDepthHighLimit (MQCFIN)**

High limit for queue depth (parameter identifier: `MQIA_Q_DEPTH_HIGH_LIMIT`).

The threshold against which the queue depth is compared to generate a Queue Depth High event.

This event indicates that an application put a message to a queue. This event caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the **QDepthHighEvent** parameter.

The value is expressed as a percentage of the maximum queue depth, *MaxQDepth*. It must be greater than or equal to 0 and less than or equal to 100.

### **QDepthLowEvent (MQCFIN)**

Controls whether Queue Depth Low events are generated (parameter identifier: `MQIA_Q_DEPTH_LOW_EVENT`).

A Queue Depth Low event indicates that an application retrieved a message from a queue. This event caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the **QDepthLowLimit** parameter.

**Note:** The value of this attribute can change implicitly. See [“Definitions of the Programmable Command Formats”](#) on page 983.

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**QDepthLowLimit (MQCFIN)**

Low limit for queue depth (parameter identifier: MQIA\_Q\_DEPTH\_LOW\_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

This event indicates that an application retrieved a message from a queue. This event caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the **QDepthLowEvent** parameter.

Specify the value as a percentage of the maximum queue depth (**MaxQDepth** attribute), in the range 0 through 100.

**QDepthMaxEvent (MQCFIN)**

Controls whether Queue Full events are generated (parameter identifier: MQIA\_Q\_DEPTH\_MAX\_EVENT).

A Queue Full event indicates that an MQPUT call to a queue was rejected because the queue is full. That is, the queue depth reached its maximum value.

**Note:** The value of this attribute can change implicitly; see [“Definitions of the Programmable Command Formats”](#) on page 983.

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**QDesc (MQCFST)**

Queue description (parameter identifier: MQCA\_Q\_DESC).

Text that briefly describes the object.

The maximum length of the string is MQ\_Q\_DESC\_LENGTH.

Use characters from the character set identified by the coded character set identifier (CCSID) for the message queue manager on which the command is executing. This choice ensures that the text is translated correctly if it is sent to another queue manager.

**QServiceInterval (MQCFIN)**

Target for queue service interval (parameter identifier: MQIA\_Q\_SERVICE\_INTERVAL).

The service interval used for comparison to generate Queue Service Interval High and Queue Service Interval OK events. See the *QServiceIntervalEvent* parameter.

Specify a value in the range 0 through 999 999 999 milliseconds.

**QServiceIntervalEvent (MQCFIN)**

Controls whether Service Interval High or Service Interval OK events are generated (parameter identifier: MQIA\_Q\_SERVICE\_INTERVAL\_EVENT).

A Queue Service Interval High event is generated when a check indicates that no messages were retrieved from, or put to, the queue for at least the time indicated by the **QServiceInterval** attribute.

A Queue Service Interval OK event is generated when a check indicates that a message was retrieved from the queue within the time indicated by the **QServiceInterval** attribute.

**Note:** The value of this attribute can change implicitly; see “Definitions of the Programmable Command Formats” on page 983.

The value can be any of the following values:

**MQQSIE\_HIGH**

Queue Service Interval High events enabled.

- Queue Service Interval High events are enabled and
- Queue Service Interval OK events are disabled.

**MQQSIE\_OK**

Queue Service Interval OK events enabled.

- Queue Service Interval High events are disabled and
- Queue Service Interval OK events are enabled.

**MQQSIE\_NONE**

No queue service interval events enabled.

- Queue Service Interval High events are disabled and
- Queue Service Interval OK events are also disabled.

**z/OS QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP ). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

<i>Table 201. QSGDisposition: Where objects are defined and how they behave</i>		
<b>QSGDisposition</b>	<b>Change</b>	<b>Copy, Create</b>
MQQSGD_COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command. It uses the MQQSGD_GROUP object of the same name as the <i>ToQName</i> object (for Copy) or the <i>QName</i> object (for Create). For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.

Table 201. QSGDisposition: Where objects are defined and how they behave (continued)

QSGDisposition	Change	Copy, Create
MQQSGD_GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE QUEUE(q-name) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This value is allowed only in a shared queue manager environment.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers to attempt to make or refresh local copies on page set zero:</p> <pre>DEFINE QUEUE(q-name) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
MQQSGD_PRIVATE	<p>The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.</p>	<p>Not permitted.</p>
MQQSGD_Q_MGR	<p>The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This value is the default value.</p>	<p>The object is defined on the page set of the queue manager that executes the command. This value is the default value. For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.</p>
MQQSGD_SHARED	<p>This value applies only to local queues. The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD_SHARED. Any object residing on the page set of the queue manager that executes the command, or any object defined by a command using the parameter MQQSGD_GROUP, is not affected by this command.</p>	<p>This option applies only to local queues. The object is defined in the shared repository. Messages are stored in the coupling facility and are available to any queue manager in the queue sharing group. You can specify MQQSGD_SHARED only if:</p> <ul style="list-style-type: none"> <li>• <i>CFStructure</i> is nonblank</li> <li>• <i>IndexType</i> is not MQIT_MSG_TOKEN</li> <li>• The queue is not one of the following: <ul style="list-style-type: none"> <li>– SYSTEM.CHANNEL.INITQ</li> <li>– SYSTEM.COMMAND.INPUT</li> </ul> </li> </ul>

### QueueAccounting (MQCFIN)

Controls the collection of accounting data (parameter identifier: MQIA\_ACCOUNTING\_Q).

The value can be:

**MQMON\_Q\_MGR**

The collection of accounting data for the queue is performed based upon the setting of the **QueueAccounting** parameter on the queue manager.

**MQMON\_OFF**

Accounting data collection is disabled for the queue.

**MQMON\_ON**

If the value of the queue manager's *QueueAccounting* parameter is not MQMON\_NONE, accounting data collection is enabled for the queue.


**QueueMonitoring (MQCFIN)**

Online monitoring data collection (parameter identifier: MQIA\_MONITORING\_Q).

Specifies whether online monitoring data is to be collected and, if so, the rate at which the data is collected. The value can be any of the following values:

**MQMON\_OFF**

Online monitoring data collection is turned off for this queue.

 This is the default value on z/OS.

**MQMON\_Q\_MGR**

The value of the queue manager's **QueueMonitoring** parameter is inherited by the queue.

**MQMON\_LOW**

The rate of data collection is low for this queue.

 If the value of the queue manager **QueueMonitoring** parameter is not MQMON\_NONE, online monitoring data collection is turned on.

**MQMON\_MEDIUM**

The rate of data collection is moderate for this queue.

 If the value of the queue manager **QueueMonitoring** parameter is not MQMON\_NONE, online monitoring data collection is turned on.

**MQMON\_HIGH**

The rate of data collection is high for this queue.

 If the value of the queue manager **QueueMonitoring** parameter is not MQMON\_NONE, online monitoring data collection is turned on.

**QueueStatistics (MQCFIN)**

Statistics data collection (parameter identifier: MQIA\_STATISTICS\_Q).

Specifies whether statistics data collection is enabled. The value can be any of the following values:


**MQMON\_Q\_MGR**



The value of the queue manager's **QueueStatistics** parameter is inherited by the queue.

**MQMON\_OFF**

Statistics data collection is disabled

**MQMON\_ON**

 If the value of the queue manager's *QueueStatistics* parameter is not MQMON\_NONE, statistics data collection is enabled

  On z/OS systems, you must enable class 5 statistics using the START TRACE command.

**RemoteQMGrName (MQCFST)**

Name of remote queue manager (parameter identifier: MQCA\_REMOTE\_Q\_MGR\_NAME).

If an application opens the local definition of a remote queue, *RemoteQMGrName* must not be blank or the name of the queue manager the application is connected to. If *XmitQName* is blank there must be a local queue called *RemoteQMGrName*. That queue is used as the transmission queue.

If this definition is used for a queue manager alias, *RemoteQMgrName* is the name of the queue manager. The queue manager name can be the name of the connected queue manager. If *XmitQName* is blank, when the queue is opened there must be a local queue called *RemoteQMgrName*. That queue is used as the transmission queue.

If this definition is used for a reply-to queue alias, *RemoteQMgrName* is the name of the queue manager that is to be the reply-to queue manager.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

### **RemoteQName (MQCFST)**

Name of remote queue as known locally on the remote queue manager (parameter identifier: MQCA\_REMOTE\_Q\_NAME).

If this definition is used for a local definition of a remote queue, *RemoteQName* must not be blank when the open occurs.

If this definition is used for a queue manager alias definition, *RemoteQName* must be blank when the open occurs.

If this definition is used for a reply-to queue alias, this name is the name of the queue that is to be the reply-to queue.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **Replace (MQCFIN)**

Replace attributes (parameter identifier: MQIACF\_REPLACE). This parameter is not valid on a Change Queue command.

If the object exists, the effect is like issuing the Change Queue command. It is like a Change Queue command without the MQFC\_YES option on the **Force** parameter, and with all of the other attributes specified. In particular, note that any messages which are on the existing queue are retained.

The Change Queue command without MQFC\_YES on the **Force** parameter, and the Create Queue command with MQRP\_YES on the **Replace** parameter, are different. The difference is that the Change Queue command does not change unspecified attributes. Create Queue with MQRP\_YES sets all the attributes. If you use MQRP\_YES, unspecified attributes are taken from the default definition, and the attributes of the object being replaced, if one exists, are ignored.)

The command fails if both of the following statements are true:

- The command sets attributes that would require the use of MQFC\_YES on the **Force** parameter if you were using the Change Queue command.
- The object is open.

The Change Queue command with MQFC\_YES on the **Force** parameter succeeds in this situation.

If MQSCO\_CELL is specified on the **Scope** parameter on AIX and Linux, and there is already a queue with the same name in the cell directory, the command fails. The command fails even if MQRP\_YES is specified.

The value can be any of the following values:

#### **MQRP\_YES**

Replace existing definition.

#### **MQRP\_NO**

Do not replace existing definition.

### **RetentionInterval (MQCFIN)**

Retention interval (parameter identifier: MQIA\_RETENTION\_INTERVAL).

The number of hours for which the queue might be needed, based on the date and time when the queue was created.

This information is available to a housekeeping application or an operator and can be used to determine when a queue is no longer required. The queue manager does not delete queues nor does



it prevent queues from being deleted if their retention interval is not expired. It is the responsibility of the user to take any required action.

Specify a value in the range 0 - 999,999,999.

### Scope (MQCFIN)

Scope of the queue definition (parameter identifier: MQIA\_SCOPE).

Specifies whether the scope of the queue definition extends beyond the queue manager which owns the queue. It does so if the queue name is contained in a cell directory, so that it is known to all the queue managers within the cell.

If this attribute is changed from MQSCO\_CELL to MQSCO\_Q\_MGR, the entry for the queue is deleted from the cell directory.

Model and dynamic queues cannot be changed to have cell scope.

If it is changed from MQSCO\_Q\_MGR to MQSCO\_CELL, an entry for the queue is created in the cell directory. If there is already a queue with the same name in the cell directory, the command fails. The command also fails if no name service supporting a cell directory is configured.

The value can be:

#### **MQSCO\_Q\_MGR**

Queue manager scope.

#### **MQSCO\_CELL**

Cell scope.

This value is not supported on IBM i.

This parameter is not available on z/OS.

### Shareability (MQCFIN)

The queue can be shared, or not (parameter identifier: MQIA\_SHAREABILITY).

Specifies whether multiple instances of applications can open this queue for input.

The value can be any of the following values:

#### **MQQA\_SHAREABLE**

Queue is shareable.

#### **MQQA\_NOT\_SHAREABLE**

Queue is not shareable.

### **StorageClass (MQCFST)**


Storage class (parameter identifier: MQCA\_STORAGE\_CLASS). This parameter applies to z/OS only.

Specifies the name of the storage class.

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.

### **StreamQ (MQCFST)**

Name of the streaming queue (parameter identifier: MQCA\_STREAM\_QUEUE\_NAME)

**Note:**  If the user setting the **Streamq** attribute does not have change authority on the chosen stream queue, the command fails with error message AMQ8135E Not Authorized, or the equivalent message CSQ9016E on z/OS.

In addition, if the stream queue does not exist, error message AMQ8135E is returned instead of AMQ8147E IBM MQ object not found or message CSQM125I on z/OS.

### **StreamQService (MQCFIN)**

Quality of service used when delivering messages to **Streamq** (parameter identifier: MQIA\_STREAM\_QUEUE\_QOS)

The value can be:

**MQST\_BEST\_EFFORT**

If the original message can be delivered, but the streamed message cannot, the original message is still delivered to its queue.

This is the default value.

**MQST\_MUST\_DUP**

The queue manager ensures that both the original message and the streamed message are successfully delivered to their queues.

If, for some reason, the streamed message cannot be delivered to its queue, the original message is not delivered to its queue either.

**TargetType (MQCFIN)**

Target type (parameter identifier: MQIA\_BASE\_TYPE).

Specifies the type of object to which the alias resolves.

The value can be any of the following values:

**MQOT\_Q**

The object is a queue.

**MQOT\_TOPIC**

The object is a topic.

**TriggerControl (MQCFIN)**

Trigger control (parameter identifier: MQIA\_TRIGGER\_CONTROL).

Specifies whether trigger messages are written to the initiation queue.

The value can be:

**MQTC\_OFF**

Trigger messages not required.

**MQTC\_ON**

Trigger messages required.

**TriggerData (MQCFST)**

Trigger data (parameter identifier: MQCA\_TRIGGER\_DATA).

Specifies user data that the queue manager includes in the trigger message. This data is made available to the monitoring application that processes the initiation queue and to the application that is started by the monitor.

The maximum length of the string is MQ\_TRIGGER\_DATA\_LENGTH.

**TriggerDepth (MQCFIN)**

Trigger depth (parameter identifier: MQIA\_TRIGGER\_DEPTH).

Specifies (when *TriggerType* is MQTT\_DEPTH) the number of messages that initiates a trigger message to the initiation queue. The value must be in the range 1 through 999 999 999.

**TriggerMsgPriority (MQCFIN)**

Threshold message priority for triggers (parameter identifier: MQIA\_TRIGGER\_MSG\_PRIORITY).

Specifies the minimum priority that a message must have before it can cause, or be counted for, a trigger event. The value must be in the range of priority values that is supported (0 through 9).

**TriggerType (MQCFIN)**

Trigger type (parameter identifier: MQIA\_TRIGGER\_TYPE).

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

The value can be any of the following values:

**MQTT\_NONE**

No trigger messages.

**MQTT\_EVERY**

Trigger message for every message.

**MQTT\_FIRST**

Trigger message when queue depth goes from 0 to 1.

**MQTT\_DEPTH**

Trigger message when depth threshold exceeded.

**Usage (MQCFIN)**

Usage (parameter identifier: MQIA\_USAGE).

Specifies whether the queue is for normal usage or for transmitting messages to a remote message queue manager.

The value can be any of the following values:

**MQUS\_NORMAL**

Normal usage.

**MQUS\_TRANSMISSION**

Transmission queue.

**XmitQName (MQCFST)**

Transmission queue name (parameter identifier: MQCA\_XMIT\_Q\_NAME).

Specifies the local name of the transmission queue to be used for messages destined for either a remote queue or for a queue manager alias definition.

If *XmitQName* is blank, a queue with the same name as *RemoteQMgrName* is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue manager alias and *RemoteQMgrName* is the name of the connected queue manager.

It is also ignored if the definition is used as a reply-to queue alias definition.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**Error codes (Change, Copy, and Create Queue)**

This command might return the following errors in the response format header, in addition to the values shown on in [“Error codes applicable to all commands”](#) on page 990.

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CELL\_DIR\_NOT\_AVAILABLE**

Cell directory is not available.

**MQRCCF\_CLUSTER\_NAME\_CONFLICT**

Cluster name conflict.

**MQRCCF\_CLUSTER\_Q\_USAGE\_ERROR**

Cluster usage conflict.

**MQRCCF\_DYNAMIC\_Q\_SCOPE\_ERROR**

Dynamic queue scope error.

**MQRCCF\_FORCE\_VALUE\_ERROR**

Force value not valid.

**MQRCCF\_Q\_ALREADY\_IN\_CELL**

Queue exists in cell.

**MQRCCF\_Q\_TYPE\_ERROR**

Queue type not valid.

**Multi** **V 9.3.0** **MQRCCF\_STREAMQ\_CONFLICT**

This queue has one or more attributes which are incompatible with having the STREAMQ attribute set on it.

**Multi** **V 9.3.0** **MQRCCF\_STREAMQ\_DEST\_CONFLICT**

The STREAMQ attribute refers to a queue that has one or more attributes which are incompatible with being a streaming queue.

**Multi** **V 9.3.0** **MQRCCF\_STREAMQ\_DEST\_NOT\_SUPP**

The STREAMQ attribute refers to a queue that cannot be used as a streaming queue.

**Multi** **V 9.3.0** **MQRCCF\_STREAMQ\_NOT\_SUPPORTED**

This queue cannot have the STREAMQ attribute set on it.

**Multi** **Change, Copy, and Create Service on Multiplatforms**

The Change Service PCF command changes existing service definitions. The Copy and Create service commands create new service definitions - the Copy command uses attribute values of an existing service definition.

The Change Service (MQCMD\_CHANGE\_SERVICE) command changes the specified attributes of an existing IBM MQ service definition. For any optional parameters that are omitted, the value does not change.

The Copy Service (MQCMD\_COPY\_SERVICE) command creates an IBM MQ service definition, using, for attributes not specified in the command, the attribute values of an existing service definition.

The Create Service (MQCMD\_CREATE\_SERVICE) command creates an IBM MQ service definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

## Required parameter (Change and Create Service)

### ServiceName (MQCFST)

The name of the service definition to be changed or created (parameter identifier: MQCA\_SERVICE\_NAME).

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

## Required parameters (Copy Service)

### FromServiceName (MQCFST)

The name of the service definition to be copied from (parameter identifier: MQCACF\_FROM\_SERVICE\_NAME).

This parameter specifies the name of the existing service definition that contains values for the attributes not specified in this command.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

### ToServiceName (MQCFST)

To service name (parameter identifier: MQCACF\_TO\_SERVICE\_NAME).

This parameter specifies the name of the new service definition. If a service definition with this name exists, *Replace* must be specified as MQRP\_YES.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

## Optional parameters (Change, Copy, and Create Service)

### Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF\_REPLACE).

If a namelist definition with the same name as *ToServiceName* exists, this specifies parameter whether it is to be replaced. The value can be:

**MQRP\_YES**

Replace existing definition.

**MQRP\_NO**

Do not replace existing definition.

**ServiceDesc (MQCFST)**

Description of service definition (parameter identifier: MQCA\_SERVICE\_DESC).

This parameter is a plain-text comment that provides descriptive information about the service definition. It must contain only displayable characters.

If characters are used that are not in the coded character set identifier (CCSID) for the queue manager on which the command is executing, they might be translated incorrectly.

The maximum length of the string is MQ\_SERVICE\_DESC\_LENGTH.

**ServiceType (MQCFIN)**

The mode in which the service is to run (parameter identifier: MQIA\_SERVICE\_TYPE).

Specify either:

**MQSVC\_TYPE\_SERVER**

Only one instance of the service can be executed at a time, with the status of the service made available by the Inquire Service Status command.

**MQSVC\_TYPE\_COMMAND**

Multiple instances of the service can be started.

**StartArguments (MQCFST)**

Arguments to be passed to the program on startup (parameter identifier: MQCA\_SERVICE\_START\_ARGS).

Specify each argument within the string as you would on a command line, with a space to separate each argument to the program.

The maximum length of the string is MQ\_SERVICE\_ARGS\_LENGTH.

**StartCommand (MQCFST)**

Service program name (parameter identifier: MQCA\_SERVICE\_START\_COMMAND).

Specifies the name of the program which is to run. You must specify a fully qualified path name to the executable program.

The maximum length of the string is MQ\_SERVICE\_COMMAND\_LENGTH.

**StartMode (MQCFIN)**

Service mode (parameter identifier: MQIA\_SERVICE\_CONTROL).

Specifies how the service is to be started and stopped. The value can be any of the following values:

**MQSVC\_CONTROL\_MANUAL**

The service is not to be started automatically or stopped automatically. It is to be controlled by user command. This value is the default value.

**MQSVC\_CONTROL\_Q\_MGR**

The service being defined is to be started and stopped at the same time as the queue manager is started and stopped.

**MQSVC\_CONTROL\_Q\_MGR\_START**

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**StderrDestination (MQCFST)**

Specifies the path to a file to which the standard error (stderr) of the service program must be redirected (parameter identifier: MQCA\_STDERR\_DESTINATION).

If the file does not exist when the service program is started, the file is created.

The maximum length of the string is MQ\_SERVICE\_PATH\_LENGTH.

**StdoutDestination (MQCFST)**

Specifies the path to a file to which the standard output (stdout) of the service program must be redirected (parameter identifier: MQCA\_STDOUT\_DESTINATION).

If the file does not exist when the service program is started, the file is created.

The maximum length of the string is MQ\_SERVICE\_PATH\_LENGTH.

**StopArguments (MQCFST)**

Specifies the arguments to be passed to the stop program when instructed to stop the service (parameter identifier: MQCA\_SERVICE\_STOP\_ARGS).

Specify each argument within the string as you would on a command line, with a space to separate each argument to the program.

The maximum length of the string is MQ\_SERVICE\_ARGS\_LENGTH.

**StopCommand (MQCFST)**

Service program stop command (parameter identifier: MQCA\_SERVICE\_STOP\_COMMAND).

This parameter is the name of the program that is to run when the service is requested to stop. You must specify a fully qualified path name to the executable program.

The maximum length of the string is MQ\_SERVICE\_COMMAND\_LENGTH.

 **Change, Copy, and Create Storage Class on z/OS**

The Change Storage Class PCF command changes existing storage class definitions. The Copy and Create Storage Class commands create new storage class definitions - the Copy command uses attribute values of an existing storage class definition.

The Change Storage Class (MQCMD\_CHANGE\_STG\_CLASS) command changes the characteristics of a storage class. For any optional parameters that are omitted, the value does not change.

The Copy Storage Class (MQCMD\_COPY\_STG\_CLASS) command creates a storage class to page set mapping using, for attributes not specified in the command, the attribute values of an existing storage class.

The Create Storage Class (MQCMD\_CREATE\_STG\_CLASS) command creates a storage class to page set mapping. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

**Required parameter (Change and Create Storage Class)****StorageClassName (MQCFST)**

The name of the storage class to be changed or created (parameter identifier: MQCA\_STORAGE\_CLASS).

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.

**Required parameters (Copy Storage Class)****FromStorageClassName (MQCFST)**

The name of the storage class to be copied from (parameter identifier: MQCACF\_FROM\_STORAGE\_CLASS).

On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD\_Q\_MGR or MQQSGD\_COPY to copy from. This parameter is ignored if a value of MQQSGD\_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToStorageClassName* and the disposition MQQSGD\_GROUP is searched for to copy from.

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.

**ToStorageClassName (MQCFST)**

The name of the storage class to copy to (parameter identifier: MQCACF\_TO\_STORAGE\_CLASS).

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.

## Optional parameters (Change, Copy, and Create Storage Class)

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### PageSetId (MQCFIN)

Page set identifier that the storage class is to be associated with (parameter identifier: MQIA\_PAGESET\_ID).

Specify a string of two numeric characters in the range 00 through 99.

If you do not specify this parameter, the default is taken from the default storage class SYSTEMST.

No check is made that the page set has been defined; an error is raised only if you try to put a message to a queue that specifies this storage class (MQRC\_PAGESET\_ERROR).

### PassTicketApplication (MQCFST)

Pass ticket application (parameter identifier: MQCA\_PASS\_TICKET\_APPL).

The application name that is passed to RACF when authenticating the passticket specified in the MQIIH header.

The maximum length is MQ\_PASS\_TICKET\_APPL\_LENGTH.

### QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

Table 202. QSGDisposition: Where objects are defined and how they behave		
QSGDisposition	Change	Copy, Create
<b>MQQSGD_COPY</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command. It uses the MQQSGD_GROUP object of the same name as the <i>ToStorageClassName</i> object (for Copy) or the <i>StorageClassName</i> object (for Create).

Table 202. QSGDisposition: Where objects are defined and how they behave (continued)

QSGDisposition	Change	Copy, Create
<b>MQQSGD_GROUP</b>	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE STGCLASS(storage-class) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This parameter is allowed only if the queue manager is in a queue sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to attempt to make or refresh local copies on page set zero:</p> <pre>DEFINE STGCLASS(storage-class) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
<b>MQQSGD_PRIVATE</b>	<p>The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.</p>	Not permitted.
<b>MQQSGD_Q_MGR</b>	<p>The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This value is the default value.</p>	<p>The object is defined on the page set of the queue manager that executes the command. This value is the default value.</p>

**Replace (MQCFIN)**

Replace attributes (parameter identifier: MQIACF\_REPLACE).

If a storage class definition with the same name as *ToStorageClassName* exists, this parameter specifies whether it is to be replaced. The value can be:

**MQRP\_YES**

Replace existing definition.

**MQRP\_NO**

Do not replace existing definition.

**StorageClassDesc (MQCFST)**

The description of the storage class (parameter identifier: MQCA\_STORAGE\_CLASS\_DESC).

The maximum length is MQ\_STORAGE\_CLASS\_DESC\_LENGTH.

**XCFGroupName (MQCFST)**

XCF group name (parameter identifier: MQCA\_XCF\_GROUP\_NAME).



If you are using the IMS bridge, this parameter is the name of the XCF group to which the IMS system belongs.

The maximum length is MQ\_XCF\_GROUP\_NAME\_LENGTH.

#### **XCFMemberName (MQCFST)**

XCF member name (parameter identifier: MQCA\_XCF\_MEMBER\_NAME).

If you are using the IMS bridge, this parameter is the XCF member name of the IMS system within the XCF group specified in *XCFGroupName*.

The maximum length is MQ\_XCF\_MEMBER\_NAME\_LENGTH.

### **Change, Copy, and Create Subscription**

The Change Subscription PCF command changes existing subscription definitions. The Copy and Create Subscription commands create new subscription definitions - the Copy command uses attribute values of an existing subscription definition.

The Change Subscription (MQCMD\_CHANGE\_SUBSCRIPTION) command changes the specified attributes of an existing IBM MQ subscription. For any optional parameters that are omitted, the value does not change.

The Copy Subscription (MQCMD\_COPY\_SUBSCRIPTION) command creates an IBM MQ subscription, using, for attributes not specified in the command, the attribute values of an existing subscription.

The Create Subscription (MQCMD\_CREATE\_SUBSCRIPTION) command creates an IBM MQ administrative subscription so that existing applications can participate in publish/subscribe application.

### **Required parameters (Change Subscription)**

#### **SubName (MQCFST)**

The name of the subscription definition to be changed (parameter identifier: MQCACF\_SUB\_NAME).

The maximum length of the string is MQ\_SUB\_NAME\_LENGTH.

or

#### **SubId (MQCFBS)**

The unique identifier of the subscription definition to be changed (parameter identifier: MQBACF\_SUB\_ID).

The maximum length of the string is MQ\_CORREL\_ID\_LENGTH.

### **Required parameters (Copy Subscription)**

#### **ToSubscriptionName (MQCFBS)**


The name of the subscription to copy to (parameter identifier: MQCACF\_TO\_SUB\_NAME).

The maximum length of the string is MQ\_SUB\_NAME\_LENGTH.

You require at least one of *FromSubscriptionName* or *SubId*.

#### **FromSubscriptionName (MQCFST)**

The name of the subscription definition to be copied from (parameter identifier: MQCACF\_FROM\_SUB\_NAME).

 On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD\_Q\_MGR or MQQSGD\_COPY to copy from. This parameter is ignored if a value of MQQSGD\_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToSubscriptionName* and the disposition MQQSGD\_GROUP is used.

The maximum length of the string is MQ\_SUB\_NAME\_LENGTH.

#### **SubId (MQCFBS)**

The unique identifier of the subscription definition to be changed (parameter identifier: MQBACF\_SUB\_ID).

The maximum length of the string is MQ\_CORREL\_ID\_LENGTH.

## Required parameters (Create Subscription)

You must provide the *SubName*.

### SubName (MQCFST)

The name of the subscription definition to be changed (parameter identifier: MQCACF\_SUB\_NAME).

The maximum length of the string is MQ\_SUB\_NAME\_LENGTH.

You require at least one of *TopicObject* or *TopicString*.

### TopicObject (MQCFST)

The name of a previously defined topic object from which is obtained the topic name for the subscription (parameter identifier: MQCA\_TOPIC\_NAME). Although the parameter is accepted, the value specified cannot be different from the original value for Change Subscription.

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

### TopicString (MQCFST)

The resolved topic string (parameter identifier: MQCA\_TOPIC\_STRING).

The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

## Optional parameters (Change, Copy, and Create Subscription)



### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### Destination (MQCFST)

Destination (parameter identifier: MQCACF\_DESTINATION).

Specifies the name of the alias, local, remote, or cluster queue to which messages for this subscription are put.

This parameter is mandatory if *DestinationClass* is set to MQDC\_PROVIDED, but is not applicable if *DestinationClass* is set to MQDC\_MANAGED.

### DestinationClass (MQCFIN)

Destination class (parameter identifier: MQIACF\_DESTINATION\_CLASS).

Specifies whether the destination is managed.

Specify either:

#### MQDC\_MANAGED

The destination is managed.

**MQDC\_PROVIDED**

The destination queue is as specified in the *Destination* field.

Although the parameter is accepted, the value specified cannot be different from the original value for Change Subscription.

**DestinationCorrelId (MQCFBS)**

Destination correlation identifier (parameter identifier: MQBACF\_DESTINATION\_CORREL\_ID).

Provides a correlation identifier that is placed in the *CorrelId* field of the message descriptor for all the messages sent to this subscription.

The maximum length is MQ\_CORREL\_ID\_LENGTH.

**DestinationQueueManager (MQCFST)**

Destination queue manager (parameter identifier: MQCACF\_DESTINATION\_Q\_MGR).

Specifies the name of the destination queue manager, either local or remote, to which messages for the subscription are forwarded.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**Expiry (MQCFIN)**

The time, in tenths of a second, at which a subscription expires after its creation date and time (parameter identifier: MQIACF\_EXPIRY).

The default value of MQEI\_UNLIMITED means that the subscription never expires.

After a subscription has expired it becomes eligible to be discarded by the queue manager and receives no further publications.

**PublishedAccountingToken (MQCFBS)**

Value of the accounting token used in the *AccountingToken* field of the message descriptor (parameter identifier: MQBACF\_ACCOUNTING\_TOKEN).

The maximum length of the string is MQ\_ACCOUNTING\_TOKEN\_LENGTH.

**PublishedApplicationIdentifier (MQCFST)**

Value of the application identity data used in the *AppIdentityData* field of the message descriptor (parameter identifier: MQCACF\_APPL\_IDENTITY\_DATA).

The maximum length of the string is MQ\_APPL\_IDENTITY\_DATA\_LENGTH.

**PublishPriority (MQCFIN)**

The priority of the message sent to this subscription (parameter identifier: MQIACF\_PUB\_PRIORITY).

The value can be:

**MQPRI\_PRIORITY\_AS\_PUBLISHED**

Priority of messages sent to this subscription is taken from the priority supplied to the published message. This value is the supplied default value.

**MQPRI\_PRIORITY\_AS\_QDEF**

Priority of messages sent to this subscription is determined by the default priority of the queue defined as a destination.

**0-9**

An integer value providing an explicit priority for messages sent to this subscription.

**PublishSubscribeProperties (MQCFIN)**

Specifies how publish/subscribe related message properties are added to messages sent to this subscription (parameter identifier: MQIACF\_PUBSUB\_PROPERTIES).

The value can be:

**MQPSPROP\_COMPAT**

If the original publication is a PCF message, then the publish/subscribe properties are added as PCF attributes. Otherwise, publish/subscribe properties are added within an MQRFH version 1

header. This method is compatible with applications coded for use with previous versions of IBM MQ.

**MQPSPROP\_NONE**

Do not add publish/subscribe properties to the messages. This value is the supplied default value.

**MQPSPROP\_RFH2**

Publish/subscribe properties are added within an MQRFH version 2 header. This method is compatible with applications coded for use with IBM Integration Bus, formerly known as WebSphere Message Broker.

**Selector (MQCFST)**

Specifies the selector applied to messages published to the topic (parameter identifier: MQCACF\_SUB\_SELECTOR). Although the parameter is accepted, the value specified cannot be different from the original value for Change Subscription.

Only those messages that satisfy the selection criteria are put to the destination specified by this subscription.

The maximum length of the string is MQ\_SELECTOR\_LENGTH.

**SubscriptionLevel (MQCFIN)**

The level within the subscription interception hierarchy at which this subscription is made (parameter identifier: MQIACF\_SUB\_LEVEL). To ensure that an intercepting application receives messages before any other subscribers, make sure that it has the highest subscription level of all subscribers. Although the parameter is accepted, the value specified cannot be different from the original value for Change Subscription.

The value can be:

**0 - 9**

An integer in the range 0-9. The default value is 1. Subscribers with a subscription level of 9 intercept publications before they reach subscribers with lower subscription levels.

**SubscriptionScope (MQCFIN)**

Determines whether this subscription is passed to other queue managers in the network (parameter identifier: MQIACF\_SUBSCRIPTION\_SCOPE). Although the parameter is accepted, the value specified cannot be different from the original value for Change Subscription.

The value can be:

**MQTSCOPE\_ALL**

The subscription is forwarded to all queue managers directly connected through a publish/subscribe collective or hierarchy. This value is the supplied default value.

**MQTSCOPE\_QMGR**

The subscription only forwards messages published on the topic within this queue manager.

**SubscriptionUser (MQCFST)**

The userid that 'owns' this subscription. This parameter is either the userid associated with the creator of the subscription, or, if subscription takeover is permitted, the userid which last took over the subscription. (parameter identifier: MQCACF\_SUB\_USER\_ID).

The maximum length of the string is MQ\_USER\_ID\_LENGTH.

**TopicString (MQCFST)**

The resolved topic string (parameter identifier: MQCA\_TOPIC\_STRING). Although the parameter is accepted, the value specified cannot be different from the original value for Change Subscription.

The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

**Userdata (MQCFST)**

User data (parameter identifier: MQCACF\_SUB\_USER\_DATA).

Specifies the user data associated with the subscription

The maximum length of the string is MQ\_USER\_DATA\_LENGTH.

### **VariableUser (MQCFST)**

Specifies whether a user other than the one who created the subscription, that is, the user shown in *SubscriptionUser* can take over the ownership of the subscription (parameter identifier: MQIACF\_VARIABLE\_USER\_ID).

The value can be:

#### **MQVU\_ANY\_USER**

Any user can take over the ownership. This value is the supplied default value.

#### **MQVU\_FIXED\_USER**

No other user can take over the ownership.

### **WildcardSchema (MQCFIN)**

Specifies the schema to be used when interpreting any wildcard characters contained in the *TopicString* (parameter identifier: MQIACF\_WILDCARD\_SCHEMA). Although the parameter is accepted, the value specified cannot be different from the original value for Change Subscription.

The value can be:

#### **MQWS\_CHAR**

Wildcard characters represent portions of strings for compatibility with IBM MQ V6.0 broker.

#### **MQWS\_TOPIC**

Wildcard characters represent portions of the topic hierarchy for compatibility with IBM Integration Bus. This value is the supplied default value.

## ***Change, Copy, and Create Topic***

The Change Topic PCF command changes existing topic definitions. The Copy and Create Topic commands create new topic definitions - the Copy command uses attribute values of an existing topic definition.

The Change Topic (MQCMD\_CHANGE\_TOPIC) command changes the specified attributes of an existing IBM MQ administrative topic definition. For any optional parameters that are omitted, the value does not change.

The Copy Topic (MQCMD\_COPY\_TOPIC) command creates an IBM MQ administrative topic definition by using, for attributes not specified in the command, the attribute values of an existing topic definition.

The Create Topic (MQCMD\_CREATE\_TOPIC) command creates an IBM MQ administrative topic definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

## **Required parameter (Change Topic)**

### **TopicName (MQCFST)**


The name of the administrative topic definition to be changed (parameter identifier: MQCA\_TOPIC\_NAME).

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

## **Required parameters (Copy Topic)**

### **FromTopicName (MQCFST)**

The name of the administrative topic object definition to be copied from (parameter identifier: MQCACF\_FROM\_TOPIC\_NAME).

 On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD\_Q\_MGR or MQQSGD\_COPY to copy from. This parameter is ignored if a value of MQQSGD\_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToTopicName* and the disposition MQQSGD\_GROUP is searched for to copy from.

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

### TopicString (MQCFST)

The topic string (parameter identifier: MQCA\_TOPIC\_STRING). This string uses the forward slash (/) character as a delimiter for elements within the topic tree.

The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

### ToTopicName (MQCFST)

The name of the administrative topic definition to copy to (parameter identifier: MQCACF\_TO\_TOPIC\_NAME).

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

## Required parameters (Create Topic)

### TopicName (MQCFST)

The name of the administrative topic definition to be created (parameter identifier: MQCA\_TOPIC\_NAME).

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

### TopicString (MQCFST)

The topic string (parameter identifier: MQCA\_TOPIC\_STRING).

This parameter is required and cannot contain the empty string. The "/" character within this string has a special meaning. It delimits the elements in the topic tree. A topic string can start with the "/" character but is not required to. A string starting with the "/" character is not the same as a string that does not start with the "/" character. A topic string cannot end with the "/" character.

The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

## Optional parameters (Change, Copy, and Create Topic)

### V 9.3.1 Multi CapExpiry (MQCFIN)

Capped expiry processing (parameter identifier MQIA\_CAP\_EXPIRY) which can be an integer value or take values of MQCEX\_NOLIMIT or MQCEX\_AS\_PARENT.

Specifies a lifetime limit for messages put using the object, expressed in 10ths of a second. A value of -1, displayed as NOLIMIT, has no effect on processing.

Note that in addition to the CapExpiry attribute itself, you can use an attribute of the **CUSTOM** parameter, which is a string. Therefore passing a string PCF parameter MQCA\_CUSTOM, has a string value of CAEXPRY (integer).

CapExpiry provides, or limits, the value in the MQMD Expiry field of any message put.

An application provided MQMD **Expiry** that is less than any resolved CapExpiry value is passed. This value is not replaced by that resolved CapExpiry value.

This process allows an IBM MQ administrator to limit the life of messages put by an application that overlooked (or was unable to provide, in the case of MQTT) message expiration criteria.

However, this option does not allow an administrator to override application behavior where the required lifetime of messages was under-estimated.

If you specify **CAEXPRY ASPARENT**, the value is established by moving up the tree towards the root, until the point of first resolution to a non *ASPARENT* value. Additionally you can use the **CUSTOM CAEXPRY** option. See [Making CAEXPRY a first-class MQSC attribute in MQ 9.3.1](#) for more information.

As for queues, the smallest **CAEXPRY** found during a put operation, is used. Further limiting can be set for specific subscribers, based on **CAEXPRY** resolution applied on the path to resolve the destination for a subscription.

The new capped value for expiry is used during the put processing as if it had been provided by the application in the MQMD structure.

The *capped* value is evaluated for each put being performed, and so is sensitive to the resolution of the put operation. For example, in a cluster, where the put operation is performed with BIND NOT FIXED, messages might pick up different expiry values depending on the CapExpiry value set for the transmission queue used by the channel.

### **ClusterName (MQCFST)**

The name of the cluster to which this topic belongs. (parameter identifier: MQCA\_CLUSTER\_NAME). The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH. Setting this parameter to a cluster that this queue manager is a member of makes all queue managers in the cluster aware of this topic. Any publication to this topic or a topic string below it put to any queue manager in the cluster is propagated to subscriptions on any other queue manager in the cluster. For more details, see [Distributed publish/subscribe networks](#).

The value can be any of the following values:

#### **Blank**

If no topic object above this topic in the topic tree has set this parameter to a cluster name, then this topic does not belong to a cluster. Publications and subscriptions for this topic are not propagated to publish/subscribe cluster-connected queue managers. If a topic node higher in the topic tree has a cluster name set, publications and subscriptions to this topic are also propagated throughout the cluster.

This value is the default value for this parameter if no value is specified.

#### **String**

The topic belongs to this cluster. It is not recommended that this is set to a different cluster from a topic object above this topic object in the topic tree. Other queue managers in the cluster will honor this object's definition unless a local definition of the same name exists on those queue managers.

Additionally, if PublicationScope or SubscriptionScope are set to MQSCOPE\_ALL, this value is the cluster to be used for the propagation of publications and subscriptions, for this topic, to publish/subscribe cluster-connected queue managers.

### **ClusterPubRoute (MQCFIN)**

The routing behavior of publications between queue managers in a cluster (parameter identifier: MQIA\_CLUSTER\_PUB\_ROUTE).

The value can be any of the following values:

#### **MQCLROUTE\_DIRECT**

When you configure a direct routed clustered topic on a queue manager, all queue managers in the cluster become aware of all other queue managers in the cluster. When performing publish and subscribe operations, each queue manager can connect direct to any other queue manager in the cluster.

#### **MQCLROUTE\_TOPIC\_HOST**

When you use topic host routing, all queue managers in the cluster become aware of the cluster queue managers that host the routed topic definition (that is, the queue managers on which you have defined the topic object). When performing publish and subscribe operations, queue managers in the cluster connect only to these topic host queue managers, and not directly to each other. The topic host queue managers are responsible for routing publications from queue managers on which publications are published to queue managers with matching subscriptions.

After a topic object has been clustered (through setting the **CLUSTER** property) you cannot change the value of the **CLROUTE** property. The object must be un-clustered (**CLUSTER** set to ' ') before you can change the value. Un-clustering a topic converts the topic definition to a local topic, which results in a period during which publications are not delivered to subscriptions on remote queue managers; this should be considered when performing this change. See [The effect of defining a non-cluster topic with the same name as a cluster topic from another queue manager](#). If you try to change the value of the **CLROUTE** property while it is clustered, the system generates an MQRCCF\_CLROUTE\_NOT\_ALTERABLE exception.

See also [Routing for publish/subscribe clusters: Notes on behavior](#) and [Designing publish/subscribe clusters](#).

z/OS

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **CommunicationInformation (MQCFST)**

The Multicast communication information object (parameter identifier: MQCA\_COMM\_INFO\_NAME).

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

### **Custom (MQCFST)**

Custom attribute for new features (parameter identifier: MQCA\_CUSTOM).

This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE). Single quotation marks must be escaped with another single quotation mark.

### **CAPEXPY (integer)**

The maximum time, expressed in tenths of a second, until a message published to a topic which inherits properties from this object, remains in the system until it becomes eligible for expiry processing.

For more information on message expiry processing, see [Enforcing lower expiration times](#).

The value can be one of the following:

#### **integer**

The value must be in the range one through to 999 999 999.

#### **NOLIMIT**

There is no limit on the expiry time of messages put using this object.

#### **ASPARENT**

The maximum message expiry time is based on the setting of the closest parent administrative topic object in the topic tree. This is the default value.

Specifying a value for CAPEXPY that is not valid, does not cause the command to fail. Instead,, the default value is used.

### **DefPersistence (MQCFIN)**

Default persistence (parameter identifier: MQIA\_TOPIC\_DEF\_PERSISTENCE).

Specifies the default for message-persistence of messages published to the topic. Message persistence determines whether messages are preserved across restarts of the queue manager.

The value can be any of the following values:

#### **MQPER\_PERSISTENCE\_AS\_PARENT**

The default persistence is based on the setting of the closest parent administrative topic object in the topic tree.



**MQPER\_PERSISTENT**

Message is persistent.

**MQPER\_NOT\_PERSISTENT**

Message is not persistent.

**DefPriority (MQCFIN)**

Default priority (parameter identifier: MQIA\_DEF\_PRIORITY).

Specifies the default priority of messages published to the topic.

Specify either:

***integer***

The default priority to be used, in the range zero through to the maximum priority value that is supported (9).

**MQPRI\_PRIORITY\_AS\_PARENT**

The default priority is based on the setting of the closest parent administrative topic object in the topic tree.

**DefPutResponse (MQCFIN)**

Default put response (parameter identifier: MQIA\_DEF\_PUT\_RESPONSE\_TYPE).

The value can be:

**MQPRT\_ASYNC\_RESPONSE**

The put operation is issued asynchronously, returning a subset of MQMD fields.

**MQPRT\_RESPONSE\_AS\_PARENT**

The default put response is based on the setting of the closest parent administrative topic object in the topic tree.

**MQPRT\_SYNC\_RESPONSE**

The put operation is issued synchronously, returning a response.

**DurableModelQName (MQCFST)**

Name of the model queue to be used for durable subscriptions (parameter identifier: MQCA\_MODEL\_DURABLE\_Q).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**DurableSubscriptions (MQCFIN)**

Whether applications are permitted to make durable subscriptions (parameter identifier: MQIA\_DURABLE\_SUB).

The value can be:

**MQSUB\_DURABLE\_AS\_PARENT**

Whether durable subscriptions are permitted is based on the setting of the closest parent administrative topic object in the topic tree.

**MQSUB\_DURABLE\_ALLOWED**

Durable subscriptions are permitted.

**MQSUB\_DURABLE\_INHIBITED**

Durable subscriptions are not permitted.

**InhibitPublications (MQCFIN)**

Whether publications are allowed for this topic (parameter identifier: MQIA\_INHIBIT\_PUB).

The value can be:

**MQTA\_PUB\_AS\_PARENT**

Whether messages can be published to this topic is based on the setting of the closest parent administrative topic object in the topic tree.

**MQTA\_PUB\_INHIBITED**

Publications are inhibited for this topic.

**MQTA\_PUB\_ALLOWED**

Publications are allowed for this topic.

**InhibitSubscriptions (MQCFIN)**

Whether subscriptions are allowed for this topic (parameter identifier: MQIA\_INHIBIT\_SUB).

The value can be:

**MQTA\_SUB\_AS\_PARENT**

Whether applications can subscribe to this topic is based on the setting of the closest parent administrative topic object in the topic tree.

**MQTA\_SUB\_INHIBITED**

Subscriptions are inhibited for this topic.

**MQTA\_SUB\_ALLOWED**

Subscriptions are allowed for this topic.

**Multicast (MQCFIN)**

Whether multicast is allowable in the topic tree (parameter identifier: MQIA\_MULTICAST).

The value can be:

**MQMC\_AS\_PARENT**

Whether multicast is allowed on this topic is based on the setting of the closest parent administrative topic object in the topic tree.

**MQMC\_ENABLED**

Multicast is allowed on this topic.

**MQMC\_DISABLED**

Multicast is not allowed on this topic.

**MQMC\_ONLY**

Only subscriptions and publications made using multicast are allowed on this topic.

**NonDurableModelQName (MQCFST)**

Name of the model queue to be used for non-durable subscriptions (parameter identifier: MQCA\_MODEL\_NON\_DURABLE\_Q).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**NonPersistentMsgDelivery (MQCFIN)**

The delivery mechanism for non-persistent messages published to this topic (parameter identifier: MQIA\_NPM\_DELIVERY).

The value can be:

**MQDLV\_AS\_PARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**MQDLV\_ALL**

Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT fails.

**MQDLV\_ALL\_DUR**

Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a non-persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no other subscribers receive the message and the MQPUT fails.

**MQDLV\_ALL\_AVAIL**

Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**PersistentMsgDelivery (MQCFIN)**

The delivery mechanism for persistent messages published to this topic (parameter identifier: MQIA\_PM\_DELIVERY).

The value can be:

**MQDLV\_AS\_PARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**MQDLV\_ALL**

Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT fails.

**MQDLV\_ALL\_DUR**

Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no other subscribers receive the message and the MQPUT fails.

**MQDLV\_ALL\_AVAIL**

Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**ProxySubscriptions (MQCFIN)**

Whether a proxy subscription is to be sent for this topic to directly connected queue managers, even if no local subscriptions exist (parameter identifier: MQIA\_PROXY\_SUB).

The value can be:

**MQTA\_PROXY\_SUB\_FORCE**

A proxy subscription is sent to connected queue managers even if no local subscriptions exist.

**Note:** The proxy subscription is sent when this value is set on Create or Change of the topic.

**MQTA\_PROXY\_SUB\_FIRSTUSE**

For each unique topic string at or below this topic object, a proxy subscription is asynchronously sent to all neighboring queue managers in the following scenarios:

- When a local subscription is created.
- When a proxy subscription is received that must be propagated to further directly connected queue managers.

This value is the default value for this parameter if no value is specified.

**PublicationScope (MQCFIN)**

Whether this queue manager propagates publications for this topic, to queue managers as part of a hierarchy or as part of a publish/subscribe cluster (parameter identifier: MQIA\_PUB\_SCOPE).

The value can be:

**MQSCOPE\_AS\_PARENT**

Whether this queue manager propagates publications, for this topic, to queue managers as part of a hierarchy or as part of a publish/subscribe cluster is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

This value is the default value for this parameter if no value is specified.

**MQSCOPE\_QMGR**

Publications for this topic are not propagated to other queue managers.

**MQSCOPE\_ALL**

Publications for this topic are propagated to hierarchically connected queue managers and to publish/subscribe cluster-connected queue managers.

**Note:** This behavior can be over-riden on a publication-by-publication basis, by using MQPMO\_SCOPE\_QMGR on the Put Message Options.

**z/OS QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

Table 203. QSGDisposition: Where objects are defined and how they behave		
QSGDisposition	Change	Copy, Create
<b>MQQSGD_COPY</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined by using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined by using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command. It uses the MQQSGD_GROUP object of the same name as the <i>ToTopicName</i> object (for Copy) or <i>TopicName</i> object (for Create).
<b>MQQSGD_GROUP</b>	<p>The object definition resides in the shared repository. The object was defined by using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group so that they refresh local copies on page set zero:</p> <pre>DEFINE TOPIC(name) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This definition is allowed only if the queue manager is in a queue sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group so that they make or refresh local copies on page set zero:</p> <pre>DEFINE TOPIC(name) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
<b>MQQSGD_PRIVATE</b>	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.

Table 203. QSGDisposition: Where objects are defined and how they behave (continued)

QSGDisposition	Change	Copy, Create
<b>MQQSGD_Q_MGR</b>	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This value is the default value.	The object is defined on the page set of the queue manager that executes the command. This value is the default value.

### Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF\_REPLACE).

If a topic definition with the same name as *ToTopicName* exists, this parameter specifies whether it is to be replaced. The value can be as follows:

#### **MQRP\_YES**

Replace existing definition.

#### **MQRP\_NO**

Do not replace existing definition.

### SubscriptionScope (MQCFIN)

Whether this queue manager propagates subscriptions for this topic, to queue managers as part of a hierarchy or as part of a publish/subscribe cluster (parameter identifier: MQIA\_SUB\_SCOPE).

The value can be:

#### **MQSCOPE\_AS\_PARENT**

Whether this queue manager propagates subscriptions, for this topic, to queue managers as part of a hierarchy or as part of a publish/subscribe-cluster is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

This value is the default value for this parameter if no value is specified.

#### **MQSCOPE\_QMGR**

Subscriptions for this topic are not propagated to other queue managers.

#### **MQSCOPE\_ALL**

Subscriptions for this topic are propagated to hierarchically connected queue managers and to publish/subscribe cluster-connected queue managers.

**Note:** This behavior can be over-ridden on a subscription-by-subscription basis, by using MQSO\_SCOPE\_QMGR on the Subscription Descriptor or SUBSCOPE(QMGR) on DEFINE SUB.

### TopicDesc (MQCFST)

Topic description (parameter identifier: MQCA\_TOPIC\_DESC).

Text that briefly describes the object

The maximum length is MQ\_TOPIC\_DESC\_LENGTH.

Use characters from the character set identified by the coded character set identifier (CCSID) for the message queue manager on which the command is executing to ensure that the text is translated correctly if it is sent to another queue manager.

### TopicType (MQCFIN)

Topic type (parameter identifier: MQIA\_TOPIC\_TYPE).

The value specified must match the type of the topic being changed. The value can be:

#### **MQTOPT\_LOCAL**

Local topic object

### **UseDLQ (MQCFIN)**

Determines whether the dead-letter queue is used when publication messages cannot be delivered to their correct subscriber queue (parameter identifier: MQIA\_USE\_DEAD\_LETTER\_Q).

The value can be any of the following values:

#### **MQUSEDLQ\_AS\_PARENT**

Determines whether to use the dead-letter queue using the setting of the closest administrative topic object in the topic tree. This value is the default supplied with IBM MQ, but your installation might have changed it.

#### **MQUSEDLQ\_NO**

Publication messages that cannot be delivered to their correct subscriber queue are treated as a failure to put the message. The MQPUT of an application to a topic fails in accordance with the settings of MQIA\_NPM\_DELIVERY and MQIA\_PM\_DELIVERY.

#### **MQUSEDLQ\_YES**

If the DEADQ queue manager attribute provides the name of a dead-letter queue then it is used, otherwise the behavior is as for MQUSEDLQ\_NO.

### **WildcardOperation (MQCFIN)**

Behavior of subscriptions including wildcards made to this topic (parameter identifier: MQIA\_WILDCARD\_OPERATION).

The value can be:

#### **MQTA\_PASSTHRU**

A less specific wildcard subscription is a subscription made by using wildcard topic names that are less specific than the topic string at this topic object. MQTA\_PASSTHRU lets less specific wildcard subscriptions receive publications made to this topic and to topic strings more specific than this topic. This value is the default supplied with IBM MQ.

#### **MQTA\_BLOCK**

A less specific wildcard subscription is a subscription made by using wildcard topic names that are less specific than the topic string at this topic object. MQTA\_BLOCK stops less specific wildcard subscriptions receiving publications made to this topic or to topic strings more specific than this topic.

This value of this attribute is used when subscriptions are defined. If you alter this attribute, the set of topics covered by existing subscriptions is not affected by the modification. This value applies also, if the topology is changed when topic objects are created or deleted; the set of topics matching subscriptions created following the modification of the **WildcardOperation** attribute is created by using the modified topology. If you want to force the matching set of topics to be re-evaluated for existing subscriptions, you must restart the queue manager.

### **MQCMD\_BACKUP\_CF\_STRUC (Backup CF Structure) on z/OS**

The Backup CF Structure (MQCMD\_BACKUP\_CF\_STRUC) PCF command initiates a CF application structure backup.

**Note:** This command is supported only on z/OS when the queue manager is a member of a queue sharing group.

### **Required parameters**

#### **CFStrucName (MQCFST)**

The name of the CF application structure to be backed up (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length is MQ\_CF\_STRUC\_NAME\_LENGTH.

### **Optional parameters**

#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **ExcludeInterval (MQCFIN)**

Exclude interval (parameter identifier: MQIACF\_EXCLUDE\_INTERVAL).

Specifies a value in seconds that defines the length of time immediately before the current time where the backup starts. The backup excludes backing-up the last *n* seconds activity. For example, if 30 seconds is specified, the backup does not include the last 30 seconds worth of activity for this application-structure.

The value must be in the range 30 through 600. The default value is 30.

## **MQCMD\_CHANGE\_PROT\_POLICY (set protection policy) AMS on Multiplatforms**

The set protection policy (MQCMD\_CHANGE\_PROT\_POLICY) PCF command uses Advanced Message Security (AMS) to set the protection policy.

**Important:** You must have an AMS license installed to issue this command. If you attempt to issue the **Set Policy** command without an AMS license installed, you receive message AMQ7155 - License file not found or not valid.

### **Syntax diagram**

See the syntax diagram in the MQSC [“SET POLICY \(set security policy\) on Multiplatforms”](#) on page 938 command for combinations of parameters and values that are allowed.

### **Required parameters**

#### **PolicyName (MQCFST)**

Specifies the name of the policy. The policy name must match the name of the queue which is to be protected (parameter identifier: MQCA\_POLICY\_NAME).

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

### **Optional parameters**

#### **SignAlg (MQCFIN)**

Specifies the digital signature algorithm (parameter identifier: MQIA\_SIGNATURE\_ALGORITHM). The following values are valid:

##### **MQMLP\_SIGN\_ALG\_NONE**

No digital signature algorithm specified. This is the default value.

##### **MQMLP\_SIGN\_ALG\_MD5**

   MD5 digital signature algorithm specified.

##### **MQMLP\_SIGN\_ALG\_SHA1**

   SHA1 digital signature algorithm specified.

##### **MQMLP\_SIGN\_ALG\_SHA256**

SHA256 digital signature algorithm specified.

**MQMLP\_SIGN\_ALG\_SHA384**

SHA384 digital signature algorithm specified.

**MQMLP\_SIGN\_ALG\_SHA512**

SHA512 digital signature algorithm specified.

**EncAlg (MQCFIN)**

Specifies the encryption algorithm (parameter identifier: MQIA\_ENCRYPTION\_ALGORITHM). The following values are valid:

**MQMLP\_ENCRYPTION\_ALG\_NONE**

No encryption algorithm specified. This is the default value.

**MQMLP\_ENCRYPTION\_ALG\_RC2**

 RC2 encryption algorithm specified.

**MQMLP\_ENCRYPTION\_ALG\_DES**

DES encryption algorithm specified.

**MQMLP\_ENCRYPTION\_ALG\_3DES**

3DES encryption algorithm specified.

**MQMLP\_ENCRYPTION\_ALG\_AES128**

AES128 encryption algorithm specified.

**MQMLP\_ENCRYPTION\_ALG\_AES256**

AES256 encryption algorithm specified.

**Signer (MQCFST)**

Specifies the distinguished name of an authorized signer. This parameter can be specified multiple times (parameter identifier: MQCA\_SIGNER\_DN).

**Recipient (MQCFST)**

Specifies the distinguished name of the intended recipient. This parameter can be specified multiple times (parameter identifier: MQCA\_RECIPIENT\_DN).

**Enforce and Tolerate (MQCFST)**

Indicates whether the security policy should be enforced or whether unprotected messages are tolerated (parameter identifier: MQIA\_TOLERATE\_UNPROTECTED). The following values are valid:

**MQMLP\_TOLERATE\_NO**

Specifies that all message must be protected when retrieved from the queue. Any unprotected message encountered is moved to the SYSTEM.PROTECTION.ERROR.QUEUE. This is the default value.

**MQMLP\_TOLERATE\_YES**

Specifies that the messages that are not protected when retrieved from the queue can ignore the policy.

Toleration is optional and exists to facilitate staged implementation, where:

- Policies have been applied to queues, but those queues might already contain unprotected messages, or
- Queues might still receive messages from remote systems that do not yet have the policy set.

**KeyReuse (MQCFIN)**

Specifies the number of times that an encryption key can be re-used, in the range 1-9,999,999, or the special values *MQKEY\_REUSE\_DISABLED* or *MQKEY\_REUSE\_UNLIMITED* (parameter identifier: MQIA\_KEY\_REUSE\_COUNT). The following values are valid:

**MQKEY\_REUSE\_DISABLED**

Prevents a symmetric key from being reused. This is the default value.

**MQKEY\_REUSE\_UNLIMITED**

Allows a symmetric key to be reused any number of times.





**Attention:** Key reuse is valid only for CONFIDENTIALITY policies, that is, **SignAlg** set to *MQESE\_SIGN\_ALG\_NONE* and **EncAlg** set to an algorithm value. For all other policy types, you must omit the parameter, or set the **Keyreuse** value to *MQKEY\_REUSE\_DISABLED*.

### Action (MQCFIN)

Specifies the action for the parameters supplied, as they apply to any existing policy (parameter identifier: MQIACF\_ACTION). The following values are valid:

#### MQACT\_REPLACE

Has the effect of replacing any existing policy with the parameters supplied. This is the default value.

#### MQACT\_ADD

Has the effect that signers and recipients parameters have an additive effect. That is, if a signer or recipient is specified, and does not already exist in a preexisting policy, the signer or recipient value is added to the existing policy definition.

#### MQACT\_REMOVE

Has the opposite effect of *MQACT\_ADD*. That is, if any of the signer or recipient values specified exist in a preexisting policy, those values are removed from the policy definition.

### Error codes

This command might return the following error codes in the response format header, in addition to the values shown at [“Error codes applicable to all commands”](#) on page 990.

#### Reason (MQLONG)

The value can be any of the following values:

##### MQRCCF\_POLICY\_TYPE\_ERROR

Policy type not valid.

### MQCMD\_CHANGE\_Q\_MGR (Change Queue Manager)

The Change Queue Manager (MQCMD\_CHANGE\_Q\_MGR) PCF command changes the specified attributes of the queue manager.

For any optional parameters that are omitted, the value does not change.

#### Required parameters:

None

### Optional parameters (Change Queue Manager)

Multi

#### AccountingConnOverride (MQCFIN)

Specifies whether applications can override the settings of the *QueueAccounting* and *MQIAccounting* queue manager parameters (parameter identifier: MQIA\_ACCOUNTING\_CONN\_OVERRIDE).

The value can be any of the following values:

##### MQMON\_DISABLED

Applications cannot override the settings of the **QueueAccounting** and **MQIAccounting** parameters.

This value is the initial default value for the queue manager.

##### MQMON\_ENABLED

Applications can override the settings of the **QueueAccounting** and **MQIAccounting** parameters by using the options field of the MQCNO structure of the MQCONN API call.

This parameter is valid only on [Multiplatforms](#).

### **Multi** AccountingInterval (MQCFIN)

The time interval, in seconds, at which intermediate accounting records are written (parameter identifier: MQIA\_ACCOUNTING\_INTERVAL).

Specify a value in the range 1 - 604,000.

This parameter is valid only on [Multiplatforms](#).

### **ActivityRecording (MQCFIN)**

Specifies whether activity reports can be generated (parameter identifier: MQIA\_ACTIVITY\_RECORDING).

The value can be:

#### **MQRECORDING\_DISABLED**

Activity reports cannot be generated.

#### **MQRECORDING\_MSG**

Activity reports can be generated and sent to the reply queue specified by the originator in the message causing the report.

#### **MQRECORDING\_Q**

Activity reports can be generated and sent to SYSTEM.ADMIN.ACTIVITY.QUEUE.

### **z/OS** AdoptNewMCACheck (MQCFIN)

The elements checked to determine whether an MCA must be adopted (restarted) when a new inbound channel is detected. It must be adopted (restarted) if it that has the same name as a currently active MCA (parameter identifier: MQIA\_ADOPTNEWMCA\_CHECK).

The value can be:

#### **MQADOPT\_CHECK\_Q\_MGR\_NAME**

Check the queue manager name.

#### **MQADOPT\_CHECK\_NET\_ADDR**

Check the network address.

#### **MQADOPT\_CHECK\_ALL**

Check the queue manager name and network address. Perform this check to prevent your channels from being inadvertently shut down. This value is the initial default value of the queue manager.

#### **MQADOPT\_CHECK\_NONE**

Do not check any elements.

This parameter applies to z/OS only.

### **z/OS** AdoptNewMCAType (MQCFIN)

Adoption of orphaned channel instances (parameter identifier: MQIA\_ADOPTNEWMCA\_TYPE).

Specify whether an orphaned MCA instance is to be adopted when a new inbound channel request is detected matching the **AdoptNewMCACheck** parameters.

The value can be:

#### **MQADOPT\_TYPE\_NO**

Do not adopt orphaned channel instances.

#### **MQADOPT\_TYPE\_ALL**

Adopt all channel types. This value is the initial default value of the queue manager.

This parameter applies to z/OS only.

### **AuthorityEvent (MQCFIN)**

Controls whether authorization (Not Authorized) events are generated (parameter identifier: MQIA\_AUTHORITY\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled. This value is not permitted on z/OS.

**BridgeEvent (MQCFIN)**

Controls whether IMS bridge events are generated (parameter identifier: MQIA\_BRIDGE\_EVENT). This parameter applies to z/OS only.

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled. This value is the default value.




**MQEVR\_ENABLED**

Event reporting enabled.

**CertificateLabel (MQCFST)**

Specifies the certificate label for this queue manager to use. The label identifies which personal certificate in the key repository has been selected (parameter identifier: MQCA\_CERT\_LABEL).

The default and migrated queue manager values are:

-  On AIX, Linux, and Windows: *ibmwebspheremqxxxx* where *xxxx* is the queue manager name folded to lowercase.
-  On IBM i:
  - If you specified SSLKEYR(\*SYSTEM), the value is blank.
  - Note that it is forbidden to use a nonblank queue manager CERTLABL with SSLKEYR(\*SYSTEM). Attempting to do so results in an MQRCCF\_Q\_MGR\_ATTR\_CONFLICT error.
  - Otherwise, *ibmwebspheremqxxxx* where *xxxx* is the queue manager name folded to lowercase.
-  On z/OS: *ibmWebSphereMQXXXX* where *XXXX* is the queue manager name.

See [z/OS systems](#) for more information.

**CertificateValPolicy (MQCFIN)**

Specifies which TLS certificate validation policy is used to validate digital certificates received from remote partner systems (parameter identifier: MQIA\_CERT\_VAL\_POLICY).

This attribute can be used to control how strictly the certificate chain validation conforms to industry security standards. For more information, see [Certificate validation policies in IBM MQ](#).

The value can be any of the following values:

**MQ\_CERT\_VAL\_POLICY\_ANY**

Apply each of the certificate validation policies supported by the secure sockets library and accept the certificate chain if any of the policies considers the certificate chain valid. This setting can be used for maximum backwards compatibility with older digital certificates which do not comply with the modern certificate standards.

**MQ\_CERT\_VAL\_POLICY\_RFC5280**

Apply only the RFC 5280 compliant certificate validation policy. This setting provides stricter validation than the ANY setting, but rejects some older digital certificates.

This parameter is only valid on AIX, Linux, and Windows and can be used only on a queue manager with a command level of 711, or higher.

Changes to **CertificateValPolicy** become effective either:

- When a new channel process is started.
- For channels that run as threads of the channel initiator, when the channel initiator is restarted.
- For channels that run as threads of the listener, when the listener is restarted.

- For channels that run as threads of a process pooling process, when the process pooling process is started or restarted and first runs a TLS channel. If the process pooling process has already run a TLS channel, and you want the change to become effective immediately, run the MQSC command **REFRESH SECURITY TYPE(SSL)**. The process pooling process is amqzmpa on AIX, Linux, and Windows.
- When a **REFRESH SECURITY TYPE(SSL)** command is issued.

### z/OS CFConlos (MQCFIN)

Specifies the action to be taken when the queue manager loses connectivity to the administration structure, or any CF structure with CFConlos set to ASQMGR (parameter identifier: MQIA\_QMGR\_CFCONLOS).

The value can be:

#### **MQCFCONLOS\_TERMINATE**

The queue manager terminates when connectivity to CF structures is lost.

#### **MQCFCONLOS\_TOLERATE**

The queue manager tolerates loss of connectivity to CF structures without terminating.

This parameter applies to z/OS only.

### ChannelAutoDef (MQCFIN)

Controls whether receiver and server-connection channels can be auto-defined (parameter identifier: MQIA\_CHANNEL\_AUTO\_DEF).

Auto-definition for cluster-sender channels is always enabled.

This parameter is supported in the following environments: IBM i, AIX, Linux, and Windows systems.

The value can be:

#### **MQCHAD\_DISABLED**

Channel auto-definition disabled.

#### **MQCHAD\_ENABLED**

Channel auto-definition enabled.

### ChannelAutoDefEvent (MQCFIN)

Controls whether channel auto-definition events are generated (parameter identifier: MQIA\_CHANNEL\_AUTO\_DEF\_EVENT), when a receiver, server-connection, or cluster-sender channel is auto-defined.

This parameter is supported in the following environments: IBM i, AIX, Linux, and Windows systems.

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

### ChannelAutoDefExit (MQCFIN)

Channel auto-definition exit name (parameter identifier: MQCA\_CHANNEL\_AUTO\_DEF\_EXIT).

This exit is invoked when an inbound request for an undefined channel is received, if:

1. The channel is a cluster-sender, or
2. Channel auto-definition is enabled (see *ChannelAutoDef*).

This exit is also invoked when a cluster-receiver channel is started.

The format of the name is the same as for the *SecurityExit* parameter described in [“Change, Copy, and Create Channel”](#) on page 1011.

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

This parameter is supported in the following environments: z/OS, IBM i, AIX, Linux, and Windows. On z/OS, it applies only to cluster-sender and cluster-receiver channels.

### **ChannelAuthenticationRecords (MQCFIN)**

Controls whether channel authentication records are used. Channel authentication records can still be set and displayed regardless of the value of this attribute. (parameter identifier: MQIA\_CHLAUTH\_RECORDS).

The value can be:

#### **MQCHLA\_DISABLED**

Channel authentication records are not checked.

#### **MQCHLA\_ENABLED**

Channel authentication records are checked.

### **ChannelEvent (MQCFIN)**

Controls whether channel events are generated (parameter identifier: MQIA\_CHANNEL\_EVENT).

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

#### **MQEVR\_EXCEPTION**

Reporting of exception channel events enabled.

### **Multi ChannelInitiatorControl (MQCFIN)**

Specifies whether the channel initiator is to be started when the queue manager starts (parameter identifier: MQIA\_CHINIT\_CONTROL).

The value can be:

#### **MQSVC\_CONTROL\_MANUAL**

The channel initiator is not to be started automatically.

#### **MQSVC\_CONTROL\_Q\_MGR**

The channel initiator is to be started automatically when the queue manager starts.

This parameter is valid only on [Multiplatforms](#).

### **ChannelMonitoring (MQCFIN)**

Default setting for online monitoring for channels (parameter identifier: MQIA\_MONITORING\_CHANNEL).

The value can be:

#### **MQMON\_NONE**

Online monitoring data collection is turned off for channels regardless of the setting of their **ChannelMonitoring** parameter.

#### **MQMON\_OFF**

Online monitoring data collection is turned off for channels specifying a value of MQMON\_Q\_MGR in their **ChannelMonitoring** parameter. This value is the initial default value of the queue manager.

#### **MQMON\_LOW**

Online monitoring data collection is turned on, with a low ratio of data collection, for channels specifying a value of MQMON\_Q\_MGR in their **ChannelMonitoring** parameter.

**MQMON\_MEDIUM**

Online monitoring data collection is turned on, with a moderate ratio of data collection, for channels specifying a value of MQMON\_Q\_MGR in their **ChannelMonitoring** parameter.

**MQMON\_HIGH**

Online monitoring data collection is turned on, with a high ratio of data collection, for channels specifying a value of MQMON\_Q\_MGR in their **ChannelMonitoring** parameter.

**ChannelStatistics (MQCFIN)**

Controls whether statistics data is to be collected for channels (parameter identifier: MQIA\_STATISTICS\_CHANNEL).

The value can be:

**MQMON\_NONE**

Statistics data collection is turned off for channels regardless of the setting of their **ChannelStatistics** parameter. This value is the initial default value of the queue manager.

**MQMON\_OFF**

Statistics data collection is turned off for channels specifying a value of MQMON\_Q\_MGR in their *ChannelStatistics* parameter.

**MQMON\_LOW**

Statistics data collection is turned on, with a low ratio of data collection, for channels specifying a value of MQMON\_Q\_MGR in their **ChannelStatistics** parameter.

**MQMON\_MEDIUM**

Statistics data collection is turned on, with a moderate ratio of data collection, for channels specifying a value of MQMON\_Q\_MGR in their **ChannelStatistics** parameter.

**MQMON\_HIGH**

Statistics data collection is turned on, with a high ratio of data collection, for channels specifying a value of MQMON\_Q\_MGR in their **ChannelStatistics** parameter.

**z/OS** On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

**z/OS ChinitAdapters (MQCFIN)**

Number of adapter subtasks (parameter identifier: MQIA\_CHINIT\_ADAPTERS).

The number of adapter subtasks to use for processing IBM MQ calls. This parameter applies to z/OS only.

Specify a value in the range 1 - 9999. The initial default value of the queue manager is 8.

**z/OS ChinitDispatchers (MQCFIN)**

Number of dispatchers (parameter identifier: MQIA\_CHINIT\_DISPATCHERS).

The number of dispatchers to use for the channel initiator. This parameter applies to z/OS only.

Specify a value in the range 1 - 9999. The initial default value of the queue manager is 5.

**z/OS ChinitServiceParm (MQCFIN)**

Reserved for use by IBM (parameter identifier: MQCA\_CHINIT\_SERVICE\_PARM).

This parameter applies to z/OS only.

**z/OS ChinitTraceAutoStart (MQCFIN)**

Specifies whether the channel initiator trace must start automatically (parameter identifier: MQIA\_CHINIT\_TRACE\_AUTO\_START).

The value can be:

**MQTRAXSTR\_YES**

Channel initiator trace is to start automatically.

## **MQTRAXSTR\_NO**

Channel initiator trace is not to start automatically. This value is the initial default value of the queue manager.

This parameter applies to z/OS only.

## **z/OS ChinitTraceTableSize (MQCFIN)**

The size, in megabytes, of the trace data space of the channel initiator (parameter identifier: MQIA\_CHINIT\_TRACE\_TABLE\_SIZE).

Specify a value in the range 2 - 2048. The initial default value of the queue manager is 2.

This parameter applies to z/OS only.

## **ClusterSenderMonitoringDefault (MQCFIN)**

Default setting for online monitoring for automatically defined cluster-sender channels (parameter identifier: MQIA\_MONITORING\_AUTO\_CLUSSDR).

Specifies the value to be used for the *ChannelMonitoring* attribute of automatically defined cluster-sender channels. The value can be any of the following values:

### **MQMON\_Q\_MGR**

Collection of online monitoring data is inherited from the setting of the queue manager's **ChannelMonitoring** parameter. This value is the initial default value of the queue manager.

### **MQMON\_OFF**

Monitoring for the channel is disabled.

### **MQMON\_LOW**

Unless *ChannelMonitoring* is MQMON\_NONE, this value specifies a low rate of data collection with a minimal effect on system performance. The data collected is not likely to be the most current.

### **MQMON\_MEDIUM**

Unless *ChannelMonitoring* is MQMON\_NONE, this value specifies a moderate rate of data collection with limited effect on system performance.

### **MQMON\_HIGH**

Unless *ChannelMonitoring* is MQMON\_NONE, this value specifies a high rate of data collection with a likely effect on system performance. The data collected is the most current available.

**z/OS** On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results.

## **ClusterSenderStatistics (MQCFIN)**

Controls whether statistics data is to be collected for auto-defined cluster-sender channels (parameter identifier: MQIA\_STATISTICS\_AUTO\_CLUSSDR).

The value can be:

### **MQMON\_Q\_MGR**

Collection of statistics data is inherited from the setting of the queue manager's **ChannelStatistics** parameter. This value is the initial default value of the queue manager.

### **MQMON\_OFF**

Statistics data collection for the channel is disabled.

### **MQMON\_LOW**

Unless *ChannelStatistics* is MQMON\_NONE, this value specifies a low rate of data collection with a minimal effect on system performance.

### **MQMON\_MEDIUM**

Unless *ChannelStatistics* is MQMON\_NONE, this value specifies a moderate rate of data collection.

### **MQMON\_HIGH**

Unless *ChannelStatistics* is MQMON\_NONE, this value specifies a high rate of data collection.



On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

### **ClusterWorkLoadData (MQCFST)**

Cluster workload exit data (parameter identifier: MQCA\_CLUSTER\_WORKLOAD\_DATA).

This parameter is passed to the cluster workload exit when it is called.

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

### **ClusterWorkLoadExit (MQCFST)**

Cluster workload exit name (parameter identifier: MQCA\_CLUSTER\_WORKLOAD\_EXIT).

If a nonblank name is defined this exit is invoked when a message is put to a cluster queue.

The format of the name is the same as for the *SecurityExit* parameter described in [“Change, Copy, and Create Channel”](#) on page 1011.

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

### **ClusterWorkLoadLength (MQCFIN)**

Cluster workload length (parameter identifier: MQIA\_CLUSTER\_WORKLOAD\_LENGTH).

The maximum length of the message passed to the cluster workload exit.

The value of this attribute must be in the range 0 - 999,999 999.

### **CLWLMRUChannels (MQCFIN)**

Cluster workload most recently used (MRU) channels (parameter identifier: MQIA\_CLWL\_MRU\_CHANNELS).

The maximum number of active most recently used outbound channels.

Specify a value in the range 1 - 999,999 999.

### **CLWLUseQ (MQCFIN)**

Use of remote queue (parameter identifier: MQIA\_CLWL\_USEQ).

Specifies whether a cluster queue manager is to use remote puts to other queues defined in other queue managers within the cluster during workload management.

Specify either:

#### **MQCLWL\_USEQ\_ANY**

Use remote queues.

#### **MQCLWL\_USEQ\_LOCAL**

Do not use remote queues.

### **CodedCharSetId (MQCFIN)**

Queue manager coded character set identifier (parameter identifier: MQIA\_CODED\_CHAR\_SET\_ID).

The coded character set identifier (CCSID) for the queue manager. The CCSID is the identifier used with all character string fields defined by the application programming interface (API). If the CCSID in a message descriptor is set to the value MQCCSI\_Q\_MGR, it applies to the character data written into the body of a message. Data is written using MQPUT or MQPUT1. Character data is identified by the format specified for the message.

Specify a value in the range 1 - 65,535.

The CCSID must specify a value that is defined for use on the platform and use an appropriate character set. The character set must be:

- EBCDIC on IBM i
- ASCII or ASCII-related on other platforms



Stop and restart the queue manager after execution of this command so that all processes reflect the changed CCSID of the queue manager.

This parameter is not supported on z/OS.

### **CommandEvent (MQCFIN)**

Controls whether command events are generated (parameter identifier: MQIA\_COMMAND\_EVENT).

The value can be any of the following values:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

#### **MQEVR\_NO\_DISPLAY**

Event reporting enabled for all successful commands except Inquire commands.

### **z/OS CommandScope (MQCFIN)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following values:

- Blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- A queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment. The command server must be enabled.
- An asterisk " \* ". The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **Multi CommandServerControl (MQCFIN)**

Specifies whether the command server is to be started when the queue manager starts (parameter identifier: MQIA\_CMD\_SERVER\_CONTROL).

The value can be:

#### **MQSVC\_CONTROL\_MANUAL**

The command server is not to be started automatically.

#### **MQSVC\_CONTROL\_Q\_MGR**

The command server is to be started automatically when the queue manager starts.

This parameter is valid only on [Multiplatforms](#).

### **ConfigurationEvent (MQCFIN)**

Controls whether configuration events are generated (parameter identifier: MQIA\_CONFIGURATION\_EVENT).

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

### **ConnAuth (MQCFST)**

The name of an authentication information object that is used to provide the location of user ID and password authentication (parameter identifier: MQCA\_CONN\_AUTH).

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH. Only authentication information objects with type IDPWOS or IDPWLDAP can be specified; other types result in an error message when the OAM (on AIX, Linux, and Windows) or the security component (on z/OS) reads the configuration.

### **Custom (MQCFST)**

Custom attribute for new features (parameter identifier: MQCA\_CUSTOM).

This attribute is reserved for the configuration of new features before separate attributes are introduced. It can contain the values of zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE). Single quotation marks must be escaped with another single quotation mark.

This description is updated when features using this attribute are introduced. Currently there are no possible values for *Custom*.

The maximum length of the string is MQ\_CUSTOM\_LENGTH.

### **DeadLetterQName (MQCFIN)**

Dead letter (undelivered message) queue name (parameter identifier: MQCA\_DEAD\_LETTER\_Q\_NAME).

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination. The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **DefClusterXmitQueueType (MQCFIN)**

The DefClusterXmitQueueType attribute controls which transmission queue is selected by default by cluster-sender channels to get messages from, to send the messages to cluster-receiver channels. (Parameter identifier: MQIA\_DEF\_CLUSTER\_XMIT\_Q\_TYPE.)

The values of **DefClusterXmitQueueType** are MQCLXQ\_SCTQ or MQCLXQ\_CHANNEL.

#### **MQCLXQ\_SCTQ**

All cluster-sender channels send messages from SYSTEM.CLUSTER.TRANSMIT.QUEUE. The correlID of messages placed on the transmission queue identifies which cluster-sender channel the message is destined for.

SCTQ is set when a queue manager is defined.

#### **MQCLXQ\_CHANNEL**

Each cluster-sender channel sends messages from a different transmission queue. Each transmission queue is created as a permanent dynamic queue from the model queue SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

### **DefXmitQName (MQCFST)**

Default transmission queue name (parameter identifier: MQCA\_DEF\_XMIT\_Q\_NAME).

This parameter is the name of the default transmission queue that is used for the transmission of messages to remote queue managers. It is selected if there is no other indication of which transmission queue to use.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **DNSGroup (MQCFST)**

DNS group name (parameter identifier: MQCA\_DNS\_GROUP).

This parameter is no longer used. See [z/OS: WLM/DNS no longer supported](#). This parameter applies to z/OS only.

The maximum length of the string is MQ\_DNS\_GROUP\_NAME\_LENGTH.

### **z/OS DNSWLM (MQCFIN)**

WLM/DNS Control: (parameter identifier: MQIA\_DNS\_WLM).

This parameter is no longer used. See [z/OS: WLM/DNS no longer supported](#).

The value can be any of the following values:

**MQDNSWLM\_NO**

This is the only value supported by the queue manager.

This parameter applies to z/OS only.

**z/OS ExpiryInterval (MQCFIN)**

Interval between scans for expired messages (parameter identifier: MQIA\_EXPIRY\_INTERVAL). This parameter applies to z/OS only.

Specifies the frequency with which the queue manager scans the queues looking for expired messages. Specify a time interval in seconds in the range 1 - 99,999,999, or the following special value:

**MQEXPI\_OFF**

No scans for expired messages.

The minimum scan interval used is 5 seconds, even if you specify a lower value.

**EncryptionPolicySuiteB (MQCFIL)**

Specifies whether Suite B-compliant cryptography is used and what level of strength is employed (parameter identifier MQIA\_SUITE\_B\_STRENGTH).

The value can be one or more of:

**MQ\_SUITE\_B\_NONE**

Suite B-compliant cryptography is not used.

**MQ\_SUITE\_B\_128\_BIT**

Suite B 128-bit strength security is used.

**MQ\_SUITE\_B\_192\_BIT**

Suite B 192-bit strength security is used.

If invalid lists are specified, such as MQ\_SUITE\_B\_NONE with MQ\_SUITE\_B\_128\_BIT, the error MQRCCF\_SUITE\_B\_ERROR is issued.

**Force (MQCFIN)**

Force changes (parameter identifier: MQIACF\_FORCE).

Specifies whether the command is forced to complete if both of the following are true:

- *DefXmitQName* is specified, and
- An application has a remote queue open, the resolution for which is affected by this change.

**z/OS GroupUR (MQCFIN)**

Controls whether CICS and XA client applications can establish transactions with a GROUP unit of recovery disposition.

This attribute is only valid on z/OS and can be enabled only when the queue manager is a member of a queue sharing group.

The value can be:

**MQGUR\_DISABLED**

CICS and XA client applications must connect using a queue manager name.

**MQGUR\_ENABLED**

CICS and XA client applications can establish transactions with a group unit of recovery disposition by specifying a queue sharing group name when they connect.

**z/OS** See [Unit of recovery disposition in a queue sharing group](#).

**IGQPutAuthority (MQCFIN)**

Command scope (parameter identifier: MQIA\_IGQ\_PUT\_AUTHORITY). This parameter is valid only on z/OS when the queue manager is a member of a queue sharing group.

Specifies the type of authority checking and, therefore, the user IDs to be used by the IGQ agent (IGQA). This parameter establishes the authority to put messages to a destination queue. The value can be any of the following values:

**MQIGQPA\_DEFAULT**

Default user identifier is used.

The user identifier used for authorization is the value of the *UserIdentifier* field. The *UserIdentifier* field is in the separate MQMD that is associated with the message when the message is on the shared transmission queue. This value is the user identifier of the program that placed the message on the shared transmission queue. It is typically the same as the user identifier under which the remote queue manager is running.

If the RESLEVEL profile indicates that more than one user identifier is to be checked, the user identifier of the local IGQ agent (*IGQUserId*) is checked.

**MQIGQPA\_CONTEXT**

Context user identifier is used.

The user identifier used for authorization is the value of the *UserIdentifier* field. The *UserIdentifier* field is in the separate MQMD that is associated with the message when the message is on the shared transmission queue. This value is the user identifier of the program that placed the message on the shared transmission queue. It is typically the same as the user identifier under which the remote queue manager is running.

If the RESLEVEL profile indicates that more than one user identifier is to be checked, the user identifier of the local IGQ agent (*IGQUserId*) is checked.. The value of the *UserIdentifier* field in the embedded MQMD is also checked. The latter user identifier is typically the user identifier of the application that originated the message.

**MQIGQPA\_ONLY\_IGQ**

Only the IGQ user identifier is used.

The user identifier used for authorization is the user identifier of the local IGQ agent (*IGQUserId*).

If the RESLEVEL profile indicates that more than one user identifier is to be checked, this user identifier is used for all checks.

**MQIGQPA\_ALTERNATE\_OR\_IGQ**

Alternate user identifier or IGQ-agent user identifier is used.

The user identifier used for authorization is the user identifier of the local IGQ agent (*IGQUserId*).

If the RESLEVEL profile indicates that more than one user identifier is to be checked, the value of the *UserIdentifier* field in the embedded MQMD is also checked. The latter user identifier is typically the user identifier of the application that originated the message.

**IGQUserId (MQCFST)**

Intra-group queuing agent user identifier (parameter identifier: MQCA\_IGQ\_USER\_ID). This parameter is valid only on z/OS when the queue manager is a member of a queue sharing group.

Specifies the user identifier that is associated with the local intra-group queuing agent. This identifier is one of the user identifiers that might be checked for authorization when the IGQ agent puts messages on local queues. The actual user identifiers checked depend on the setting of the *IGQPutAuthority* attribute, and on external security options.

The maximum length is MQ\_USER\_ID\_LENGTH.

**ImageInterval (MQCFIN)**

The target frequency with which the queue manager automatically writes media images, in minutes since the previous media image for an object (parameter identifier: MQIA\_MEDIA\_IMAGE\_INTERVAL). This parameter is not valid on z/OS.

The value can be:

The time in minutes from 1 - 999 999 999, at which the queue manager automatically writes media images.

The default value is 60 minutes.

**MQMEDIMGINTVL\_OFF**

Automatic media images are not written on a time interval basis.

**ImageLogLength (MQCFIN)**

The target size of the recovery log, written before the queue manager automatically writes media images, in number of megabytes since the previous media image for an object. This limits the amount of log to be read when recovering an object (parameter identifier: MQIA\_MEDIA\_IMAGE\_LOG\_LENGTH). This parameter is not valid on z/OS.

The value can be:

The target size of the recovery log in megabytes from 1 - 999 999 999.

**MQMEDIMGLOGLN\_OFF**

Automatic media images are not written based on the size of log written.

MQMEDIMGLOGLN\_OFF is the default value.

**ImageRecoverObject (MQCFST)**

Specifies whether authentication information, channel, client connection, listener, namelist, process, alias queue, remote queue, and service objects are recoverable from a media image, if linear logging is being used (parameter identifier: MQIA\_MEDIA\_IMAGE\_RECOVER\_OBJ). This parameter is not valid on z/OS.

The value can be:

**MQIMGRCOV\_NO**

The “rcdmqimg (record media image)” on page 139 and “rcrmqobj (re-create object)” on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

**MQIMGRCOV\_YES**

These objects are recoverable.

MQIMGRCOV\_YES is the default value.

**ImageRecoverQueue (MQCFST)**

Specifies the default **ImageRecoverQueue** attribute for local and permanent dynamic queue objects, when used with this parameter (parameter identifier: MQIA\_MEDIA\_IMAGE\_RECOVER\_Q). This parameter is not valid on z/OS.

The value can be:

**MQIMGRCOV\_NO**

The **ImageRecoverQueue** attribute for local and permanent dynamic queue objects is set to MQIMGRCOV\_NO .

**MQIMGRCOV\_YES**

The **ImageRecoverQueue** attribute for local and permanent dynamic queue objects is set to MQIMGRCOV\_YES .

MQIMGRCOV\_YES is the default value.

**ImageSchedule (MQCFST)**

Whether the queue manager automatically writes media images (parameter identifier: MQIA\_MEDIA\_IMAGE\_SCHEDULING). This parameter is not valid on z/OS.

The value can be:

#### **MQMEDIMGSCHED\_AUTO**

The queue manager attempts to automatically write a media image for an object, before **ImageInterval** minutes have elapsed, or **ImageLogLength** megabytes of recovery log have been written, since the previous media image for the object was taken.

The previous media image might have been taken manually or automatically, depending on the settings of **ImageInterval** or **ImageLogLength**.

#### **MQMEDIMGSCHED\_MANUAL**

Automatic media images are not written.

MQMEDIMGSCHED\_MANUAL is the default value.

#### **InhibitEvent (MQCFIN)**

Controls whether inhibit (Inhibit Get and Inhibit Put) events are generated (parameter identifier: MQIA\_INHIBIT\_EVENT).

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

#### **InitialKey (MQCFST)**

The initial key for the password protection system (parameter identifier: MQCA\_INITIAL\_KEY).

The length of the string is MQ\_INITIAL\_KEY\_LENGTH.

IBM MQ encrypts the value of some queue manager attributes using the IBM MQ Password Protection system. An initial key is used by the encryption algorithm to encrypt and decrypt these attributes. You should use this attribute to set a unique initial key for the queue manager, prior to setting values of the attributes that are encrypted.

If an initial key is not set using this attribute, a default initial key is used. If this attribute is changed, the values of the attributes that are encrypted are invalidated and must be reset before they can be used.

The value of the **SSLKeyRepositoryPassword** attribute is encrypted using the initial key.



**Warning:** If you change this attribute, the values of the attributes that are encrypted are invalidated and you must reset them before they can be used.

#### **z/OS IntraGroupqueuing (MQCFIN)**

Command scope (parameter identifier: MQIA\_INTRA\_GROUP\_QUEUING). This parameter is valid only on z/OS when the queue manager is a member of a queue sharing group.

Specifies whether intra-group queuing is used. The value can be any of the following values:

#### **MQIGQ\_DISABLED**

Intra-group queuing disabled.

#### **MQIGQ\_ENABLED**

Intra-group queuing enabled.

#### **IPAddressVersion (MQCFIN)**

IP address version selector (parameter identifier: MQIA\_IP\_ADDRESS\_VERSION).

Specifies which IP address version, either IPv4 or IPv6, is used. The value can be:

#### **MQIPADDR\_IPV4**

IPv4 is used.

#### **MQIPADDR\_IPV6**

IPv6 is used.

This parameter is only relevant for systems that run both IPv4 and IPv6. It affects only channels defined as having a *TransportType* of MQXPY\_TCP when one of the following conditions is true:

- The channel attribute *ConnectionName* is a host name that resolves to both an IPv4 and IPv6 address and its **LocalAddress** parameter is not specified.
- The channel attributes *ConnectionName* and *LocalAddress* are both host names that resolve to both IPv4 and IPv6 addresses.

#### **z/OS** ListenerTimer (MQCFIN)

Listener restart interval (parameter identifier: MQIA\_LISTENER\_TIMER).

The time interval, in seconds, between attempts by IBM MQ to restart the listener after an APPC or TCP/IP failure. This parameter applies to z/OS only.

Specify a value in the range 5 - 9,999. The initial default value of the queue manager is 60.

#### **LocalEvent (MQCFIN)**

Controls whether local error events are generated (parameter identifier: MQIA\_LOCAL\_EVENT).

The value can be:

##### **MQEVR\_DISABLED**

Event reporting disabled.

##### **MQEVR\_ENABLED**

Event reporting enabled.

#### **Multi** LoggerEvent (MQCFIN)

Controls whether recovery log events are generated (parameter identifier: MQIA\_LOGGER\_EVENT).

The value can be:

##### **MQEVR\_DISABLED**

Event reporting disabled.

##### **MQEVR\_ENABLED**

Event reporting enabled. This value is valid only on queue managers that use linear logging.

This parameter is valid only on [Multiplatforms](#).

#### **z/OS** LUGroupName (MQCFST)

Generic LU name for the LU 6.2 listener (parameter identifier: MQCA\_LU\_GROUP\_NAME).

The generic LU name to be used by the LU 6.2 listener that handles inbound transmissions for the queue sharing group.

This parameter applies to z/OS only.

The maximum length of the string is MQ\_LU\_NAME\_LENGTH.

#### **z/OS** LUName (MQCFST)

LU name to use for outbound LU 6.2 transmissions (parameter identifier: MQCA\_LU\_NAME).

The name of the LU to use for outbound LU 6.2 transmissions. Set this parameter to be the same as the name of the LU to be used by the listener for inbound transmissions.

This parameter applies to z/OS only.

The maximum length of the string is MQ\_LU\_NAME\_LENGTH.

#### **z/OS** LU62ARMSuffix (MQCFST)

APPCPM suffix (parameter identifier: MQCA\_LU62\_ARM\_SUFFIX).

The suffix of the APPCPM member of SYS1.PARMLIB. This suffix nominates the LUADD for this channel initiator.

This parameter applies to z/OS only.

The maximum length of the string is MQ\_ARM\_SUFFIX\_LENGTH.

**z/OS LU62Channels (MQCFIN)**

Maximum number of LU 6.2 channels (parameter identifier: MQIA\_LU62\_CHANNELS).

The maximum number of channels that can be current, or clients that can be connected, that use the LU 6.2 transmission protocol.

This parameter applies to z/OS only.

Specify a value in the range 0 - 9999. The initial default value of the queue manager is 200.

**z/OS MaxActiveChannels (MQCFIN)**

Maximum number of active channels (parameter identifier: MQIA\_ACTIVE\_CHANNELS ).

The maximum number of channels that can be *active* at any time.

This parameter applies to z/OS only.

Sharing conversations do not contribute to the total for this parameter.

Specify a value in the range 1 - 9999. The initial default value of the queue manager is 200.

**z/OS MaxChannels (MQCFIN)**

Maximum number of current channels (parameter identifier: MQIA\_MAX\_CHANNELS).

The maximum number of channels that can be *current* (including server-connection channels with connected clients).

This parameter applies to z/OS only.

Sharing conversations do not contribute to the total for this parameter.

Specify a value in the range 1 - 9999.

**MaxHandles (MQCFIN)**

Maximum number of handles (parameter identifier: MQIA\_MAX\_HANDLES).

The maximum number of handles that any one connection can have open at the same time.

Specify a value in the range 0 - 999,999,999.

**MaxMsgLength (MQCFIN)**

Maximum message length (parameter identifier: MQIA\_MAX\_MSG\_LENGTH).

Specifies the maximum length of messages allowed on queues on the queue manager. No message that is larger than either the queue attribute *MaxMsgLength* or the queue manager attribute *MaxMsgLength* can be put on a queue.

If you reduce the maximum message length for the queue manager, you must also reduce the maximum message length of the SYSTEM.DEFAULT.LOCAL.QUEUE definition, and your other queues. Reduce the definitions on the queues to less than or equal to the limit of the queue manager. If you do not reduce the message lengths appropriately, and applications inquire only the value of the queue attribute *MaxMsgLength*, they might not work correctly.

The lower limit for this parameter is 32 KB (32,768 bytes). The upper limit is 100 MB (104,857,600 bytes).

This parameter is not valid on z/OS.

**MaxPropertiesLength (MQCFIN)**

Maximum property length (parameter identifier: MQIA\_MAX\_PROPERTIES\_LENGTH).

Specifies the maximum length of the properties, including both the property name in bytes and the size of the property value in bytes.

Specify a value in the range 0 - 100 MB (104,857,600 bytes), or the special value:



## **MQPROP\_UNRESTRICTED\_LENGTH**

The size of the properties is restricted only by the upper limit.

## **MaxUncommittedMsgs (MQCFIN)**

Maximum uncommitted messages (parameter identifier: MQIA\_MAX\_UNCOMMITTED\_MSGS).

Specifies the maximum number of uncommitted messages. The maximum number of uncommitted messages under any sync point is the sum of the following messages:

The number of messages that can be retrieved.

The number of messages that can be put.

The number of trigger messages generated within this unit of work.

The limit does not apply to messages that are retrieved or put outside sync point.

Specify a value in the range 1 - 10,000.

**Multi**

## **MQIAccounting (MQCFIN)**

Controls whether accounting information for MQI data is to be collected (parameter identifier: MQIA\_ACCOUNTING\_MQI).

The value can be:

### **MQMON\_OFF**

MQI accounting data collection is disabled. This value is the initial default value of the queue manager.

### **MQMON\_ON**

MQI accounting data collection is enabled.

This parameter is valid only on [Multiplatforms](#).

**Multi**

## **MQIStatistics (MQCFIN)**

Controls whether statistics monitoring data is to be collected for the queue manager (parameter identifier: MQIA\_STATISTICS\_MQI).

The value can be:

### **MQMON\_OFF**

Data collection for MQI statistics is disabled. This value is the initial default value of the queue manager.

### **MQMON\_ON**

Data collection for MQI statistics is enabled.

This parameter is valid only on [Multiplatforms](#).

## **MsgMarkBrowseInterval (MQCFIN)**

Mark-browse interval (parameter identifier: MQIA\_MSG\_MARK\_BROWSE\_INTERVAL).

Specifies the time interval in milliseconds after which the queue manager can automatically unmark messages.

Specify a value up to the maximum of 999,999,999, or the special value MQMMBI\_UNLIMITED. The default value is 5000.



**Attention:** You should not reduce the value below the default of 5000.

MQMMBI\_UNLIMITED indicates that the queue manager does not automatically unmark messages.

**z/OS**

## **OutboundPortMax (MQCFIN)**

The maximum value in the range for the binding of outgoing channels (parameter identifier: MQIA\_OUTBOUND\_PORT\_MAX).

The maximum value in the range of port numbers to be used when binding outgoing channels. This parameter applies to z/OS only.

Specify a value in the range 0 - 65,535. The initial default value of the queue manager is zero.

Specify a corresponding value for *OutboundPortMin* and ensure that the value of *OutboundPortMax* is greater than or equal to the value of *OutboundPortMin*.

### **z/OS** **OutboundPortMin (MQCFIN)**

The minimum value in the range for the binding of outgoing channels (parameter identifier: MQIA\_OUTBOUND\_PORT\_MIN).

The minimum value in the range of port numbers to be used when binding outgoing channels. This parameter applies to z/OS only.

Specify a value in the range 0 - 65,535. The initial default value of the queue manager is zero.

Specify a corresponding value for *OutboundPortMax* and ensure that the value of *OutboundPortMin* is less than or equal to the value of *OutboundPortMax*.

### **Parent (MQCFST)**

The name of the queue manager to which this queue manager is to connect hierarchically as its child (parameter identifier: MQCA\_PARENT).

A blank value indicates that this queue manager has no parent queue manager. If there is an existing parent queue manager it is disconnected. This value is the initial default value of the queue manager.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

#### **Note:**

- The use of IBM MQ hierarchical connections requires that the queue manager attribute PSMODE is set to MQPSM\_ENABLED.
- The value of *Parent* can be set to a blank value if PSMODE is set to MQPSM\_DISABLED.
- Before connecting to a queue manager hierarchically as its child, channels in both directions must exist between the parent queue manager and child queue manager.
- If a parent is defined, the **Change Queue Manager** command disconnects from the original parent and sends a connection flow to the new parent queue manager.
- Successful completion of the command does not mean that the action completed or that it is going to complete successfully. Use the **Inquire Pub/Sub Status** command to track the status of the requested parent relationship.

### **PerformanceEvent (MQCFIN)**

Controls whether performance-related events are generated (parameter identifier: MQIA\_PERFORMANCE\_EVENT).

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

### **PubSubClus (MQCFIN)**

Controls whether the queue manager participates in publish/subscribe clustering (parameter identifier: MQIA\_PUBSUB\_CLUSTER).

The value can be:

#### **MQPSCLUS\_ENABLED**

The creating or receipt of clustered topic definitions and cluster subscriptions is permitted.

**Note:** The introduction of a clustered topic into a large IBM MQ cluster can cause a degradation in performance. This degradation occurs because all partial repositories are notified of all the other members of the cluster. Unexpected subscriptions might be created at all other nodes; for example, where proxysub (FORCE) is specified. Large numbers of channels might be started from a queue manager; for example, on resync after a queue manager failure.

**MQPSCLUS\_DISABLED**

The creating or receipt of clustered topic definitions and cluster subscriptions is inhibited. The creations or receipts are recorded as warnings in the queue manager error logs.

**PubSubMaxMsgRetryCount (MQCFIN)**

The number of attempts to reprocess a message when processing a failed command message under sync point (parameter identifier: MQIA\_PUBSUB\_MAXMSG\_RETRY\_COUNT).

The value can be:

**0 to 999 999 999**

The initial value is 5.

**PubSubMode (MQCFIN)**

Specifies whether the publish/subscribe engine and the queued publish/subscribe interface are running. The publish/subscribe engine enables applications to publish or subscribe by using the application programming interface. The publish/subscribe interface monitors the queues used the queued publish/subscribe interface (parameter identifier: MQIA\_PUBSUB\_MODE).

The value can be:

**MQPSM\_COMPAT**

The publish/subscribe engine is running. It is therefore possible to publish or subscribe by using the application programming interface. The queued publish/subscribe interface is not running. Therefore any message that is put to the queues that are monitored by the queued publish/subscribe interface is not acted on. MQPSM\_COMPAT is used for compatibility with versions of IBM Integration Bus (formerly known as WebSphere Message Broker) prior to version 7 that use this queue manager.

**MQPSM\_DISABLED**

The publish/subscribe engine and the queued publish/subscribe interface are not running. It is therefore not possible to publish or subscribe using the application programming interface. Any publish/subscribe messages that are put to the queues that are monitored by the queued publish/subscribe interface are not acted on.

**MQPSM\_ENABLED**

The publish/subscribe engine and the queued publish/subscribe interface are running. It is therefore possible to publish or subscribe by using the application programming interface and the queues that are monitored by the queued publish/subscribe interface. This value is the initial default value of the queue manager.

**PubSubNPInputMsg (MQCFIN)**

Whether to discard (or keep) an undelivered input message (parameter identifier: MQIA\_PUBSUB\_NP\_MSG).

The value can be:

**MQUNDELIVERED\_DISCARD**

Non-persistent input messages are discarded if they cannot be processed.

**MQUNDELIVERED\_KEEP**

Non-persistent input messages are not discarded if they cannot be processed. In this situation, the queued publish/subscribe interface continues to try the process again at appropriate intervals and does not continue processing subsequent messages.

**PubSubNPResponse (MQCFIN)**

Controls the behavior of undelivered response messages (parameter identifier: MQIA\_PUBSUB\_NP\_RESP).

The value can be:

**MQUNDELIVERED\_NORMAL**

Non-persistent responses that cannot be placed on the reply queue are put on the dead letter queue. If they cannot be placed on the dead letter queue they are discarded.

**MQUNDELIVERED\_SAFE**

Non-persistent responses that cannot be placed on the reply queue are put on the dead letter queue. If the response cannot be sent and cannot be placed on the dead letter queue the queued publish/subscribe interface rolls back the current operation. The operation is tried again at appropriate intervals and does not continue processing subsequent messages.

**MQUNDELIVERED\_DISCARD**

Non-persistent responses that are not placed on the reply queue are discarded.

**MQUNDELIVERED\_KEEP**

Non-persistent responses are not placed on the dead letter queue or discarded. Instead, the queued publish/subscribe interface backs out the current operation and then try it again at appropriate intervals.

**PubSubSyncPoint (MQCFIN)**

Whether only persistent (or all) messages must be processed under sync point (parameter identifier: MQIA\_PUBSUB\_SYNC\_PT).

The value can be:

**MQSYNCPOINT\_IFPER**

This value makes the queued publish/subscribe interface receive non-persistent messages outside sync point. If the interface receives a publication outside sync point, the interface forwards the publication to subscribers known to it outside sync point.

**MQSYNCPOINT\_YES**

This value makes the queued publish/subscribe interface receive all messages under sync point.

**QMGrDesc (MQCFST)**

Queue manager description (parameter identifier: MQCA\_Q\_MGR\_DESC).

This parameter is text that briefly describes the object.

The maximum length of the string is MQ\_Q\_MGR\_DESC\_LENGTH.

Use characters from the character set identified by the coded character set identifier (CCSID) for the queue manager on which the command is executing. Using this character set ensures that the text is translated correctly.

 **QSGCertificateLabel (MQCFST)**

Specifies the certificate label for the queue sharing group to use (parameter identifier: MQCA\_QSG\_CERT\_LABEL).

This parameter takes precedence over **CERTLABL** in the event that the queue manager is a member of a QSG.

**QueueAccounting (MQCFIN)**

Controls the collection of accounting (thread-level and queue-level accounting) data for queues (parameter identifier: MQIA\_ACCOUNTING\_Q). Note, that changes to this value are only effective for connections to the queue manager that occur after the change to the attribute.

The value can be:

**MQMON\_NONE**

Accounting data collection for queues is disabled. This value must not be overridden by the value of the **QueueAccounting** parameter on the queue.

**MQMON\_OFF**

Accounting data collection is disabled for queues specifying a value of MQMON\_Q\_MGR in the **QueueAccounting** parameter.

**MQMON\_ON**

Accounting data collection is enabled for queues specifying a value of MQMON\_Q\_MGR in the **QueueAccounting** parameter.

**QueueMonitoring (MQCFIN)**

Default setting for online monitoring for queues (parameter identifier: MQIA\_MONITORING\_Q).

If the **QueueMonitoring** queue attribute is set to MQMON\_Q\_MGR, this attribute specifies the value which is assumed by the channel. The value can be any of the following values:

**MQMON\_OFF**

Online monitoring data collection is turned off. This value is the initial default value of the queue manager.

**MQMON\_NONE**

Online monitoring data collection is turned off for queues regardless of the setting of their **QueueMonitoring** attribute.

**MQMON\_LOW**

Online monitoring data collection is turned on, with a low ratio of data collection.

**MQMON\_MEDIUM**

Online monitoring data collection is turned on, with a moderate ratio of data collection.

**MQMON\_HIGH**

Online monitoring data collection is turned on, with a high ratio of data collection.

**QueueStatistics (MQCFIN)**

Controls whether statistics data is to be collected for queues (parameter identifier: MQIA\_STATISTICS\_Q).

The value can be:

**Multi MQMON\_NONE**

Statistics data collection is turned off for queues regardless of the setting of their **QueueStatistics** parameter. This value is the initial default value of the queue manager.

**z/OS** This value is not applicable to z/OS

**MQMON\_OFF**

Statistics data collection is turned off for queues specifying a value of MQMON\_Q\_MGR in their **QueueStatistics** parameter.

**z/OS** This is the default value on z/OS.

**MQMON\_ON**

Statistics data collection is turned on for queues specifying a value of MQMON\_Q\_MGR in their **QueueStatistics** parameter.

**z/OS V 9.3.0** On z/OS systems, you must enable class 5 statistics using the START TRACE command.

**z/OS ReceiveTimeout (MQCFIN)**

How long a TCP/IP channel waits to receive data from its partner (parameter identifier: MQIA\_RECEIVE\_TIMEOUT).

The approximate length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state.

This parameter applies to z/OS only. It applies to message channels, and not to MQI channels. This number can be qualified as follows:

- This number is a multiplier to be applied to the negotiated *HeartBeatInterval* value to determine how long a channel is to wait. Set *ReceiveTimeoutType* to MQRCVTIME\_MULTIPLY. Specify a value of zero or in the range 2 - 99. If you specify zero, the channel waits indefinitely to receive data from its partner.
- This number is a value, in seconds, to be added to the negotiated *HeartBeatInterval* value to determine how long a channel is to wait. Set *ReceiveTimeoutType* to MQRCVTIME\_ADD. Specify a value in the range 1 - 999,999.
- This number is a value, in seconds, that the channel is to wait, set *ReceiveTimeoutType* to MQRCVTIME\_EQUAL. Specify a value in the range 0 - 999,999. If you specify 0, the channel waits indefinitely to receive data from its partner.

The initial default value of the queue manager is zero.

#### **z/OS ReceiveTimeoutMin (MQCFIN)**

The minimum length of time that a TCP/IP channel waits to receive data from its partner (parameter identifier: MQIA\_RECEIVE\_TIMEOUT\_MIN).

The minimum length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state. This parameter applies to z/OS only.

Specify a value in the range 0 - 999,999.

#### **z/OS ReceiveTimeoutType (MQCFIN)**

The qualifier to apply to *ReceiveTimeout* (parameter identifier: MQIA\_RECEIVE\_TIMEOUT\_TYPE).

The qualifier to apply to *ReceiveTimeoutType* to calculate how long a TCP/IP channel waits to receive data, including heartbeats, from its partner. It waits to receive data before returning to the inactive state. This parameter applies to z/OS only.

The value can be any of the following values:

##### **MQRCVTIME\_MULTIPLY**

The *ReceiveTimeout* value is a multiplier to be applied to the negotiated value of *HeartbeatInterval* to determine how long a channel waits. This value is the initial default value of the queue manager.

##### **MQRCVTIME\_ADD**

*ReceiveTimeout* is a value, in seconds, to be added to the negotiated value of *HeartbeatInterval* to determine how long a channel waits.

##### **MQRCVTIME\_EQUAL**

*ReceiveTimeout* is a value, in seconds, representing how long a channel waits.

#### **RemoteEvent (MQCFIN)**

Controls whether remote error events are generated (parameter identifier: MQIA\_REMOTE\_EVENT).

The value can be:

##### **MQEVR\_DISABLED**

Event reporting disabled.

##### **MQEVR\_ENABLED**

Event reporting enabled.

#### **RepositoryName (MQCFST)**

Cluster name (parameter identifier: MQCA\_REPOSITORY\_NAME).

The name of a cluster for which this queue manager provides a repository manager service.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

No more than one of the resultant values of *RepositoryName* can be nonblank.

#### **RepositoryNameList (MQCFST)**

Repository namelist (parameter identifier: MQCA\_REPOSITORY\_NAMELIST).

The name, of a namelist of clusters, for which this queue manager provides a repository manager service.

This queue manager does not have a full repository, but can be a client of other repository services that are defined in the cluster, if

- Both *RepositoryName* and *RepositoryNameList* are blank, or
- *RepositoryName* is blank and the namelist specified by *RepositoryNameList* is empty.

No more than one of the resultant values of *RepositoryNameList* can be nonblank.

#### **RevDns (MQCFIN)**

Whether reverse lookup of the host name from a Domain Name Server is carried out. (parameter identifier: MQIA\_REVERSE\_DNS\_LOOKUP).

This attribute has an effect only on channels using a transport type (TRPTYPE) of TCP.

The value can be:

**MQRDNS\_DISABLED**

DNS host names are not reverse looked-up for the IP addresses of inbound channels. With this setting any CHLAUTH rules using host names are not matched.

**MQRDNS\_ENABLED**

DNS host names are reverse looked-up for the IP addresses of inbound channels when this information is required. This setting is required for matching against CHLAUTH rules that contain host names, and for writing out error messages.

**z/OS SecurityCase (MQCFIN)**

Security case supported (parameter identifier: MQIA\_SECURITY\_CASE).

Specifies whether the queue manager supports security profile names in mixed case, or in uppercase only. The value is activated when a Refresh Security command is run with *SecurityType* (MQSECTYPE\_CLASSES) specified. This parameter is valid only on z/OS.

The value can be:

**MQSCYC\_UPPER**

Security profile names must be in uppercase.

**MQSCYC\_MIXED**

Security profile names can be in uppercase or in mixed case.

**z/OS SharedQMgrName (MQCFIN)**

Shared-queue queue manager name (parameter identifier: MQIA\_SHARED\_Q\_Q\_MGR\_NAME).

A queue manager makes an MQOPEN call for a shared queue. The queue manager that is specified in the **ObjectQmgrName** parameter of the MQOPEN call is in the same queue sharing group as the processing queue manager. The SQQMNAME attribute specifies whether the **ObjectQmgrName** is used or whether the processing queue manager opens the shared queue directly. This parameter is valid only on z/OS.

The value can be any of the following values:

**MQSQQM\_USE**

*ObjectQmgrName* is used and the appropriate transmission queue is opened.

**MQSQQM\_IGNORE**

The processing queue manager opens the shared queue directly. This value can reduce the traffic in your queue manager network.

**SSLCRLNameList (MQCFST)**

The TLS namelist (parameter identifier: MQCA\_SSL\_CRL\_NAMELIST).

The length of the string is MQ\_NAMELIST\_NAME\_LENGTH.

Indicates the name of a namelist of authentication information objects which are used to provide certificate revocation locations to allow enhanced TLS certificate checking.

If *SSLCRLNameList* is blank, certificate revocation checking is not invoked.

Changes to *SSLCRLNameList*, or to the names in a previously specified namelist, or to previously referenced authentication information objects become effective:

- **Multi** On Multiplatforms, when a new channel process is started.
- **Multi** For channels that run as threads of the channel initiator on Multiplatforms, when the channel initiator is restarted.
- **Multi** For channels that run as threads of the listener on Multiplatforms, when the listener is restarted.

- **z/OS** On z/OS, when the channel initiator is restarted.
- When a **REFRESH SECURITY TYPE(SSL)** command is issued.
- **IBM i** On IBM i queue managers, this parameter is ignored. However, it is used to determine which authentication information objects are written to the AMQCLCHL . TAB file.

Only authentication information objects with types of CRLLDAP or OCSP are allowed in the namelist referred to by *SSLCRLNamelist* (MQCFST). Any other type results in an error message when the list is processed and is subsequently ignored.

### **SSLCryptoHardware (MQCFST)**

The TLS cryptographic hardware (parameter identifier: MQCA\_SSL\_CRYPTO\_HARDWARE).

The length of the string is MQ\_SSL\_CRYPTO\_HARDWARE\_LENGTH.

Sets the name of the parameter string required to configure the cryptographic hardware present on the system.

This parameter is valid only on AIX, Linux, and Windows.

All supported cryptographic hardware supports the PKCS #11 interface. Specify a string of the following format:

```
GSK_PKCS11=PKCS_#11_driver_path_and_file_name;PKCS_#11_token_label;PKCS_#11_token_password;symmetric_cipher_setting;
```

The PKCS #11 driver path is an absolute path to the shared library providing support for the PKCS #11 card. The PKCS #11 driver file name is the name of the shared library. An example of the value required for the PKCS #11 driver path and file name is `/usr/lib/pkcs11/PKCS11_API.so`

To access symmetric cipher operations through GSKit, specify the symmetric cipher setting parameter. The value of this parameter is either:

#### **SYMMETRIC\_CIPHER\_OFF**

Do not access symmetric cipher operations.

#### **SYMMETRIC\_CIPHER\_ON**

Access symmetric cipher operations.

If the symmetric cipher setting is not specified, this value has the same effect as specifying SYMMETRIC\_CIPHER\_OFF.

The maximum length of the string is 256 characters. The default value is blank.

If you specify a string in the wrong format, you get an error.

When the *SSLCryptoHardware* (MQCFST) value is changed, the cryptographic hardware parameters specified become the ones used for new TLS connection environments. The new information becomes effective:

- When a new channel process is started.
- For channels that run as threads of the channel initiator, when the channel initiator is restarted.
- For channels that run as threads of the listener, when the listener is restarted.
- When a Refresh Security command is issued to refresh the contents of the TLS key repository.

### **SSLEvent (MQCFIN)**

Controls whether TLS events are generated (parameter identifier: MQIA\_SSL\_EVENT).

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.



### SSLFipsRequired (MQCFIN)

SSLFIPS specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM MQ, rather than in cryptographic hardware (parameter identifier: MQIA\_SSL\_FIPS\_REQUIRED).

If cryptographic hardware is configured, the cryptographic modules used are those modules provided by the hardware product. These modules might, or might not, be FIPS-certified to a particular level depending on the hardware product in use. This parameter applies to z/OS, AIX, Linux, and Windows platforms only.

The value can be any of the following values:

#### MQSSL\_FIPS\_NO

IBM MQ provides an implementation of TLS cryptography which supplies some FIPS-certified modules on some platforms. If you set *SSLFIPSRequired* to MQSSL\_FIPS\_NO, any CipherSpec supported on a particular platform can be used. This value is the initial default value of the queue manager.

If the queue manager runs without using cryptographic hardware, refer to the CipherSpecs listed in [Specifying CipherSpecs](#) employing FIPS 140-2 certified cryptography:

#### MQSSL\_FIPS\_YES

Specifies that only FIPS-certified algorithms are to be used in the CipherSpecs allowed on all TLS connections from and to this queue manager.

For a listing of appropriate FIPS 140-2 certified CipherSpecs; see [Specifying CipherSpecs](#).

Changes to SSLFIPS become effective either:

- On AIX, Linux, and Windows, when a new channel process is started.
- For channels that run as threads of the channel initiator on AIX, Linux, and Windows, when the channel initiator is restarted.
- For channels that run as threads of the listener on AIX, Linux, and Windows, when the listener is restarted.
- For channels that run as threads of a process pooling process, when the process pooling process is started or restarted and first runs a TLS channel. If the process pooling process has already run a TLS channel, and you want the change to become effective immediately, run the MQSC command **REFRESH SECURITY TYPE(SSL)**. The process pooling process is **amqzmpa** on AIX, Linux, and Windows.
- On z/OS, when the channel initiator is restarted.
- When a **REFRESH SECURITY TYPE(SSL)** command is issued, except on z/OS.


### SSLKeyRepository (MQCFST)

The TLS key repository (parameter identifier: MQCA\_SSL\_KEY\_REPOSITORY).

The length of the string is MQ\_SSL\_KEY\_REPOSITORY\_LENGTH.

Indicates the name of the Secure Sockets Layer key repository.

The format of the name depends on the environment:

- On z/OS, it is the name of a key ring.
-  On IBM i, it is of the form *pathname/keyfile.kdb*, where *keyfile* identifies a GSKit CMS key database file. If the file suffix is not specified, it is assumed to be *.kdb*.

The default value is `/QIBM/UserData/ICSS/Cert/Server/Default`.

If you specify `*SYSTEM`, IBM MQ uses the system certificate store as the key repository for the queue manager. As a result, the queue manager is registered as a server application in Digital Certificate Manager (DCM). You can assign any server/client certificate in the system store to this application.

If you change the `SSLKeyRepository` parameter to a value other than `*SYSTEM`, IBM MQ unregisters the queue manager as an application with DCM.

- **V9.3.0** **V9.3.0** On AIX and Linux, it is of the form `pathname/keyfile` and on Windows `pathname\keyfile`, where `keyfile` identifies a GSKit CMS or PKCS#12 key database file. If the file suffix is not specified, it is assumed to be `.kdb`.

The default value for AIX and Linux is `/var/mqm/qmgrs/QMGR/ssl/key`, and on Windows it is `C:\Program Files\IBM\MQ\qmgrs\QMGR\ssl\key`, where `QMGR` is replaced by the queue manager name.

**V9.3.0** **V9.3.0** If TLS AMQP channels are used, the suffix of the key repository file must be one of the following:

- `.kdb`, for a CMS key repository
- `.p12` or `.pkcs12`, for a PKCS #12 key repository.

**Multi** On Multiplatforms, the syntax of this parameter is validated to ensure that it contains a valid, absolute, directory path.

If `SSLKeyRepository` is blank, or is a value that does not correspond to a key ring or key database file, channels using TLS fail to start.

Changes to `SSLKeyRepository` become effective as follows:

- **Multi** On Multiplatforms:
  - when a new channel process is started
  - for channels that run as threads of the channel initiator, when the channel initiator is restarted.
  - for channels that run as threads of the listener, when the listener is restarted.
- **z/OS** On z/OS, when the channel initiator is restarted.

### **V9.3.0** **ALW** **V9.3.0** **SSLKeyRepositoryPassword (MQCFST)**

The password to access the TLS key repository (parameter identifier: `MQCA_SSL_KEY_REPO_PASSWORD`).

The length of the string is `MQ_SSL_KEY_REPO_PWD_LEN`.

If you specify a value for this attribute, it is used as the password to access the Secure Sockets Layer key repository. If this attribute is blank, the stash file that is associated with the key repository is used.



**Attention:** If the stash file is not present or cannot be read, the key repository cannot be accessed and channels using TLS fail to start.

You should set **InitialKey** to a unique value for the queue manager before this attribute is set. The default value is blank.

### **SSLKeyResetCount (MQCFIN)**

SSL key reset count (parameter identifier: `MQIA_SSL_RESET_COUNT`).

Specifies when TLS channel MCAs that initiate communication reset the secret key used for encryption on the channel. The value of this parameter represents the total number of unencrypted bytes that are sent and received on the channel before the secret key is renegotiated. This number of bytes includes control information sent by the MCA.

The secret key is renegotiated when (whichever occurs first):

- The total number of unencrypted bytes sent and received by the initiating channel MCA exceeds the specified value, or,
- If channel heartbeats are enabled, before data is sent or received following a channel heartbeat.

Specify a value in the range 0 - 999,999,999. A value of zero, the initial default value of the queue manager, signifies that secret keys are never renegotiated. If you specify a TLS secret key reset count between 1 byte through 32 KB, TLS channels use a secret key reset count of 32Kb. This count is to avoid the performance effect of excessive key resets which would occur for small TLS secret key reset values.

### **SSLTasks (MQCFIN)**

Number of server subtasks to use for processing TLS calls (parameter identifier: MQIA\_SSL\_TASKS). This parameter applies to z/OS only.

The number of server subtasks to use for processing TLS calls. To use TLS channels, you must have at least two of these tasks running.

Specify a value in the range 0 - 9999. However, to avoid problems with storage allocation, do not set this parameter to a value greater than 50.

### **StartStopEvent (MQCFIN)**

Controls whether start and stop events are generated (parameter identifier: MQIA\_START\_STOP\_EVENT).

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

### **Multi StatisticsInterval (MQCFIN)**

The time interval, in seconds, at which statistics monitoring data is written to the monitoring queue (parameter identifier: MQIA\_STATISTICS\_INTERVAL).

Specify a value in the range 1 - 604,000.

This parameter is valid only on [Multiplatforms](#).

### **z/OS TCPChannels (MQCFIN)**

The maximum number of channels that can be current, or clients that can be connected, that use the TCP/IP transmission protocol (parameter identifier: MQIA\_TCP\_CHANNELS).

Specify a value in the range 0 - 9999. The initial default value of the queue manager is 200.

Sharing conversations do not contribute to the total for this parameter.

This parameter applies to z/OS only.

### **z/OS TCPKeepAlive (MQCFIN)**

Specifies whether the TCP KEEPALIVE facility is to be used to check whether the other end of a connection is still available (parameter identifier: MQIA\_TCP\_KEEP\_ALIVE).

The value can be:

#### **MQTCPKEEP\_YES**

The TCP KEEPALIVE facility is to be used as specified in the TCP profile configuration data set. The interval is specified in the *KeepAliveInterval* channel attribute.

#### **MQTCPKEEP\_NO**

The TCP KEEPALIVE facility is not to be used. This value is the initial default value of the queue manager.

This parameter applies only to z/OS.

### **z/OS TCPName (MQCFST)**

The name of the TCP/IP system that you are using (parameter identifier: MQIA\_TCP\_NAME).

The maximum length of the string is MQ\_TCP\_NAME\_LENGTH.

This parameter applies only to z/OS.

### **z/OS TCPStackType (MQCFIN)**

Specifies whether the channel initiator can use only the TCP/IP address space specified in *TCPName*, or can optionally bind to any selected TCP/IP address (parameter identifier: MQIA\_TCP\_STACK\_TYPE).

The value can be:

#### **MQTCPSTACK\_SINGLE**

The channel initiator uses the TCP/IP address space that is specified in *TCPName*. This value is the initial default value of the queue manager.

#### **MQTCPSTACK\_MULTIPLE**

The channel initiator can use any TCP/IP address space available to it. It defaults to the one specified in *TCPName* if no other is specified for a channel or listener.

This parameter applies only to z/OS.

### **TraceRouteRecording (MQCFIN)**

Specifies whether trace-route information can be recorded and a reply message generated (parameter identifier: MQIA\_TRACE\_ROUTE\_RECORDING).

The value can be:

#### **MQRECORDING\_DISABLED**

Trace-route information cannot be recorded.

#### **MQRECORDING\_MSG**

Trace-route information can be recorded and replies sent to the destination specified by the originator of the message causing the trace-route record.

#### **MQRECORDING\_Q**

Trace-route information can be recorded and replies sent to SYSTEM.ADMIN.TRACE.ROUTE.QUEUE.

If participation in route tracing is enabled using this queue manager attribute, the value of the attribute is only important if a reply is generated. Route tracing is enabled by not setting *TraceRouteRecording* to MQRECORDING\_DISABLED. The reply must go either to SYSTEM.ADMIN.TRACE.ROUTE.QUEUE, or to the destination specified by the message itself. Provided the attribute is not disabled then messages not yet at the final destination might have information added to them. For more information about trace-route records, see [Controlling trace-route messaging](#).

### **TreeLifeTime (MQCFIN)**

The lifetime, in seconds, of non-administrative topics (parameter identifier: MQIA\_TREE\_LIFE\_TIME).

Non-administrative topics are those topics created when an application publishes to, or subscribes as, a topic string that does not exist as an administrative node. When this non-administrative node no longer has any active subscriptions, this parameter determines how long the queue manager waits before removing that node. Only non-administrative topics that are in use by a durable subscription remain after the queue manager is recycled.

Specify a value in the range 0 - 604,000. A value of 0 means that non-administrative topics are not removed by the queue manager. The initial default value of the queue manager is 1800.

### **TriggerInterval (MQCFIN)**

Trigger interval (parameter identifier: MQIA\_TRIGGER\_INTERVAL).

Specifies the trigger time interval, expressed in milliseconds, for use only with queues where *TriggerType* has a value of MQTT\_FIRST.

In this case, trigger messages are normally generated only when a suitable message arrives on the queue, and the queue was previously empty. Under certain circumstances, however, an additional trigger message can be generated with MQTT\_FIRST triggering, even if the queue was not empty.

These additional trigger messages are not generated more often than every *TriggerInterval* milliseconds.

Specify a value in the range 0 - 999,999,999.

## Error codes (Change Queue Manager)

This command might return the following errors in the response format header, in addition to the values shown on page “[Error codes applicable to all commands](#)” on page 990.

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRCCF\_CERT\_LABEL\_NOT\_ALLOWED**

Certificate label error.

#### **MQRCCF\_CHAD\_ERROR**

Channel automatic definition error.

#### **MQRCCF\_CHAD\_EVENT\_ERROR**

Channel automatic definition event error.

#### **MQRCCF\_CHAD\_EVENT\_WRONG\_TYPE**

Channel automatic definition event parameter not allowed for this channel type.

#### **MQRCCF\_CHAD\_EXIT\_ERROR**

Channel automatic definition exit name error.

#### **MQRCCF\_CHAD\_EXIT\_WRONG\_TYPE**

Channel automatic definition exit parameter not allowed for this channel type.

#### **MQRCCF\_CHAD\_WRONG\_TYPE**

Channel automatic definition parameter not allowed for this channel type.

#### **MQRCCF\_FORCE\_VALUE\_ERROR**

Force value not valid.

#### **MQRCCF\_PATH\_NOT\_VALID**

Path not valid.

#### **MQRCCF\_PWD\_LENGTH\_ERROR**

Password length error.

#### **MQRCCF\_PSCLUS\_DISABLED\_TOPDEF**

Administrator or application attempted to define a cluster topic when **PubSubClub** is set to MQPSCLUS\_DISABLED.

#### **MQRCCF\_PSCLUS\_TOPIC\_EXSITS**

Administrator tried to set **PubSubClub** to MQPSCLUS\_DISABLED when a cluster topic definition exists.

#### **MQRCCF\_Q\_MGR\_ATTR\_CONFLICT**

Queue manager attribute error. A possible cause is that you attempted to specify SSLKEYR(\*SYSTEM) with a nonblank queue manager CERTLABL.

#### **MQRCCF\_Q\_MGR\_CCSID\_ERROR**

Coded character set value not valid.

#### **MQRCCF\_REPOS\_NAME\_CONFLICT**

Repository names not valid.

#### **MQRCCF\_UNKNOWN\_Q\_MGR**

Queue manager not known.

#### **MQRCCF\_WRONG\_CHANNEL\_TYPE**

Channel type error.

### Related concepts

[Channel states](#)

## Related tasks

Specifying that only FIPS-certified CipherSpecs are used at run time on the MQI client

## Related reference

[Federal Information Processing Standards \(FIPS\) for AIX, Linux, and Windows](#)

## **MQCMD\_CHANGE\_SECURITY (Change Security) on z/OS**

The Change Security (MQCMD\_CHANGE\_SECURITY) PCF command changes specified attributes of an existing security definition.

## Required parameters

None

## Optional parameters

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### SecurityInterval (MQCFIN)

Timeout check interval (parameter identifier: MQIACF\_SECURITY\_INTERVAL).

Specifies the interval between checks for user IDs and associated resources to determine whether the *SecurityTimeout* has occurred. The value specifies a number of minutes in the range zero through 10080 (one week). If *SecurityInterval* is specified as zero, no user timeouts occur. If *SecurityInterval* is specified as nonzero, the user ID times out at a time between *SecurityTimeout* and *SecurityTimeout* plus *SecurityInterval*.

### SecurityTimeout (MQCFIN)

Security information timeout (parameter identifier: MQIACF\_SECURITY\_TIMEOUT).

Specifies how long security information about an unused user ID and associated resources is retained by IBM MQ. The value specifies a number of minutes in the range zero through 10080 (one week). If *SecurityTimeout* is specified as zero, and *SecurityInterval* is nonzero, all such information is discarded by the queue manager every *SecurityInterval* number of minutes.

## **MQCMD\_CHANGE\_SMDS (Change SMDS) on z/OS**

The Change SMDS (MQCMD\_CHANGE\_SMDS) PCF command changes the current shared message data set options for the specified queue manager and CF structure.

### SMDS (MQCFST)

Specifies the queue manager for which the shared message data set properties are to be changed, or an asterisk to change the properties for all shared message data sets associated with the specified CFSTRUCT.

### CFStrucName (MQCFST)

The name of the CF application structure with SMDS parameters that you want to change (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

## Optional parameters

### DSBufs (MQCFIN)

The shared message data set buffers group (parameter identifier: MQIA\_CF\_SMDS\_BUFFERS).

Specifies the number of buffers to be allocated in each queue manager for accessing shared message data sets. The size of each buffer is equal to the logical block size.

A value in the range 1 - 9999 or MQDSB\_DEFAULT.

When DEFAULT is used any previous value is overridden and the DSBUFS value from the CFSTRUCT definition is used. The size of each buffer is equal to the logical block size.

Value can not be set unless CFLEVEL(5) is defined.

### DSEXPAND (MQCFIN)

The shared message data set expand option (parameter identifier: MQIACF\_CF\_SMDS\_EXPAND).

Specifies whether or not the queue manager should expand a shared message data set when it is nearly full, and further blocks are required in the data set. The value can be any of the following values:

#### MQDSE\_YES

The data set can be expanded.

#### MQDSE\_NO

The data set cannot be expanded.

#### MQDSE\_DEFAULT

Only returned on DISPLAY CFSTRUCT when not explicitly set

Value can not be set unless CFLEVEL(5) is defined.

### MQCMD\_CLEAR\_Q (Clear Queue)

The Clear Queue (MQCMD\_CLEAR\_Q) PCF command deletes all the messages from a local queue.

The command fails if the queue contains uncommitted messages.

## Required parameters

### QName (MQCFST)

Queue name (parameter identifier: MQCA\_Q\_NAME).

The name of the local queue to be cleared. The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**Note:** The target queue must be type local.

## Optional parameters



### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.

- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_PRIVATE**

Clear the private queue named in *QName*. The queue is private if it was created using a command with the attributes MQQSGD\_PRIVATE or MQQSGD\_Q\_MGR. This value is the default value.

#### **MQQSGD\_SHARED**

Clear the shared queue named in *QName*. The queue is shared if it was created using a command with the attribute MQQSGD\_SHARED. This value applies only to local queues.

## **Error codes**

This command might return the following error codes in the response format header, in addition to the values shown on page [“Error codes applicable to all commands” on page 990](#).

### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRC\_Q\_NOT\_EMPTY**

(2055, X'807') Queue contains one or more messages or uncommitted put or get requests.

This reason occurs only if there are uncommitted updates.

#### **MQRCCF\_Q\_WRONG\_TYPE**

Action not valid for the queue of specified type.

### **MQCMD\_CLEAR\_TOPIC\_STRING (Clear Topic String)**

The Clear Topic String (MQCMD\_CLEAR\_TOPIC\_STRING) PCF command clears the retained message which is stored for the specified topic.

## **Required parameters**

### **TopicString (MQCFST)**

Topic String (parameter identifier: MQCA\_TOPIC\_STRING).

The topic string to be cleared The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

### **ClearType (MQCFIN)**

Clear type (parameter identifier: MQIACF\_CLEAR\_TYPE).

Specifies the type of clear command being issued. The value must be:

MQCLRT\_RETAINED Remove the retained publication from the specified topic string.

## **Optional parameters**

### **Scope (MQCFIN)**

Scope of clearance (parameter identifier: MQIACF\_CLEAR\_SCOPE).

Whether the topic string is to be cleared locally or globally. The value can be:



## **MQCLRS\_LOCAL**

The retained message is removed from the specified topic string at the local queue manager only.

**z/OS**

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **MQCMD\_DELETE\_AUTH\_INFO (Delete Authentication Information Object)**

The Delete authentication information (MQCMD\_DELETE\_AUTH\_INFO) PCF command deletes the specified authentication information object.

## **Required parameters**

### **AuthInfoName (MQCFST)**

Authentication information object name (parameter identifier: MQCA\_AUTH\_INFO\_NAME).

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH.

## **Optional parameters for z/OS**

**z/OS**

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

### **MQQSGD\_COPY**

The object definition resides on the page set of the queue manager which executes this command. The object was defined by a command using the parameter MQQSGD\_COPY. Any object in the shared repository, or any object defined by a command using the parameter MQQSGD\_Q\_MGR, is not affected by this command.

### **MQQSGD\_GROUP**

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD\_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE AUTHINFO(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

### **MQQSGD\_Q\_MGR**

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_Q\_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

MQQSGD\_Q\_MGR is the default value.

## **Optional parameters for AIX, Linux, and Windows**

**ALW**

**V 9.3.0**

### **IgnoreState (MQCFST)**

Specifies whether the command fails if the object does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### **MQIS\_YES**

The command succeeds regardless of whether the object exists.

#### **MQIS\_NO**

The command fails if the object does not exist. This is the default value.

**Multi**

### **MQCMD\_DELETE\_AUTH\_REC (Delete Authority Record) on Multiplatforms**

The Delete Authority Record (MQCMD\_DELETE\_AUTH\_REC) PCF command deletes an authority record. The authorizations associated with the profile no longer apply to IBM MQ objects with names that match the profile name specified.

## **Required parameters**

### **ObjectType (MQCFIN)**

The type of object for which to delete authorizations (parameter identifier: MQIACF\_OBJECT\_TYPE).

The value can be any of the following values:

#### **MQOT\_AUTH\_INFO**

Authentication information.

#### **MQOT\_CHANNEL**

Channel object.

#### **MQOT\_CLNTCONN\_CHANNEL**

Client-connection channel object.

#### **MQOT\_COMM\_INFO**

Communication information object

**MQOT\_LISTENER**

Listener object.

**MQOT\_NAMELIST**

Namelist.

**MQOT\_PROCESS**

Process.

**MQOT\_Q**

Queue, or queues, that match the object name parameter.

**MQOT\_Q\_MGR**

Queue manager.

**MQOT\_REMOTE\_Q\_MGR\_NAME**

Remote queue manager.

**MQOT\_SERVICE**

Service object.

**MQOT\_TOPIC**

Topic object.

**ProfileName (MQCFST)**

Name of the profile to be deleted (parameter identifier: MQCACF\_AUTH\_PROFILE\_NAME).

If you have defined a generic profile then you can specify it here, using wildcard characters to specify a named generic profile to be removed. If you specify an explicit profile name, the object must exist.

The maximum length of the string is MQ\_AUTH\_PROFILE\_NAME\_LENGTH.

**Optional parameters****GroupNames (MQCFSL)**

Group names (parameter identifier: MQCACF\_GROUP\_ENTITY\_NAMES).

The names of groups having a profile deleted. At least one group name or principal name must be specified. An error occurs if neither are specified.

Each member in this list can be a maximum length of MQ\_ENTITY\_NAME\_LENGTH.

**PrincipalNames (MQCFSL)**

Principal names (parameter identifier: MQCACF\_PRINCIPAL\_ENTITY\_NAMES).

The names of principals having a profile deleted. At least one group name or principal name must be specified. An error occurs if neither are specified.

Each member in this list can be a maximum length of MQ\_ENTITY\_NAME\_LENGTH.

**V9.3.0 ALW IgnoreState (MQCFST)**

Specifies whether the command fails if the object does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

**MQIS\_YES**

The command succeeds regardless of whether the authority record exists.

This is the default value for objects of type QUEUE, QMGR, RQMNAME, and TOPIC.

**MQIS\_NO**

The command fails if the authority record does not exist.

This value is not valid for objects of type QUEUE, QMGR, RQMNAME, and TOPIC. This is the default value for all other object types.

**Error codes (Delete Authority Record)**

This command might return the following error codes in the response format header, in addition to the values shown on page [“Error codes applicable to all commands”](#) on page 990.

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRC\_OBJECT\_TYPE\_ERROR**

Invalid object type.

#### **MQRC\_UNKNOWN\_ENTITY**

Userid not authorized, or unknown.

#### **MQRCCF\_ENTITY\_NAME\_MISSING**

Entity name missing.

#### **MQRCCF\_OBJECT\_TYPE\_MISSING**

Object type missing.

#### **MQRCCF\_PROFILE\_NAME\_ERROR**

Invalid profile name.

### **MQCMD\_DELETE\_CF\_STRUC (Delete CF Structure) on z/OS**

The Delete CF Structure (MQCMD\_DELETE\_CF\_STRUC) PCF command deletes an existing CF application structure definition.

**Note:** This command is supported only on z/OS when the queue manager is a member of a queue sharing group.

### Required parameters

#### **CFStrucName (MQCFST)**

CF structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The CF application structure definition to be deleted. The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

#### **MQCMD\_DELETE\_CHANNEL (Delete Channel)**

The Delete Channel (MQCMD\_DELETE\_CHANNEL) PCF command deletes the specified channel definition.

### Required parameters

#### **ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The name of the channel definition to be deleted. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

### Optional parameters

None of the following attributes are applicable to MQTT channels unless specifically mentioned in the parameter description.

#### **ChannelType (MQCFIN)**

The type of channel (parameter identifier: MQIACH\_CHANNEL\_TYPE). This parameter is currently only used with MQTT Telemetry channels, and is required when deleting a Telemetry channel. The only value that can currently be given to the parameter is **MQCHT\_MQTT**.

#### **ChannelTable (MQCFIN)**

Channel table (parameter identifier: MQIACH\_CHANNEL\_TABLE).

Specifies the ownership of the channel definition table that contains the specified channel definition.

The value can be any of the following values:

#### **MQCHTAB\_Q\_MGR**

Queue manager table.

MQCHTAB\_Q\_MGR is the default. This table contains channel definitions for channels of all types except MQCHT\_CLNTCONN.

### **MQCHTAB\_CLNTCONN**

Client-connection table.

This table only contains channel definitions for channels of type MQCHT\_CLNTCONN.

This parameter is not applicable to MQ Telemetry.

### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **z/OS QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_COPY**

The object definition resides on the page set of the queue manager that executes the command.

The object was defined by a command using the parameter MQQSGD\_COPY. Any object residing in the shared repository, or any object defined by a command using the parameter MQQSGD\_Q\_MGR, is not affected by this command.

#### **MQQSGD\_GROUP**

The object definition resides in the shared repository. The object was defined by a command using the parameters MQQSGD\_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE CHANNEL(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

#### **MQQSGD\_Q\_MGR**

The object definition resides on the page set of the queue manager that executes the command.

The object was defined by a command using the parameter MQQSGD\_Q\_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

MQQSGD\_Q\_MGR is the default value.

**IgnoreState (MQCFST)**

Specifies whether the command fails if the channel does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

**MQIS\_YES**

The command succeeds regardless of whether the channel exists.

**MQIS\_NO**

The command fails if the channel does not exist. This is the default value.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in topic [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

**MQRCCF\_CHANNEL\_TABLE\_ERROR**

Channel table value not valid.

**MQCMD\_DELETE\_CHANNEL (delete channel) MQTT on AIX, Linux, and****Windows**

The Delete Telemetry Channel (MQCMD\_DELETE\_CHANNEL) PCF command deletes the specified channel definition.

**Required parameters****ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The name of the channel definition to be deleted. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**ChannelType (MQCFIN)**

The type of channel (parameter identifier: MQIACH\_CHANNEL\_TYPE). Required when deleting a Telemetry channel. The only value that can currently be given to the parameter is **MQCHT\_MQTT**.

**Error codes**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

**Multi** **MQCMD\_DELETE\_COMM\_INFO (Delete Communication Information Object) on Multiplatforms**

The Delete Communication Information Object (MQCMD\_DELETE\_COMM\_INFO) PCF command deletes the specified communication information object.

### Required parameter

#### ComminfoName (MQCFST)

The name of the communication information definition to be deleted (parameter identifier: MQCA\_COMM\_INFO\_NAME).

### Optional parameters

**ALW**

**V 9.3.0** **IgnoreState (MQCFST)**

Specifies whether the command fails if the object does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### MQIS\_YES

The command succeeds regardless of whether the object exists.

#### MQIS\_NO

The command fails if the object does not exist. This is the default value.

**Multi** **MQCMD\_DELETE\_LISTENER (Delete Channel Listener) on Multiplatforms**

The Delete Channel Listener (MQCMD\_DELETE\_LISTENER) PCF command deletes an existing channel listener definition.

### Required parameters

#### ListenerName (MQCFST)

Listener name (parameter identifier: MQCACH\_LISTENER\_NAME).

This parameter is the name of the listener definition to be deleted. The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

**V 9.3.0** **ALW** **IgnoreState (MQCFST)**

Specifies whether the command fails if the listener does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### MQIS\_YES

The command succeeds regardless of whether the listener exists.

#### MQIS\_NO

The command fails if the listener does not exist. This is the default value.

### MQCMD\_DELETE\_NAMELIST (Delete Namelist)

The Delete Namelist (MQCMD\_DELETE\_NAMELIST) PCF command deletes an existing namelist definition.

### Required parameters

#### NamelistName (MQCFST)

Namelist name (parameter identifier: MQCA\_NAMELIST\_NAME).

This parameter is the name of the namelist definition to be deleted. The maximum length of the string is MQ\_NAMELIST\_NAME\_LENGTH.

### Optional parameters for z/OS

**z/OS**

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_COPY**

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD\_Q\_MGR, is not affected by this command.

#### **MQQSGD\_GROUP**

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD\_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE NAMELIST(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

#### **MQQSGD\_Q\_MGR**

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_Q\_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

MQQSGD\_Q\_MGR is the default value.

## **Optional parameters for AIX, Linux, and Windows**

**ALW**

**V 9.3.0**

### **IgnoreState (MQCFST)**

Specifies whether the command fails if the namelist does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### **MQIS\_YES**

The command succeeds regardless of whether the namelist exists.



## **MQIS\_NO**

The command fails if the namelist does not exist. This is the default value.

## **MQCMD\_DELETE\_PROCESS (Delete Process)**

The Delete Process (MQCMD\_DELETE\_PROCESS) PCF command deletes an existing process definition.

### **Required parameters**

#### **ProcessName (MQCFST)**

Process name (parameter identifier: MQCA\_PROCESS\_NAME).

The process definition to be deleted. The maximum length of the string is MQ\_PROCESS\_NAME\_LENGTH.

### **Optional parameters for z/OS**



#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

#### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_COPY**

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD\_Q\_MGR, is not affected by this command.

#### **MQQSGD\_GROUP**

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD\_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE PROCESS(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

## **MQQSGD\_Q\_MGR**

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_Q\_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

MQQSGD\_Q\_MGR is the default value.

## **Optional parameters for AIX, Linux, and Windows**

**ALW**

### **V 9.3.0 IgnoreState (MQCFST)**

Specifies whether the command fails if the process definition does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### **MQIS\_YES**

The command succeeds regardless of whether the process definition exists.

#### **MQIS\_NO**

The command fails if the process definition does not exist. This is the default value.

**Multi**

## **MQCMD\_DELETE\_PROT\_POLICY (delete security policy) on Multiplatforms**

The Delete Policy (MQCMD\_DELETE\_PROT\_POLICY) PCF command deletes a security policy.

## **Required parameters**

### **Policy-name (MQCFST)**

The name of the security policy to be deleted (parameter identifier: MQCA\_POLICY\_NAME).

The name of the policy, or policies, to delete are the same as the name of the queue, or queues, that the policies control.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

## **Optional parameters**

### **V 9.3.0 ALW IgnoreState (MQCFST)**

Specifies whether the command fails if the policy does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### **MQIS\_YES**

The command succeeds regardless of whether the policy exists.

#### **MQIS\_NO**

The command fails if the policy does not exist. This is the default value.

## **Error codes (Delete Security Policy)**

This command might return the following error codes in the response format header, in addition to the values shown on page [“Error codes applicable to all commands” on page 990](#).

### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRC\_OBJECT\_TYPE\_ERROR**

Invalid object type.

#### **MQRCCF\_POLICY\_NAME\_ERROR**

Invalid policy name.

## **MQCMD\_DELETE\_Q (Delete Queue)**

The Delete Queue (MQCMD\_DELETE\_Q) PCF command deletes a queue.

### **Required parameters**

#### **QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

The name of the queue to be deleted.

If the **Scope** attribute of the queue is MQSCO\_CELL, the entry for the queue is deleted from the cell directory.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **Optional parameters**

#### **Authrec (MQCFIN)**

Authrec (parameter identifier: MQIACF\_REMOVE\_AUTHREC).

Specifies whether the associated authority record is also deleted.

This parameter does not apply to z/OS.

The value can be any of the following values:

#### **MQRAR\_YES**

The authority record associated with the object is deleted. This is the default.

#### **MQRAR\_NO**

The authority record associated with the object is not deleted.



#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

#### **Purge (MQCFIN)**

Purge queue (parameter identifier: MQIACF\_PURGE).

If there are messages on the queue MQPO\_YES must be specified, otherwise the command fails. If this parameter is not present the queue is not purged.

Valid only for queue of type local.

The value can be any of the following values:

#### **MQPO\_YES**

Purge the queue.

#### **MQPO\_NO**

Do not purge the queue.

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_COPY**

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD\_Q\_MGR, is not affected by this command.

#### **MQQSGD\_GROUP**

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD\_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the deletion is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE queue(q-name) QSGDISP(COPY)
```

or, for a local queue only:

```
DELETE QLOCAL(q-name) NOPURGE QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

**Note:** You always get the NOPURGE option even if you specify MQPO\_YES for *Purge*. To delete messages on local copies of the queues, you must explicitly issue, for each copy, the Delete Queue command with a *QSGDisposition* value of MQQSGD\_COPY and a *Purge* value of MQPO\_YES.

#### **MQQSGD\_Q\_MGR**

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_Q\_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

MQQSGD\_Q\_MGR is the default value.

#### **MQQSGD\_SHARED**

Valid only for queue of type local.

The object resides in the shared repository. The object was defined by a command using the parameter MQQSGD\_SHARED. Any object residing on the page set of the queue manager that executes the command, or any object defined by a command using the parameter MQQSGD\_GROUP, is not affected by this command.

### **QType (MQCFIN)**

Queue type (parameter identifier: MQIA\_Q\_TYPE).

If this parameter is present, the queue must be of the specified type.

The value can be:

#### **MQQT\_ALIAS**

Alias queue definition.

#### **MQQT\_LOCAL**

Local queue.

#### **MQQT\_REMOTE**

Local definition of a remote queue.

## **MQQT\_MODEL**

Model queue definition.

V 9.3.0

ALW

### **IgnoreState (MQCFST)**

Specifies whether the command fails if the queue does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### **MQIS\_YES**

The command succeeds regardless of whether the queue exists.

#### **MQIS\_NO**

The command fails if the queue does not exist. This is the default value.

## **Error codes (Delete Queue)**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRC\_Q\_NOT\_EMPTY**

(2055, X'807') Queue contains one or more messages or uncommitted put or get requests.

Multi

### **MQCMD\_DELETE\_SERVICE (Delete Service) on Multiplatforms**

The Delete Service (MQCMD\_DELETE\_SERVICE) PCF command deletes an existing service definition.

## **Required parameters**

### **ServiceName (MQCFST)**

Service name (parameter identifier: MQCA\_SERVICE\_NAME).

This parameter is the name of the service definition to be deleted.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

## **Optional parameters**

ALW

V 9.3.0

### **IgnoreState (MQCFST)**

Specifies whether the command fails if the service does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### **MQIS\_YES**

The command succeeds regardless of whether the service exists.

#### **MQIS\_NO**

The command fails if the service does not exist. This is the default value.

z/OS

### **MQCMD\_DELETE\_STG\_CLASS (Delete Storage Class) on z/OS**

The Delete Storage Class (MQCMD\_DELETE\_STG\_CLASS) PCF command deletes an existing storage class definition.

## **Required parameters**

### **StorageClassName (MQCFST)**

Storage class name (parameter identifier: MQCA\_STORAGE\_CLASS).

The storage class definition to be deleted. The maximum length of the string is

MQ\_STORAGE\_CLASS\_LENGTH.

## Optional parameters

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

#### MQQSGD\_COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD\_Q\_MGR, is not affected by this command.

#### MQQSGD\_GROUP

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD\_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE STGCLASS(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

#### MQQSGD\_Q\_MGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_Q\_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

MQQSGD\_Q\_MGR is the default value.

### ***MQCMD\_DELETE\_SUBSCRIPTION (Delete Subscription)***

The Delete Subscription (MQCMD\_DELETE\_SUBSCRIPTION) PCF command deletes a subscription.

## Required parameters

### SubName (MQCFST)

Subscription name (parameter identifier: MQCACF\_SUB\_NAME).

Specifies the unique subscription name. The subscription name, if provided, must be fully specified; a wildcard is not acceptable.

The subscription name must refer to a durable subscription.

If *SubName* is not provided, *SubId* must be specified to identify the subscription to be deleted.

The maximum length of the string is MQ\_SUB\_NAME\_LENGTH.

### **SubId (MQCFBS)**

Subscription identifier (parameter identifier: MQBACF\_SUB\_ID).

Specifies the unique internal subscription identifier.

You must supply a value for *SubId* if you have not supplied a value for *SubName*.

The maximum length of the string is MQ\_CORREL\_ID\_LENGTH.

## **Optional parameters for z/OS**



### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- Blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- A queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- An asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter on which to filter.

## **Optional parameters for AIX, Linux, and Windows**



### **V9.3.0 IgnoreState (MQCFST)**

Specifies whether the command fails if the subscription does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

#### **MQIS\_YES**

The command succeeds regardless of whether the subscription exists.

#### **MQIS\_NO**

The command fails if the subscription does not exist. This is the default value.

### **MQCMD\_DELETE\_TOPIC (Delete Topic)**

The Delete Topic (MQCMD\_DELETE\_TOPIC) PCF command deletes the specified administrative topic object.

## **Required parameters**

### **TopicName (MQCFST)**

The name of the administrative topic definition to be deleted (parameter identifier: MQCA\_TOPIC\_NAME).

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

## Optional parameters for z/OS

z/OS

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be any of the following values:

#### MQQSGD\_COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD\_Q\_MGR, is not affected by this command.

#### MQQSGD\_GROUP

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD\_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the deletion is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to make, or delete, local copies on page set zero:

```
DELETE TOPIC(name) QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

#### MQQSGD\_Q\_MGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD\_Q\_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

MQQSGD\_Q\_MGR is the default value.

## Optional parameters for Multiplatforms

Multi

### Authrec (MQCFIN)

Authrec (parameter identifier: MQIACF\_REMOVE\_AUTHREC).

Specifies whether the associated authority record is also deleted.



This parameter does not apply to z/OS.

The value can be any of the following values:

**MQRAR\_YES**

The authority record associated with the object is deleted. This is the default.

**MQRAR\_NO**

The authority record associated with the object is not deleted.

**V 9.3.0 IgnoreState (MQCFST)**

Specifies whether the command fails if the topic does not exist (parameter identifier MQIACF\_IGNORE\_STATE). The value can be any of the following values:

**MQIS\_YES**

The command succeeds regardless of whether the topic exists.

**MQIS\_NO**

The command fails if the topic does not exist. This is the default value.

**Multi MQCMD\_ESCAPE (Escape) on Multiplatforms**

The Escape (MQCMD\_ESCAPE) PCF command conveys any IBM MQ command (MQSC) to a remote queue manager.

Use the Escape command when the queue manager (or application) sending the command does not support the particular IBM MQ command, and so does not recognize it and cannot construct the required PCF command.

The Escape command can also be used to send a command for which no Programmable Command Format has been defined.

The only type of command that can be carried is one that is identified as an MQSC, that is recognized at the receiving queue manager.

**Required parameters**

**EscapeType (MQCFIN)**

Escape type (parameter identifier: MQIACF\_ESCAPE\_TYPE).

The only value supported is:

**MQET\_MQSC**

IBM MQ command.

**EscapeText (MQCFST)**

Escape text (parameter identifier: MQIACF\_ESCAPE\_TEXT).

A string to hold a command. The length of the string is limited only by the size of the message.

**Error codes**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_ESCAPE\_TYPE\_ERROR**

Escape type not valid.

**Multi MQCMD\_ESCAPE (Escape) Response on Multiplatforms**

The response to the Escape (MQCMD\_ESCAPE) PCF command consists of the response header followed by two parameter structures, one containing the escape type, and the other containing the text response.

More than one such message might be issued, depending upon the command contained in the Escape request.

The *Command* field in the response header MQCFH contains the MQCMD\_\* command identifier of the text command contained in the **EscapeText** parameter in the original Escape command. For example, if *EscapeText* in the original Escape command specified PING QMGR, *Command* in the response has the value MQCMD\_PING\_Q\_MGR.

If it is possible to determine the outcome of the command, the *CompCode* in the response header identifies whether the command was successful. The success or otherwise can therefore be determined without the recipient of the response having to parse the text of the response.

If it is not possible to determine the outcome of the command, *CompCode* in the response header has the value MQCC\_UNKNOWN, and *Reason* is MQRC\_NONE.

## Parameters

### EscapeType (MQCFIN)

Escape type (parameter identifier: MQIACF\_ESCAPE\_TYPE).

The only value supported is:

#### MQET\_MQSC

IBM MQ command.

### EscapeText (MQCFST)

Escape text (parameter identifier: MQCACF\_ESCAPE\_TEXT).

A string holding the response to the original command.

## **MQCMD\_INQUIRE\_APPL\_STATUS (inquire application status) on Multiplatforms**

The Inquire Application Status (MQCMD\_INQUIRE\_APPL\_STATUS) PCF command inquires about the applications and application instances connected to a queue manager or uniform cluster.

You must specify the application name for which you want to receive status information.

## Required parameters

### ApplicationName (MQCFST)

Application name set using the APPPLTAG parameter (parameter identifier: MQCACF\_APPL\_NAME).

Generic application names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all applications having names that start with the selected character string. An asterisk on its own matches all possible names.

The application name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ\_APPL\_NAME\_LENGTH.

## Optional parameters

### ApplicationInfoAttrs (MQCFIL)

Application information attributes (parameter identifier: MQIACF\_APPL\_INFO\_ATTRS)

If not provided, defaults to MQIACF\_ALL

Alternatively, you can specify any of the parameter values listed in the [Inquire Application Status \(Response\)command](#), which are valid for the requested status type.

### ApplicationStatusInfoType (MQCFIN)

The type of status to return (parameter identifier: MQIACF\_APPL\_INFO\_TYPE).

The value can be:

- MQIACF\_APPL\_INFO\_APPL

The command displays status information relating to each unique application name. This represents a summary of the details from the local queue manager and any queue manager in the same uniform cluster.

- **MQIACF\_APPL\_INFO\_QMGR**

The command displays status information relating to applications at a queue manager level, including the local queue manager and any queue manager in the same uniform cluster.

- **MQIACF\_APPL\_INFO\_LOCAL**

The command displays status information for applications, for each application instance connected to the local queue manager.

The default value, if this parameter is not specified, is **MQIACF\_APPL\_INFO\_APPL**.

### **V 9.3.0 ApplicationType (MQCFIL)**

Application type (parameter identifier: **MQIACF\_BALANCING\_TYPE**)

The application type (**MQBNO\_BALTYPE\_SIMPLE**, **MQBNO\_BALTYPE\_REQREP**, or **MQBNO\_BALTYPE\_RAMANAGED**) in effect for this application instance.

Note that different instances of the same application can provide different balancing options without causing any error.

### **V 9.3.0 BalancingOptions (MQCFIL)**

Application balancing options (parameter identifier: **MQIACF\_BALANCING\_OPTIONS**)

The balancing options (**MQBNO\_OPTIONS\_NONE** or **MQBNO\_OPTIONS\_IGNORE\_TRANS**) in effect for this application instance.

### **Connections(MQCFIN)**

The number of queue manager connections this application instance currently has open. (parameter identifier: **MQIACF\_CONNECTION\_COUNT**).

### **ConnectionTag (MQCFBS)**

The connection tag associated with this application instance. When generated by the queue manager, this is a UTF8 string. (parameter identifier: **MQBACF\_CONN\_TAG**).

The maximum length of this field is **MQ\_CONN\_TAG\_LENGTH**

### **ImmovableCount (MQCFIN)**

The count of times this application instance has been requested to move to another queue manager and has not yet disconnected. Any value higher than one is an indication that the application is failing to rebalance when requested to. (parameter identifier: **MQIACF\_APPL\_IMMOVABLE\_COUNT**).

### **ImmovableDate (MQCFST)**

Date on which this local instance is considered eligible for being moved around a uniform cluster. This field is blank unless there is a temporary condition which prevents an application instance being moved to another queue manager in a uniform cluster. (parameter identifier: **MQCACF\_APPL\_IMMOVABLE\_DATE**).

The length of the string is **MQ\_DATE\_LENGTH**

### **ImmovableReason (MQCFIN)**

The reason why this application is currently considered to be immovable and, therefore, will not be rebalanced around the cluster. Some reasons are temporary, and have an associated *ImmovableDate* and *ImmovableTime*, or *ImmovableTime* reason. Other reasons persist for the lifetime of this application instance. (parameter identifier: **MQIACF\_APPL\_IMMOVABLE\_REASON**).

The value can be any of the following values:

#### **MQIMMREASON\_NONE**

This application instance is currently considered movable.

#### **MQIMMREASON\_NOT\_CLIENT**

This application instance cannot be moved as it is not a client connection.

**MQIMMREASON\_NOT\_RECONNECTABLE**

This application instance cannot be moved as it is not a reconnectable client connection.

**MQIMMREASON\_MOVING**

This application instance cannot be moved as it has recently been requested to move, and has not yet disconnected.

**MQIMMREASON\_APPLNAME\_CHANGED**

This application instance cannot be moved as it is sharing a socket with a connection from an application instance which has a different application name.

**V 9.3.0 MQIMMREASON\_IN\_TRANSACTION**

The application instance has successfully performed at least one MQI operation within syncpoint, and the timeout specified to forcibly rebalance an instance has not been reached.

**V 9.3.0 MQIMMREASON\_TDQ\_OPEN\_INPUT**

The application instance has at least one dynamic queue open for input, and the timeout specified to forcibly rebalance an instance has not been reached.

**V 9.3.0 MQIMMREASON\_AWAITS\_REPLY**

The application instance has successfully performed at least one MQI operation within syncpoint, and the timeout specified to forcibly rebalance an instance has not been reached.

**ImmovableTime (MQCFST)**

Time on which this local instance is considered eligible for being moved around a uniform cluster. This field is blank unless there is a temporary condition which prevents an application instance being moved to another queue manager in a uniform cluster. (parameter identifier: MQCACF\_APPL\_IMMOVABLE\_TIME).

The length of the string is MQ\_TIME\_LENGTH

**IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor that you use to restrict the output from the command. The parameter identifier must be an integer type, and must be one of the values allowed for the **ApplicationStatusInfoType** selected, except MQIACF\_ALL.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

**Movable (MQCFIN)**

Indicates whether this application instance is considered eligible for moving around the uniform cluster. At a minimum, a movable application must be a client connection which has connected as reconnectable. (parameter identifier: MQIACF\_APPL\_MOVABLE).

The value can be any of the following values:

**MQACTIVE\_YES**

This application instance is considered movable.

**MQACTIVE\_NO**

This application instance is not considered movable.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter except MQCA\_APPL\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition.

Ensure that the parameter is valid for the type selected in **ApplicationStatusInfoType**.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

**Note:** Although the ConnectionTag (MQBACF\_CONN\_TAG) field in Inquire Application Status (Response) is defined as a binary field, the contents should be UTF8.

Because of this, use a *StringFilter* for this field, not a *ByteStringFilter*, and you can use all valid *StringFilter* operators.

### V 9.3.0 Timeout (MQCFIN)

Application timeout (parameter identifier: MQIACF\_BALANCING\_TIMEOUT)

The time out value of NEVER, IMMEDIATE, or the time in seconds up to a maximum of 999999999 seconds.

### Multi MQCMD\_INQUIRE\_APPL\_STATUS (inquire application status) Response on Multiplatforms

The response to the Inquire Application Status (MQCMD\_INQUIRE\_APPL\_STATUS) PCF command consists of the response header followed by the *ApplicationName* structure and the requested combination of attribute parameter structures (where applicable) for the requested *ApplicationStatusInfoType*.

#### Always returned:

*ApplicationName*

#### Returned if *ApplicationStatusInfoType* is MQIACF\_APPL\_INFO\_APPL:

*Balanced, ClusterName, InstanceCount, MovableInstanceCount, MqiacfApplInfoAppl*

#### Returned if *ApplicationStatusInfoType* is MQIACF\_APPL\_INFO\_QMGR:

*BalanceState, InstanceCount, LastMessageDate, LastMessageTime, MovableInstanceCount, QueueManagerActive, QueueManagerID, QueueManagerName, MqiacfApplInfoQmgr*

#### Returned if *ApplicationStatusInfoType* is MQIACF\_APPL\_INFO\_LOCAL:

*ApplicationType, BalancingOptions, Connections, ConnectionTag, ImmovableCount, ImmovableDate, ImmovableReason, ImmovableTime, Movable, MqiacfApplInfoLocal, Timeout*

### Response data (MQIACF\_APPL\_INFO\_APPL)

#### Balanced (MQCFIN)

The overall state of this application relative to whether it is balanced in a uniform cluster or not (parameter identifier: MQIACF\_BALANCED).

The value can be any of the following values:

##### MQBALANCED\_NO

This application is not considered balanced in the uniform cluster.

##### MQBALANCED\_YES

This application is considered balanced in the uniform cluster.

##### MQBALANCED\_NOT\_APPLICABLE

This application is not shared across a uniform cluster.

##### MQBALANCED\_UNKNOWN

This is a temporary state, representing an application that has not yet undergone a scan to calculate whether it is balanced or not, on at least one queue manager, across the uniform cluster.

#### ClusterName (MQCFST)

The name of the uniform cluster in which details of this application are being distributed (parameter identifier: MQCA\_CLUSTER\_NAME).

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

The value can be any of the following values:

##### Blank

If this application is not being distributed around a uniform cluster. This might be because the application has never connected in a way which is compatible with being moved (not

reconnectable, for example) or it might be that the queue manager is not a member of a uniform cluster.

**String**

The name of the uniform cluster.

**InstanceCount (MQCFIN)**

The summary count of application instances for this application. This includes the local queue managers count of instances, plus those from any queue manager in a uniform cluster that has distributed details about this application (parameter identifier: MQIACF\_APPL\_COUNT).

**MovableInstanceCount (MQCFIN)**

The summary count of the movable application instances for this application. This includes the local queue managers count of movable instances, plus those from any queue manager in a uniform cluster that has distributed details about this application (parameter identifier: MQIACF\_MOVABLE\_APPL\_COUNT).

**MqiacfApplInfoAppl**

Signifies that the response type is an application.

**Response data (MQIACF\_APPL\_INFO\_QMGR)**

**BalanceState (MQCFIN)**

The current state of this application for the queue manager being reported against, relative to whether it is considered balanced across a uniform cluster or not. This information is only updated periodically at the time when a scan causes the rebalancing to occur and might not be based on the current values for *InstanceCount* and *MovableInstanceCount* (parameter identifier: MQIACF\_BALSTATE).

The value can be any of the following values:

**MQBALSTATE\_LOW**

This application is not balanced in the uniform cluster and has a deficit of application instances. A queue manager in this state usually requests applications to be rebalanced to it, so as to balance out the cluster.

**MQBALSTATE\_OK**

This application is balanced in the uniform cluster.

**MQBALSTATE\_HIGH**

This application is not balanced in the uniform cluster and has a surplus of application instances. A queue manager in this state usually honors requests to rebalance some of the applications connected to it, over to a queue manager in the LOW state.

**MQBALSTATE\_NOT\_APPLICABLE**

This queue manager is not in a uniform cluster and, therefore, balancing cannot occur.

**MQBALSTATE\_UNKNOWN**

This is a temporary state representing an application that is new to the uniform cluster, and which has not yet undergone a scan to calculate whether it is balanced or not.

**InstanceCount (MQCFIN)**

The count of application instances for this application, on the queue manager being reported (parameter identifier: MQIACF\_APPL\_COUNT).

**LastMessageDate (MQCFST)**

Local date on which the queue manager being reported against, has distributed information on its application instances. For the local queue manager, this is just the current date. (parameter identifier: MQCACF\_LAST\_MSG\_DATE).

The length of the string is MQ\_DATE\_LENGTH

**LastMessageTime (MQCFST)**

Local time on which the queue manager being reported against, has distributed information on its application instances. For the local queue manager, this is just the current time. (parameter identifier: MQCACF\_LAST\_MSG\_TIME).

The length of the string is MQ\_TIME\_LENGTH

**MovableInstanceCount (MQCFIN)**

The summary count of the movable application instances for this application on the queue manager being reported for (parameter identifier: MQIA\_MOVABLE\_APPL\_COUNT).

**QueueManagerActive(MQCFIN)**

Indicates whether the queue manager being reported for is currently considered active. Application instances on an inactive queue manager are not included in the numbers used to calculate application instance balancing. (parameter identifier: MQIACF\_REMOTE\_QMGR\_ACTIVE).

The value can be any of the following values:

**MQACTIVE\_NO**

This queue manager is not considered active, because it has not distributed its application balancing information to the local queue manager recently.

**MQACTIVE\_YES**

This queue manager is considered active, and is actively distributing its application balancing information.

**QueueManagerID (MQCFST)**

The internally generated unique queue manager identifier of the queue manager being reported for (parameter identifier: MQCA\_Q\_MGR\_IDENTIFIER).

The length of the string is MQ\_Q\_MGR\_IDENTIFIER\_LENGTH.

**QueueManagerName (MQCFST)**

The queue manager name of the queue manager being reported for (parameter identifier: MQCA\_Q\_MGR\_NAME).

The length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**MqiacfApplInfoQmgr**

Signifies that the response type is a queue manager.

**Response data (MQIACF\_APPL\_INFO\_LOCAL)****V 9.3.0 ApplicationType (MQCFIL)**

Application type (parameter identifier: MQIACF\_BALANCING\_TYPE)

The application type (MQBNO\_BALTYPE\_SIMPLE, MQBNO\_BALTYPE\_REQREP, or MQBNO\_BALTYPE\_RAMANAGED) in effect for this application instance.

**V 9.3.0 BalancingOptions (MQCFIL)**

Application balancing options (parameter identifier: MQIACF\_BALANCING\_OPTIONS)

The balancing options (MQBNO\_OPTIONS\_NONE or MQBNO\_OPTIONS\_IGNORE\_TRANS) in effect for this application instance.

**Connections(MQCFIN)**

The number of queue manager connections this application instance currently has open. (parameter identifier: MQIACF\_CONNECTION\_COUNT).

**ConnectionTag (MQCFBS)**

The connection tag associated with this application instance. When generated by the queue manager, this is a UTF8 string. (parameter identifier: MQBACF\_CONN\_TAG).

The maximum length of this field is MQ\_CONN\_TAG\_LENGTH

**ImmovableCount (MQCFIN)**

The count of times this application instance has been requested to move to another queue manager and has not yet disconnected. Any value higher than one is an indication that the application is failing to rebalance when requested to. (parameter identifier: MQIACF\_APPL\_IMMOVABLE\_COUNT).

**ImmovableDate (MQCFST)**

Date on which this local instance is considered eligible for being moved around a uniform cluster. This field is blank unless there is a temporary condition which prevents an application

instance being moved to another queue manager in a uniform cluster. (parameter identifier: MQCACF\_APPL\_IMMOVABLE\_DATE).

The length of the string is MQ\_DATE\_LENGTH

### **ImmovableReason (MQCFIN)**

The reason why this application is currently considered to be immovable and, therefore, will not be rebalanced around the cluster. Some reasons are temporary, and have an associated *ImmovableDate* and *ImmovableTime*, or *ImmovableTime* reason. Other reasons persist for the lifetime of this application instance. (parameter identifier: MQIACF\_APPL\_IMMOVABLE\_REASON).

The value can be any of the following values:

#### **MQIMMREASON\_NONE**

This application instance is currently considered movable.

#### **MQIMMREASON\_NOT\_CLIENT**

This application instance cannot be moved as it is not a client connection.

#### **MQIMMREASON\_NOT\_RECONNECTABLE**

This application instance cannot be moved as it is not a reconnectable client connection.

#### **MQIMMREASON\_MOVING**

This application instance cannot be moved as it has recently been requested to move, and has not yet disconnected.

#### **MQIMMREASON\_APPLNAME\_CHANGED**

This application instance cannot be moved as it is sharing a socket with a connection from an application instance which has a different application name.

#### **V 9.3.0 MQIMMREASON\_IN\_TRANSACTION**

The application instance has successfully performed at least one MQI operation within syncpoint, and the timeout specified to forcibly rebalance an instance has not been reached.

#### **V 9.3.0 MQIMMREASON\_TDQ\_OPEN\_INPUT**

The application instance has at least one dynamic queue open for input, and the timeout specified to forcibly rebalance an instance has not been reached.

#### **V 9.3.0 MQIMMREASON\_AWAITS\_REPLY**

The application instance has successfully performed at least one MQI operation within syncpoint, and the timeout specified to forcibly rebalance an instance has not been reached.

### **ImmovableTime (MQCFST)**

Time on which this local instance is considered eligible for being moved around a uniform cluster. This field is blank unless there is a temporary condition which prevents an application instance being moved to another queue manager in a uniform cluster. (parameter identifier: MQCACF\_APPL\_IMMOVABLE\_TIME).

The length of the string is MQ\_TIME\_LENGTH

### **Movable (MQCFIN)**

Indicates whether this application instance is considered eligible for moving around the uniform cluster. At a minimum, a movable application must be a client connection which has connected as reconnectable. (parameter identifier: MQIACF\_APPL\_MOVABLE).

The value can be any of the following values:

#### **MQACTIVE\_YES**

This application instance is considered movable.

#### **MQACTIVE\_NO**

This application instance is not considered movable.

### **MqiacfApplInfoLocal**

Signifies that the response type is local.



## V 9.3.0 Timeout (MQCFIN)

Application timeout (parameter identifier: MQIACF\_BALANCING\_TIMEOUT)

The time out value of NEVER, IMMEDIATE, or the time in seconds up to a maximum of 999999999 seconds.

### Related tasks

[Monitoring application balancing](#)

### Related reference

“DISPLAY APSTATUS (display application status) on Multiplatforms” on page 632

Use the MQSC command **DISPLAY APSTATUS** to display the status of one or more applications and application instances connected to a queue manager or a uniform cluster.

## z/OS MQCMD\_INQUIRE\_ARCHIVE (Inquire Archive) on z/OS

The Inquire Archive (MQCMD\_INQUIRE\_ARCHIVE) PCF command returns archive system parameters and information.

### Optional parameters

#### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## z/OS MQCMD\_INQUIRE\_ARCHIVE (Inquire Archive) Response on z/OS

The response to the Inquire Archive (MQCMD\_INQUIRE\_ARCHIVE) PCF command consists of the response header followed by the *ParameterType* structure and the combination of attribute parameter structures determined by the value of *ParameterType*.

### Always returned:

*ParameterType* Specifies the type of archive information being returned. The value can be any of the following values:

#### MQSYSP\_TYPE\_INITIAL

The initial settings of the archive parameters.

#### MQSYSP\_TYPE\_SET

The settings of the archive parameters if they have been altered since their initial setting.

#### MQSYSP\_TYPE\_ARCHIVE\_TAPE

Parameters relating to the tape unit (if in use). There is one such message per tape unit in use for archive logging.

### Returned if *ParameterType* is MQSYSP\_TYPE\_INITIAL (one message is returned):

*AllocPrimary, AllocSecondary, AllocUnits, ArchivePrefix1, ArchivePrefix2, ArchiveRetention, ArchiveUnit1, ArchiveUnit2, ArchiveWTOR, BlockSize, Catalog, Compact, Protect, QuiesceInterval, RoutingCode, TimeStampFormat*

**Returned if *ParameterType* is MQSYSP\_TYPE\_SET and any value is set (one message is returned):**

*AllocPrimary, AllocSecondary, AllocUnits, ArchivePrefix1, ArchivePrefix2, ArchiveRetention, ArchiveUnit1, ArchiveUnit2, ArchiveWTOR, BlockSize, Catalog, Compact, Protect, QuiesceInterval, RoutingCode, TimeStampFormat*

**Returned if *ParameterType* is MQSYSP\_TYPE\_ARCHIVE\_TAPE (one message is returned for each tape unit in use for archive logging):**

*DataSetName, LogCorrelId, UnitAddress, UnitStatus, UnitVolser*

## Response data - archive parameter information

### **AllocPrimary (MQCFIN)**

Primary space allocation for DASD data sets (parameter identifier: MQIACF\_SYSP\_ALLOC\_PRIMARY).

Specifies the primary space allocation for DASD data sets in the units specified in the **AllocUnits** parameter.

### **AllocSecondary (MQCFIN)**

Secondary space allocation for DASD data sets (parameter identifier: MQIACF\_SYSP\_ALLOC\_SECONDARY).

Specifies the secondary space allocation for DASD data sets in the units specified in the **AllocUnits** parameter.

### **AllocUnits (MQCFIN)**

Allocation unit (parameter identifier: MQIACF\_SYSP\_ALLOC\_UNIT).

Specifies the unit in which primary and secondary space allocations are made. The value can be any of the following values:

#### **MQSYSP\_ALLOC\_BLK**

Blocks.

#### **MQSYSP\_ALLOC\_TRK**

Tracks.

#### **MQSYSP\_ALLOC\_CYL**

Cylinders.

### **ArchivePrefix1 (MQCFST)**

Prefix for the first archive log data set name (parameter identifier: MQCACF\_SYSP\_ARCHIVE\_PFX1).

The maximum length of the string is MQ\_ARCHIVE\_PFX\_LENGTH.

### **ArchivePrefix2 (MQCFST)**

Prefix for the second archive log data set name (parameter identifier: MQCACF\_SYSP\_ARCHIVE\_PFX2).

The maximum length of the string is MQ\_ARCHIVE\_PFX\_LENGTH.

### **ArchiveRetention (MQCFIN)**

Archive retention period (parameter identifier: MQIACF\_SYSP\_ARCHIVE\_RETAIN).

Specifies the retention period, in days, to be used when the archive log data set is created.

### **ArchiveUnit1 (MQCFST)**

Specifies the device type or unit name of the device that is used to store the first copy of the archive log data set (parameter identifier: MQCACF\_SYSP\_ARCHIVE\_UNIT1).

The maximum length of the string is MQ\_ARCHIVE\_UNIT\_LENGTH.

### **ArchiveUnit2 (MQCFST)**

Specifies the device type or unit name of the device that is used to store the second copy of the archive log data set (parameter identifier: MQCACF\_SYSP\_ARCHIVE\_UNIT2).

The maximum length of the string is MQ\_ARCHIVE\_UNIT\_LENGTH.

**ArchiveWTOR (MQCFIN)**

Specifies whether a message is to be sent to the operator and a reply is received before attempting to mount an archive log data set (parameter identifier: MQIACF\_SYSP\_ARCHIVE\_WTOR).

The value can be:

**MQSYSP\_YES**

A message is to be sent and a reply received before an attempt to mount an archive log data set.

**MQSYSP\_NO**

A message is not to be sent and a reply received before an attempt to mount an archive log data set.

**BlockSize (MQCFIN)**

Block size of the archive log data set (parameter identifier: MQIACF\_SYSP\_BLOCK\_SIZE).

**Catalog (MQCFIN)**

Specifies whether archive log data sets are cataloged in the primary integrated catalog facility (parameter identifier: MQIACF\_SYSP\_CATALOG).

The value can be:

**MQSYSP\_YES**

Archive log data sets are cataloged.

**MQSYSP\_NO**

Archive log data sets are not cataloged.

**Compact (MQCFIN)**

Specifies whether data written to archive logs is to be compacted (parameter identifier: MQIACF\_SYSP\_COMPACT).

The value can be any of the following values:

**MQSYSP\_YES**

Data is to be compacted.

**MQSYSP\_NO**

Data is not to be compacted.

**Protect (MQCFIN)**

Protection by external security manager (ESM) (parameter identifier: MQIACF\_SYSP\_PROTECT).

Specifies whether archive log data sets are protected by ESM profiles when the data sets are created.

The value can be any of the following values:

**MQSYSP\_YES**

Data set profiles are created when logs are offloaded.

**MQSYSP\_NO**

Profiles are not created.

**QuiesceInterval (MQCFIN)**

Maximum time allowed for the quiesce (parameter identifier: MQIACF\_SYSP\_QUIESCE\_INTERVAL).

Specifies the maximum time, in seconds, allowed for the quiesce.

**RoutingCode (MQCFIL)**

z/OS routing code list (parameter identifier: MQIACF\_SYSP\_ROUTING\_CODE).

Specifies the list of z/OS routing codes for messages about the archive log data sets to the operator. There can be 1 - 14 entries in the list.

**TimeStampFormat (MQCFIN)**

Time stamp included (parameter identifier: MQIACF\_SYSP\_TIMESTAMP).

Specifies whether the archive log data set name has a time stamp in it.

The value can be:

**MQSYSP\_YES**

Names include a time stamp.

**MQSYSP\_NO**

Names do not include a time stamp.

**MQSYSP\_EXTENDED**

Names include a time stamp.

**Response data - tape unit status information****DataSetName (MQCFST)**

Data set name (parameter identifier: MQCACF\_DATA\_SET\_NAME).

Specifies the data set name on the tape volume that is being processed, or was last processed.

The maximum length of the string is MQ\_DATA\_SET\_NAME\_LENGTH.

**LogCorrelId (MQCFST)**

Correlation identifier (parameter identifier: MQCACF\_SYSP\_LOG\_CORREL\_ID).

Specifies the correlation ID associated with the user of the tape being processed. This parameter is blank if there is no current user.

The maximum length of the string is MQ\_LOG\_CORREL\_ID\_LENGTH.

**UnitAddress (MQCFIN)**

Tape unit address: MQIACF\_SYSP\_UNIT\_ADDRESS).

Specifies the physical address of the tape unit allocated to read the archive log.

**UnitStatus (MQCFIN)**

Status if the tape unit: MQIACF\_SYSP\_UNIT\_STATUS).

The value can be:

**MQSYSP\_STATUS\_BUSY**

The tape unit is busy, actively processing an archive log data set.

**MQSYSP\_STATUS\_PREMOUNT**

The tape unit is active and allocated for premounting.

**MQSYSP\_STATUS\_AVAILABLE**

The tape unit is available, inactive, and waiting for work.

**MQSYSP\_STATUS\_UNKNOWN**

The tape unit status is unknown.

**UnitVolser (MQCFST)**

The volume serial number of the tape that is mounted (parameter identifier: MQCACF\_SYSP\_UNIT\_VOLSER).

The maximum length of the string is MQ\_VOLSER\_LENGTH.

**MQCMD\_INQUIRE\_AUTH\_INFO (Inquire Authentication Information Object)**

The Inquire authentication information object (MQCMD\_INQUIRE\_AUTH\_INFO) PCF command inquires about the attributes of authentication information objects.

**Required parameters****AuthInfoName (MQCFST)**

Authentication information object name (parameter identifier: MQCA\_AUTH\_INFO\_NAME).

Specifies the name of the authentication information object about which information is to be returned.

Generic authentication information object names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all authentication information objects

having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH.

## Optional parameters

### AuthInfoAttrs (MQCFIL)

Authentication information object attributes (parameter identifier: MQIACF\_AUTH\_INFO\_ATTRS).

The attribute list can specify the following value - the default value if the parameter is not specified):

#### **MQIACF\_ALL**

All attributes.

or a combination of the following:

#### **MQIA\_ADOPT\_CONTEXT**

Adopt the presented credentials as the context for the application.

#### **MQCA\_ALTERATION\_DATE**

Date on which the definition was last altered.

#### **MQCA\_ALTERATION\_TIME**

Time at which the definition was last altered.

#### **MQCA\_AUTH\_INFO\_DESC**

Description of the authentication information object.

#### **MQCA\_AUTH\_INFO\_NAME**

Name of the authentication information object.

#### **MQIA\_AUTH\_INFO\_TYPE**

Type of authentication information object.

#### **MQCA\_AUTH\_INFO\_CONN\_NAME**

Connection name of the authentication information object.

This attribute is relevant only when **AuthInfoType** is set to MQAIT\_CRL\_LDAP or MQAIT\_IDPW\_LDAP.

#### **MQIA\_AUTHENTICATION\_FAIL\_DELAY**

Delay in seconds before an authentication failure is returned to an application.

#### **MQIA\_AUTHENTICATION\_METHOD**

Authentication method for user passwords.

#### **MQIA\_CHECK\_CLIENT\_BINDING**

Authentication requirements for client applications.

#### **MQIA\_CHECK\_LOCAL\_BINDING**

Authentication requirements for locally bound applications.

#### **MQIA\_LDAP\_AUTHORMD**

Authorization method for the queue manager.

#### **MQCA\_LDAP\_BASE\_DN\_GROUPS**

The base Distinguished Name for groups in the LDAP server.

#### **MQCA\_LDAP\_BASE\_DN\_USERS**

The base Distinguished Name for users in the LDAP server.

#### **MQCA\_LDAP\_FIND\_GROUP\_FIELD**

Name of the attribute used within an LDAP entry to determine group membership.

#### **MQCA\_LDAP\_GROUP\_ATTR\_FIELD**

LDAP attribute that represents a simple name for the group.

#### **MQCA\_LDAP\_GROUP\_OBJECT\_CLASS**

The LDAP object class used for group records in the LDAP repository.

**MQIA\_LDAP\_NESTGRP**

Whether LDAP groups are checked for membership of other groups.

**MQCA\_LDAP\_PASSWORD**

LDAP password in the authentication information object.

This attribute is relevant only when **AuthInfoType** is set to MQAIT\_CRL\_LDAP or MQAIT\_IDPW\_LDAP.

**MQIA\_LDAP\_SECURE\_COMM**

Whether connectivity to the LDAP server should be done securely using TLS.

**MQCA\_LDAP\_SHORT\_USER\_FIELD**

The field in the LDAP user record to be used as a short user name in IBM MQ.

**MQCA\_LDAP\_USER\_ATTR\_FIELD**

The field in the LDAP user record to be used to interpret the user ID provided by an application, if the user ID does not contain a qualifier.

**MQCA\_LDAP\_USER\_NAME**

LDAP user name in the authentication information object.

This attribute is relevant only when **AuthInfoType** is set to MQAIT\_CRL\_LDAP or MQAIT\_IDPW\_LDAP.

**MQCA\_LDAP\_USER\_OBJECT\_CLASS**

The LDAP object class used for user records in the LDAP repository.

**MQCA\_AUTH\_INFO\_OCSP\_URL**

The URL of the OCSP responder used to check for certificate revocation.

**AuthInfoType (MQCFIN)**

Type of authentication information object. The following values are accepted:

**MQAIT\_CRL\_LDAP**

Authentication information objects specifying Certificate Revocation Lists held on LDAP servers.

**MQAIT\_OCSP**

Authentication information objects specifying certificate revocation checking using OCSP.

**MQAIT\_IDPW\_OS**

Authentication information objects specifying certificate revocation checking using user ID and password checking through the operating system.

**MQAIT\_IDPW\_LDAP**

Authentication information objects specifying certificate revocation checking using user ID and password checking through an LDAP server.

**MQAIT\_ALL**

Authentication information objects of any type.

**z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- Blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- Aqueue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- An asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use **CommandScope** as a parameter to filter on.

### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in **AuthInfoAttrs**, except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 1523 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

### **z/OS QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. This value is the default value if the parameter is not specified.

#### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. This value is permitted only in a shared queue environment.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

#### **MQQSGD\_PRIVATE**

The object is defined as either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

You cannot use **QSGDisposition** as a parameter to filter on.

### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in **AuthInfoAttrs**, except MQCA\_AUTH\_INFO\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. For information about using this filter condition, see “MQCFSF - PCF string filter parameter” on page 1530.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

### **MQCMD\_INQUIRE\_AUTH\_INFO (Inquire Authentication Information Object) Response**

The response of the Inquire authentication information (MQCMD\_INQUIRE\_AUTH\_INFO) PCF command consists of the response header followed by the *AuthInfoName* structure (and on z/OS only, the *QSGDisposition* structure), and the requested combination of attribute parameter structures (where applicable).

#### **Always returned:**

*AuthInfoName* **z/OS**, *QSGDisposition*

**Returned if requested:**

*AdoptContext, AlterationDate, AlterationTime, AuthInfoConnName, BaseDNGroup, BaseDNUser, AuthInfoType, CheckClient, CheckLocal, ClassUser, FailureDelay, LDAPPASSWORD, LDAPUserName, OCSPResponderURL, SecureComms, ShortUser, UserField*

**Response data****AdoptContext**

Whether to use the presented credentials as the context for this application.

**AlterationDate (MQCFST)**

Alteration date of the authentication information object, in the form yyyy-mm-dd (parameter identifier: MQCA\_ALTERATION\_DATE).

**AlterationTime (MQCFST)**

Alteration time of the authentication information object, in the form hh.mm.ss (parameter identifier: MQCA\_ALTERATION\_TIME).

**AuthInfoConnName (MQCFST)**

The connection name of the authentication information object (parameter identifier: MQCA\_AUTH\_INFO\_CONN\_NAME).

The maximum length of the string is MQ\_AUTH\_INFO\_CONN\_NAME\_LENGTH. On z/OS, it is MQ\_LOCAL\_ADDRESS\_LENGTH.

This parameter is relevant only when AuthInfoType is set to *MQAIT\_CRL\_LDAP* or *MQAIT\_IDPW\_LDAP*.

**AuthInfoDesc (MQCFST)**

The description of the authentication information object (parameter identifier: MQCA\_AUTH\_INFO\_DESC).

The maximum length is MQ\_AUTH\_INFO\_DESC\_LENGTH.

**AuthInfoName (MQCFST)**

Authentication information object name (parameter identifier: MQCA\_AUTH\_INFO\_NAME).

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH.

**AuthInfoType (MQCFIN)**

The type of authentication information object (parameter identifier: MQIA\_AUTH\_INFO\_TYPE).

The value can be:

**MQAIT\_CRL\_LDAP**

This authentication information object specifies Certificate Revocation Lists that are held on LDAP servers.

**MQAIT\_OCSP**

This authentication information object specifies certificate revocation checking using OCSP.

**MQAIT\_IDPW\_OS**

This authentication information object specifies certificate revocation checking using user ID and password checking through the operating system.

**MQAIT\_IDPW\_LDAP**

This authentication information object specifies certificate revocation checking using user ID and password checking through an LDAP server.

See [Securing IBM MQ](#) for more information.

**AuthenticationMethod (MQCFIN)**

Authentication methods for user passwords (parameter identifier: MQIA\_AUTHENTICATION\_METHOD). Possible values are:

**MQAUTHENTICATE\_OS**

Use the traditional UNIX password verification method.

**MQAUTHENTICATE\_PAM**

Use the Pluggable Authentication Method to authenticate the user passwords.



You can set the PAM value only on AIX and Linux.

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_OS*, and is not valid on IBM MQ for z/OS.

#### **AuthorizationMethod (MQCFIN)**

Authorization methods for the queue manager (parameter identifier MQIA\_LDAP\_AUTHORMD). Possible values are:

##### **MQLDAP\_AUTHORMD\_OS**

Use operating system groups to determine permissions associated with a user.

##### **MQLDAP\_AUTHORMD\_SEARCHGRP**

A group entry in the LDAP repository contains an attribute listing the Distinguished Name of all the users belonging to that group.

##### **MQLDAP\_AUTHORMD\_SEARCHUSER**

A user entry in the LDAP repository contains an attribute listing the Distinguished Name of all the groups to which the specified user belongs.

##### **MQLDAP\_AUTHORMD\_SRCHGRPSN**

A group entry in the LDAP repository contains an attribute listing the short user name of all the users belonging to that group.

#### **BaseDNGroup (MQCFST)**

In order to be able to find group names, this parameter must be set with the base DN to search for groups in the LDAP server (parameter identifier MQCA\_LDAP\_BASE\_DN\_GROUPS).

The maximum length of the string is MQ\_LDAP\_BASE\_DN\_LENGTH.

#### **BaseDNUser (MQCFST)**

In order to be able to find the short user name attribute (see [ShortUser](#) ) this parameter must be set with the base DN to search for users within the LDAP server.

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_LDAP* and is mandatory (parameter identifier MQ\_LDAP\_BASE\_DN\_USERS).

The maximum length is MQ\_LDAP\_BASE\_DN\_LENGTH.

#### **Checklocal or Checkclient (MQCFIN)**

These attributes are valid only for an **AuthInfoType** of *MQAIT\_IDPW\_OS* or *MQAIT\_IDPW\_LDAP* (parameter identifier MQIA\_CHECK\_LOCAL\_BINDING or MQIA\_CHECK\_CLIENT\_BINDING). The possible values are:

##### **MQCHK\_NONE**

Switches off checking.


##### **MQCHK\_OPTIONAL**

Ensures that if a user ID and password are provided by an application, they are a valid pair, but that it is not mandatory to provide them. This option might be useful during migration, for example.

##### **MQCHK\_REQUIRED**

Requires that all applications provide a valid user ID and password.

##### **MQCHK\_REQUIRED\_ADMIN**

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the OPTIONAL setting. See also the following note.  (This setting is not allowed on z/OS systems.)

#### **ClassGroup (MQCFST)**

The LDAP object class used for group records in the LDAP repository (parameter identifier MQCA\_LDAP\_GROUP\_OBJECT\_CLASS).

#### **Classuser (MQCFST)**

The LDAP object class used for user records in the LDAP repository (parameter identifier MQCA\_LDAP\_USER\_OBJECT\_CLASS).

The maximum length is MQ\_LDAP\_CLASS\_LENGTH.

### **FailureDelay (MQCFIN)**

The failure delay (parameter identifier MQIA\_AUTHENTICATION\_FAIL\_DELAY) when an authentication fails due to the user ID or password being incorrect, in seconds, before the failure is returned to the application.

### **FindGroup (MQCFST)**

Name of the attribute used within an LDAP entry to determine group membership (parameter identifier MQCA\_LDAP\_FIND\_GROUP\_FIELD).

The maximum length of the string is MQ\_LDAP\_FIELD\_LENGTH.

### **GroupField (MQCFST)**

LDAP attribute that represents a simple name for the group (parameter identifier MQCA\_LDAP\_GROUP\_ATTR\_FIELD).

The maximum length of the string is MQ\_LDAP\_FIELD\_LENGTH.

### **GroupNesting (MQCFIN)**

Whether groups are members of other groups (parameter identifier MQIA\_LDAP\_NESTGRP). The values can be:

#### **MQLDAP\_NESTGRP\_NO**

Only the initially discovered groups are considered for authorization.

#### **MQLDAP\_NESTGRP\_YES**

The group list is searched recursively to enumerate all the groups to which a user belongs.

### **LDAPPassword (MQCFST)**

The LDAP password (parameter identifier: MQCA\_LDAP\_PASSWORD).

The maximum length is MQ\_LDAP\_PASSWORD\_LENGTH.

This parameter is relevant only when AuthInfoType is set to *MQAIT\_CRL\_LDAP* or *MQAIT\_IDPW\_LDAP*.

### **LDAPUserName (MQCFST)**

The LDAP user name (parameter identifier: MQCA\_LDAP\_USER\_NAME).

The Distinguished Name of the user who is binding to the directory.

The maximum length is MQ\_DISTINGUISHED\_NAME\_LENGTH. On z/OS, it is MQ\_SHORT\_DNAME\_LENGTH.

This parameter is relevant only when AuthInfoType is set to *MQAIT\_CRL\_LDAP* or *MQAIT\_IDPW\_LDAP*.

### **OCSPResponderURL (MQCFST)**

The URL of the OCSP responder used to check for certificate revocation.

### **z/OS QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be any of the following values:

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

### **SecureComms (MQCFIN)**

Whether connectivity to the LDAP server should be done securely using TLS (parameter identifier MQIA\_LDAP\_SECURE\_COMM).

The maximum length is MQ\_LDAP\_SECURE\_COMM\_LENGTH.

### **ShortUser (MQCFST)**

A field in the user record to be used as a short user name in IBM MQ (parameter identifier MQCA\_LDAP\_SHORT\_USER\_FIELD).

This field must contain values of 12 characters or less. This short user name is used for the following purposes:

- If LDAP authentication is enabled, but LDAP authorization is not enabled, this is used as an operating system user ID for authorization checks. In this case, the attribute must represent an operating system user ID.
- If LDAP authentication and authorization are both enabled, this is used as the user ID carried with the message in order for the LDAP user name to be rediscovered when the user ID inside the message needs to be used.

For example, on another queue manager, or when writing report messages. In this case, the attribute does not need to represent an operating system user ID, but must be a unique string. An employee serial number is an example of a good attribute for this purpose.

This attribute is valid only for an **AuthInfoType** of *MQAIT\_IDPW\_LDAP* and is mandatory.

The maximum length is MQ\_LDAP\_FIELD\_LENGTH.

### **UserField (MQCFST)**

Identifies the field in the LDAP user record that is used to interpret the provided user ID, only if the user ID does not contain a qualifier (parameter identifier MQCA\_LDAP\_USER\_ATTR\_FIELD).

The maximum length is MQ\_LDAP\_FIELD\_LENGTH.

### **MQCMD\_INQUIRE\_AUTH\_INFO\_NAMES (Inquire Authentication Information Object Names)**

The Inquire authentication information names (MQCMD\_INQUIRE\_AUTH\_INFO\_NAMES) PCF command asks for a list of authentication information names that match the generic authentication information name specified.

## **Required parameters**

### **AuthInfoName (MQCFST)**

Authentication information object name (parameter identifier: MQCA\_AUTH\_INFO\_NAME).

Specifies the name of the authentication information object about which information is to be returned.

Generic authentication information object names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all authentication information objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_AUTH\_INFO\_NAME\_LENGTH.

## **Optional parameters**

### **AuthInfoType (MQCFIN)**

Type of authentication information object. The following values are accepted:

#### **MQAIT\_CRL\_LDAP**

Authentication information objects specifying Certificate Revocation Lists held on LDAP servers.

#### **MQAIT\_OCSP**

Authentication information objects specifying certificate revocation checking using OCSP.

#### **MQAIT\_ALL**

Authentication information objects of any type. MQAIT\_ALL is the default value



### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.



### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

#### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

#### **MQQSGD\_PRIVATE**

The object is defined as either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

### **MQCMD\_INQUIRE\_AUTH\_INFO\_NAMES (Inquire Authentication Information Object Names) Response**



The response to the inquire authentication information names (MQCMD\_INQUIRE\_AUTH\_INFO\_NAMES) PCF command consists of the response header followed by a parameter structure giving zero or more names that match the specified authentication information name.



Additionally, on z/OS only, parameter structures, *QSGDispositions* and *AuthInfoTypes* (with the same number of entries as the *AuthInfoNames* structure), are returned. Each entry in this

structure indicates the disposition of the object with the corresponding entry in the *AuthInfoNames* structure.

**Always returned:**

*AuthInfoNames* , *QSGDispositions*, , *AuthInfoTypes*

**Returned if requested:**

None

**Response data**

**AuthInfoNames (MQCFSL)**

List of authentication information object names (parameter identifier: MQCACF\_AUTH\_INFO\_NAMES).



**QSGDispositions (MQCFIL)**

List of queue sharing group dispositions (parameter identifier: MQIACF\_QSG\_DISPS).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be any of the following values:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**AuthInfoTypes (MQCFIL)**

List of authentication information object types (parameter identifier: MQIACH\_AUTH\_INFO\_TYPES).

Specifies the type of the object. This parameter is valid on z/OS only. The value can be any of the following values:

**MQAIT\_CRL\_LDAP**

This defines this authentication information object as specifying an LDAP server containing Certificate Revocation Lists.

**MQAIT\_OCSP**

This value defines this authentication information object as specifying certificate revocation checking using OCSP.

**MQAIT\_IDPW\_OS**

This value defines this authentication information object as specifying certificate revocation checking using user ID and password checking through the operating system.

 **MQCMD\_INQUIRE\_AUTH\_RECS (Inquire Authority Records) on Multiplatforms**

The Inquire Authority Records (MQCMD\_INQUIRE\_AUTH\_RECS) PCF command retrieves authority records associated with a profile name.

**Required parameters**

**Options (MQCFIN)**

Options to control the set of authority records that is returned (parameter identifier: MQIACF\_AUTH\_OPTIONS).

This parameter is required and you must include one of the following two values:

**MQAUTHOPT\_NAME\_ALL\_MATCHING**

Return all profiles the names of which match the specified *ProfileName*. This means that a *ProfileName* of ABCD results in the profiles ABCD, ABC\*, and AB\* being returned (if ABC\* and AB\* have been defined as profiles).

**MQAUTHOPT\_NAME\_EXPLICIT**

Return only those profiles the names of which exactly match the *ProfileName*. No matching generic profiles are returned unless the *ProfileName* is, itself, a generic profile. You cannot specify this value and MQAUTHOPT\_ENTITY\_SET.

and one of the following two values:

**MQAUTHOPT\_ENTITY\_EXPLICIT**

Return all profiles the entity fields of which match the specified *EntityName*. No profiles are returned for any group in which *EntityName* is a member; only the profile defined for the specified *EntityName*.

**MQAUTHOPT\_ENTITY\_SET**

Return the profile the entity field of which matches the specified *EntityName* and the profiles pertaining to any groups in which *EntityName* is a member that contribute to the cumulative authority for the specified entity. You cannot specify this value and MQAUTHOPT\_NAME\_EXPLICIT.

You can also optionally specify:

**MQAUTHOPT\_NAME\_AS\_WILDCARD**

Interpret *ProfileName* as a filter on the profile name of the authority records. If you do not specify this attribute and *ProfileName* contains wildcard characters, it is interpreted as a generic profile and only those authority records where the generic profile names match the value of *ProfileName* are returned.

You cannot specify MQAUTHOPT\_NAME\_AS\_WILDCARD if you also specify MQAUTHOPT\_ENTITY\_SET.

**ProfileName (MQCFST)**

Profile name (parameter identifier: MQCACF\_AUTH\_PROFILE\_NAME).

This parameter is the name of the profile for which to retrieve authorizations. Generic profile names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all profiles having names that start with the selected character string. An asterisk on its own matches all possible names.

If you have defined a generic profile, you can return information about it by not setting MQAUTHOPT\_NAME\_AS\_WILDCARD in *Options*.

If you set *Options* to MQAUTHOPT\_NAME\_AS\_WILDCARD, the only valid value for *ProfileName* is a single asterisk (\*). This means that all authority records that satisfy the values specified in the other parameters are returned.

Do not specify *ProfileName* if the value of *ObjectType* is MQOT\_Q\_MGR.

The profile name is always returned regardless of the attributes requested.

The maximum length of the string is MQ\_AUTH\_PROFILE\_NAME\_LENGTH.

**ObjectType (MQCFIN)**

The type of object referred to by the profile (parameter identifier: MQIACF\_OBJECT\_TYPE).

The value can be any of the following values:

**MQOT\_ALL**

All object types. MQOT\_ALL is the default if you do not specify a value for *ObjectType*.

**MQOT\_AUTH\_INFO**

Authentication information.

**MQOT\_CHANNEL**

Channel object.

**MQOT\_CLNTCONN\_CHANNEL**

Client-connection channel object.

**MQOT\_COMM\_INFO**

Communication information object

**MQOT\_LISTENER**

Listener object.

**MQOT\_NAMELIST**

Namelist.

**MQOT\_PROCESS**

Process.

**MQOT\_Q**

Queue, or queues, that match the object name parameter.

**MQOT\_Q\_MGR**

Queue manager.

**MQOT\_REMOTE\_Q\_MGR\_NAME**

Remote queue manager.

**MQOT\_SERVICE**

Service object.

**MQOT\_TOPIC**


Topic object.

**Optional parameters****EntityName (MQCFST)**

Entity name (parameter identifier: MQCACF\_ENTITY\_NAME).

Depending on the value of *EntityType*, this parameter is either:

- A principal name. This name is the name of a user for whom to retrieve authorizations to the specified object. On IBM MQ for Windows, the name of the principal can optionally include a domain name, specified in this format: `user@domain`.
- A group name. This name is the name of the user group on which to make the inquiry. You can specify one name only and this name must be the name of an existing user group.

 For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following formats:

```
GroupName@domain
domain\GroupName
```

The maximum length of the string is MQ\_ENTITY\_NAME\_LENGTH.

**EntityType (MQCFIN)**

Entity type (parameter identifier: MQIACF\_ENTITY\_TYPE).

The value can be:

**MQZAET\_GROUP**

The value of the **EntityName** parameter refers to a group name.

**MQZAET\_PRINCIPAL**

The value of the **EntityName** parameter refers to a principal name.

**ProfileAttrs (MQCFIL)**

Profile attributes (parameter identifier: MQIACF\_AUTH\_PROFILE\_ATTRS).

The attribute list might specify the following value on its own - the default value if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

or a combination of the following:

**MQCACF\_ENTITY\_NAME**

Entity name.

**MQIACF\_AUTHORIZATION\_LIST**

Authorization list.

**MQIACF\_ENTITY\_TYPE**

Entity type.

**Note:** If an entity is specified by using the parameters MQCACF\_ENTITY\_NAME and MQIACF\_ENTITY\_TYPE, then all the required parameters must be passed in first.

**ServiceComponent (MQCFST)**

Service component (parameter identifier: MQCACF\_SERVICE\_COMPONENT).

If installable authorization services are supported, this parameter specifies the name of the authorization service from which to retrieve authorization.

If you omit this parameter, the authorization inquiry is made to the first installable component for the service.

The maximum length of the string is MQ\_SERVICE\_COMPONENT\_LENGTH.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

**Reason (MQLONG)**

The value can be any of the following values:

**MQRC\_OBJECT\_TYPE\_ERROR**

Invalid object type.

**MQRC\_UNKNOWN\_ENTITY**

User ID not authorized, or unknown.

**MQRCCF\_CFST\_CONFLICTING\_PARM**

Conflicting parameters.

**MQRCCF\_PROFILE\_NAME\_ERROR**

Invalid profile name.

**MQRCCF\_ENTITY\_NAME\_MISSING**

Entity name missing.

**MQRCCF\_OBJECT\_TYPE\_MISSING**

Object type missing.

**MQRCCF\_PROFILE\_NAME\_MISSING**

Profile name missing.

### **MQCMD\_INQUIRE\_AUTH\_RECS (Inquire Authority Records) Response on Multiplatforms**

The response to the Inquire Authority Records (MQCMD\_INQUIRE\_AUTH\_RECS) PCF command consists of the response header followed by the *QMgrName*, *Options*, *ProfileName*, and *ObjectType* structures and the requested combination of attribute parameter structures.

One PCF message is returned for each authority record that is found the profile name of which matches the options specified in the Inquire Authority Records request.

**Always returned:**

*ObjectType*, *Options*, *ProfileName*, *QMgrName*



**Returned if requested:**

*AuthorizationList, EntityName, EntityType*

**Response data****AuthorizationList (MQCFIL)**

Authorization list (parameter identifier: MQIACF\_AUTHORIZATION\_LIST).

This list can contain zero or more authorization values. Each returned authorization value means that any user ID in the specified group or principal has the authority to perform the operation defined by that value. The value can be any of the following values:

**MQAUTH\_NONE**

The entity has authority set to 'none'.

**MQAUTH\_ALT\_USER\_AUTHORITY**

Specify an alternate user ID on an MQI call.

**MQAUTH\_BROWSE**

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

**MQAUTH\_CHANGE**

Change the attributes of the specified object, using the appropriate command set.

**MQAUTH\_CLEAR**

Clear a queue.

**MQAUTH\_CONNECT**

Connect the application to the specified queue manager by issuing an MQCONN call.

**MQAUTH\_CREATE**

Create objects of the specified type using the appropriate command set.

**MQAUTH\_DELETE**

Delete the specified object using the appropriate command set.

**MQAUTH\_DISPLAY**

Display the attributes of the specified object using the appropriate command set.

**MQAUTH\_INPUT**

Retrieve a message from a queue by issuing an MQGET call.

**MQAUTH\_INQUIRE**

Make an inquiry on a specific queue by issuing an MQINQ call.

**MQAUTH\_OUTPUT**

Put a message on a specific queue by issuing an MQPUT call.

**MQAUTH\_PASS\_ALL\_CONTEXT**

Pass all context.

**MQAUTH\_PASS\_IDENTITY\_CONTEXT**

Pass the identity context.

**MQAUTH\_SET**

Set attributes on a queue from the MQI by issuing an MQSET call.

**MQAUTH\_SET\_ALL\_CONTEXT**

Set all context on a queue.

**MQAUTH\_SET\_IDENTITY\_CONTEXT**

Set the identity context on a queue.

**MQAUTH\_CONTROL**

For listeners and services, start and stop the specified channel, listener, or service.

For channels, start, stop, and ping the specified channel.

For topics, define, alter, or delete subscriptions.

**MQAUTH\_CONTROL\_EXTENDED**

Reset or resolve the specified channel.

**MQAUTH\_PUBLISH**

Publish to the specified topic.

**MQAUTH\_SUBSCRIBE**

Subscribe to the specified topic.

**MQAUTH\_RESUME**

Resume a subscription to the specified topic.

**MQAUTH\_SYSTEM**

Use queue manager for internal system operations.

**MQAUTH\_ALL**

Use all operations applicable to the object.

**MQAUTH\_ALL\_ADMIN**

Use all operations applicable to the object.

**MQAUTH\_ALL\_MQI**

Use all MQI calls applicable to the object.

Use the *Count* field in the MQCFIL structure to determine how many values are returned.

**EntityName (MQCFST)**

Entity name (parameter identifier: MQCACF\_ENTITY\_NAME).

This parameter can either be a principal name or a group name.

The maximum length of the string is MQ\_ENTITY\_NAME\_LENGTH.

**EntityType (MQCFIN)**

Entity type (parameter identifier: MQIACF\_ENTITY\_TYPE).

The value can be:

**MQZAET\_GROUP**

The value of the **EntityName** parameter refers to a group name.

**MQZAET\_PRINCIPAL**

The value of the **EntityName** parameter refers to a principal name.

**MQZAET\_UNKNOWN**

On Windows, an authority record still exists from a previous queue manager which did not originally contain entity type information.

**ObjectType (MQCFIN)**

Object type (parameter identifier: MQIACF\_OBJECT\_TYPE).

The value can be:

**MQOT\_AUTH\_INFO**

Authentication information.

**MQOT\_CHANNEL**

Channel object.

**MQOT\_CLNTCONN\_CHANNEL**

Client-connection channel object.

**MQOT\_COMM\_INFO**

Communication information object

**MQOT\_LISTENER**

Listener object.

**MQOT\_NAMELIST**

Namelist.

**MQOT\_PROCESS**

Process.

**MQOT\_Q**

Queue, or queues, that match the object name parameter.

**MQOT\_Q\_MGR**

Queue manager.

**MQOT\_REMOTE\_Q\_MGR\_NAME**

Remote queue manager.

**MQOT\_SERVICE**

Service object.

**MQOT\_TOPIC**

Topic object.

**Options (MQCFIN)**

Options used to indicate the level of information that is returned (parameter identifier: MQIACF\_AUTH\_OPTIONS).

**ProfileName (MQCFST)**

Profile name (parameter identifier: MQCACF\_AUTH\_PROFILE\_NAME).

The maximum length of the string is MQ\_AUTH\_PROFILE\_NAME\_LENGTH.

**QMgrName (MQCFST)**

Name of the queue manager on which the Inquire command is issued (parameter identifier: MQCA\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**Multi MQCMD\_INQUIRE\_AUTH\_SERVICE (Inquire Authority Service) on Multiplatforms**

The Inquire Authority Service (MQCMD\_INQUIRE\_AUTH\_SERVICE) PCF command retrieves information about the level of function supported by installed authority managers.

**Required parameters****AuthServiceAttrs (MQCFIL)**

Authority service attributes (parameter identifier: MQIACF\_AUTH\_SERVICE\_ATTRS).

The attribute list might specify the following value on its own - default value if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

or a combination of the following:

**MQIACF\_INTERFACE\_VERSION**

Current interface version of the authority service.

**MQIACF\_USER\_ID\_SUPPORT**

Whether the authority service supports user IDs.

**Optional parameters****ServiceComponent (MQCFST)**

Name of authorization service (parameter identifier: MQCACF\_SERVICE\_COMPONENT).

The name of the authorization service which is to handle the Inquire Authority Service command.

If this parameter is omitted, or specified as a blank or null string, the inquire function is called in each installed authorization service in reverse order to the order in which the services have been installed, until all authorization services have been called or until one returns a value of MQZCI\_STOP in the Continuation field.

The maximum length of the string is MQ\_SERVICE\_COMPONENT\_LENGTH.

## Error codes

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRC\_SELECTOR\_ERROR**

Attribute selector not valid.

#### **MQRC\_UNKNOWN\_COMPONENT\_NAME**

Unknown service component name.

## **MQCMD\_INQUIRE\_AUTH\_SERVICE (Inquire Authority Service) Response on Multiplatforms**

The response to the Inquire Authority Service (MQCMD\_INQUIRE\_AUTH\_SERVICE) PCF command consists of the response header followed by the *ServiceComponent* structure and the requested combination of attribute parameter structures.

### Always returned:

*ServiceComponent*

### Returned if requested:

*InterfaceVersion, UserIDSupport*

## Response data

### InterfaceVersion (MQCFIN)

Interface version (parameter identifier: MQIACF\_INTERFACE\_VERSION).

This parameter is the current interface version of the OAM.

### ServiceComponent (MQCFSL)

Name of authorization service (parameter identifier: MQCACF\_SERVICE\_COMPONENT).

If you included a specific value for *ServiceComponent* on the Inquire Authority Service command, this field contains the name of the authorization service that handled the command. If you did not include a specific value for *ServiceComponent* on the Inquire Authority Service command, the list contains the names of all the installed authorization services.

If there is no OAM or if the OAM requested in the *ServiceComponent* does not exist this field is blank.

The maximum length of each element in the list is MQ\_SERVICE\_COMPONENT\_LENGTH.

### UserIDSupport (MQCFIN)

User ID support (parameter identifier: MQIACF\_USER\_ID\_SUPPORT).

The value can be:

#### **MQIDSUPP\_YES**

The authority service supports user IDs.

#### **MQIDSUPP\_NO**

The authority service does not support user IDs.

## **MQCMD\_INQUIRE\_CF\_STRUC (Inquire CF Structure) on z/OS**

The Inquire CF Structure (MQCMD\_INQUIRE\_CF\_STRUC) PCF command returns information about the attributes of one or more CF application structures.

**Note:** This command is supported only on z/OS when the queue manager is a member of a queue sharing group.

## Required parameters

### **CFStrucName (MQCFST)**

CF Structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME).

Specifies the name of the CF application structure about which information is to be returned.

Generic CF structure names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all CF application structures having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length is MQ\_CF\_STRUC\_NAME\_LENGTH.

## Optional parameters

### **CFStrucAttrs (MQCFIL)**

CF application structure attributes (parameter identifier: MQIACF\_CF\_STRUC\_ATTRS).

The attribute list might specify the following value on its own - default value used if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes.

or a combination of the following:

#### **MQCA\_ALTERATION\_DATE**

The date on which the definition was last altered.

#### **MQCA\_ALTERATION\_TIME**

The time at which the definition was last altered.

#### **MQIA\_CF\_CFCNLOS**

The action to be taken when the queue manager loses connectivity to the CF application structure.

#### **MQIA\_CF\_LEVEL**

Functional capability level for the CF application structure.

#### **MQIA\_CF\_OFFLOAD**

The shared message data set OFFLOAD property for the CF application structure.

#### **MQIA\_CF\_RECOVER**

Whether CF recovery for the application structure is supported.

#### **MQIA\_CF\_RECAUTO**

Whether automatic recovery action is taken when a structure is failed or when a queue manager loses connectivity to the structure and no systems in the SysPlex have connectivity to the Coupling Facility the structure is located in.

#### **MQIACF\_CF\_SMDS\_BLOCK\_SIZE**

The shared message data set DSGROUP property for the CF application structure.

#### **MQIA\_CF\_SMDS\_BUFFERS**

The shared message data set DSGROUP property for the CF application structure.

#### **MQIACF\_CF\_SMDS\_EXPAND**

The shared message data set DSEXPAND property for the CF application structure.

#### **MQCACF\_CF\_SMDS\_GENERIC\_NAME**

The shared message data set DSBUFS property for the CF application structure.

#### **MQCA\_CF\_STRUC\_DESC**

Description of CF application structure.

#### **MQCA\_CF\_STRUC\_NAME**

Name of CF application structure.

### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *CFStrucAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the

command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 1523 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

#### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *CFStrucAttrs* except MQCA\_CF\_STRUC\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 1530 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

### **MQCMD\_INQUIRE\_CF\_STRUC (Inquire CF Structure) Response on z/OS**

The response to the Inquire CF Structure (MQCMD\_INQUIRE\_CF\_STRUC) PCF command consists of the response header followed by the *CFStrucName* structure and the requested combination of attribute parameter structures.

If a generic CF application structure name was specified, one such message is generated for each CF application structure found.

#### **Always returned:**

*CFStrucName*

#### **Returned if requested:**

*AlterationDate, AlterationTime, CFConlos, CFLevel, CFStrucDesc, DSBLOCK, DSBUFS, DSEXPAND, DSGROUP, OFFLD1SZ, OFFLD12SZ, OFFLD3SZ, OFFLD1TH, OFFLD2TH, OFFLD3TH, Offload, RCVDATE, RCVTIME, Recauto, Recovery*

### **Response data**

#### **AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date on which the definition was last altered, in the form yyyy-mm-dd.

The maximum length of the string is MQ\_DATE\_LENGTH.

#### **AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time at which the definition was last altered, in the form hh.mm.ss.

The maximum length of the string is MQ\_TIME\_LENGTH.

#### **CFConlos (MQCFIN)**

The CFConlos property (parameter identifier: MQIA\_CF\_CFCNLOS).

Specifies the action to be taken when a queue manager loses connectivity to the CF structure. The value can be any of the following values:

##### **MQCFCONLOS\_TERMINATE**

The queue manager will terminate when connectivity to the structure is lost.

##### **MQCFCONLOS\_TOLERATE**

The queue manager will tolerate loss of connectivity to the structure without terminating.

##### **MQCFCONLOS\_ASQMGR**

The action taken is based on the setting of the CFCNLOS queue manager attribute

This parameter is only valid from CFLEVEL(5).

#### **CFLevel (MQCFIN)**

The functional capability level for this CF application structure (parameter identifier: MQIA\_CF\_LEVEL).

Specifies the functional capability level for the CF application structure. The value can be any of the following values:

**1**

A CF structure that can be "auto-created" by a queue manager at command level 520.

**2**

A CF structure at command level 520 that can only be created or deleted by a queue manager at command level 530 or greater. This level is the default *CFLevel* for queue managers at command level 530 or greater.

**3**

A CF structure at command level 530. This *CFLevel* is required if you want to use persistent messages on shared queues, or for message grouping, or both.

**4**

A CF structure at command level 600. This *CFLevel* can be used for persistent messages or for messages longer than 64 512 bytes.

**5**

A CF structure at command level 710. This *CFLevel* supports shared message data sets (SMDS) and Db2 for offloading messages.

Structures are required to be at CFLEVEL(5) to support toleration of loss of connectivity.

#### **CFStrucDesc (MQCFST)**

The description of the CF structure (parameter identifier: MQCA\_CF\_STRUC\_DESC).

The maximum length is MQ\_CF\_STRUC\_DESC\_LENGTH.

#### **CFStrucName (MQCFST)**

CF Structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length is MQ\_CF\_STRUC\_NAME\_LENGTH.

#### **DSBLOCK (MQCFIN)**

The CF DSBLOCK property (parameter identifier: MQIACF\_CF\_SMDS\_BLOCK\_SIZE).

The returned value is one of the following constants: MQDSB\_8K, MQDSB\_16K, MQDSB\_32K, MQDSB\_64K, MQDSB\_128K, MQDSB\_256K, MQDSB\_512K, MQDSB\_1024K, MQDSB\_1M.

#### **DSBUFS (MQCFIN)**

The CF DSBUFS property (parameter identifier: MQIA\_CF\_SMDS\_BUFFERS).

The returned value is in the range 0 - 9999.

The value is the number of buffers to be allocated in each queue manager for accessing shared message data sets. The size of each buffer is equal to the logical block size.

#### **DSEXPAND (MQCFIN)**

The CF DSEXPAND property (parameter identifier: MQIACF\_CF\_SMDS\_EXPAND).

##### **MQDSE\_YES**

The data set can be expanded.

##### **MQDSE\_NO**

The data set cannot be expanded.

##### **MQDSE\_DEFAULT**

Only returned on Inquire CF Struct when not explicitly set

#### **DSGROUP (MQCFST)**

The CF DSGROUP property (parameter identifier: MQCACF\_CF\_SMDS\_GENERIC\_NAME).

The returned value is a string containing a generic data set name used for the group of shared message data sets associated with this CF structure.

**OFFLD1SZ (MQCFST)**

The CF OFFLD1SZ property (parameter identifier: MQCACF\_CF\_OFFLOAD\_SIZE1).

The returned value is a string in the range 0K - 64K.

Returned if the MQIACF\_ALL or MQIA\_CF\_OFFLOAD parameters are specified.

The maximum length is 3.

**OFFLD2SZ (MQCFST)**

The CF OFFLD2SZ property (parameter identifier: MQCACF\_CF\_OFFLOAD\_SIZE2).

The returned value is a string in the range 0K - 64K.

Returned if the MQIACF\_ALL or MQIA\_CF\_OFFLOAD parameters are specified.

The maximum length is 3.

**OFFLD3SZ (MQCFST)**

The CF OFFLD3SZ property (parameter identifier: MQCACF\_CF\_OFFLOAD\_SIZE3).

The returned value is a string in the range 0K - 64K.

Returned if the MQIACF\_ALL or MQIA\_CF\_OFFLOAD parameters are specified.

The maximum length is 3.

**OFFLD1TH (MQCFIN)**

The CF OFFLD1TH property (parameter identifier: MQIA\_CF\_OFFLOAD\_THRESHOLD1).

The returned value is in the range 0 - 100.

Returned if the MQIACF\_ALL or MQIA\_CF\_OFFLOAD parameters are specified.

**OFFLD2TH (MQCFIN)**

The CF OFFLD2TH property (parameter identifier: MQIA\_CF\_OFFLOAD\_THRESHOLD2).

The returned value is in the range 0 - 100.

Returned if the MQIACF\_ALL or MQIA\_CF\_OFFLOAD parameters are specified.

**OFFLD3TH (MQCFIN)**

The CF OFFLD3TH property (parameter identifier: MQIA\_CF\_OFFLOAD\_THRESHOLD3).

The returned value is in the range 0 - 100.

Returned if the MQIACF\_ALL or MQIA\_CF\_OFFLOAD parameters are specified.

**Offload (MQCFIN)**

The CF OFFLOAD property (parameter identifier: MQIA\_CF\_OFFLOAD).

The returned values can be:

**MQCFOFFLD\_DB2**

Large shared messages can be stored in Db2.

**MQCFOFFLD\_SMDS**

Large shared messages can be stored in z/OS shared message data sets.

**MQCFOFFLD\_NONE**

Used when the property *Offload* has not been explicitly set.

**RCVDATE (MQCFST)**

The recovery start date (parameter identifier: MQCACF\_RECOVERY\_DATE).

If recovery is currently enabled for the data set, this indicates the date when it was activated, in the form yyyy-mm-dd. If recovery is not enabled, this is displayed as RCVDATE().

**RCVTIME (MQCFST)**

The recovery start time (parameter identifier: MQCACF\_RECOVERY\_TIME).



If recovery is currently enabled for the data set, this indicates the time when it was activated, in the form hh.mm.ss. If recovery is not enabled, this is displayed as RCVTIME().

### **Recauto (MQCFIN)**

Recauto (parameter identifier: MQIA\_CF\_RECAUTO).

Indicates whether automatic recovery action is taken when a queue manager detects that the structure is failed, or when a queue manager loses connectivity to the structure and no systems in the SysPlex have connectivity to the Coupling Facility that the structure is allocated in. The value can be:

#### **MQRECAUTO\_YES**

The structure and associated shared message data sets which also need recovery will be automatically recovered.

#### **MQRECAUTO\_NO**

The structure will not be automatically recovered.

### **Recovery (MQCFIN)**

Recovery (parameter identifier: MQIA\_CF\_RECOVER).

Specifies whether CF recovery is supported for the application structure. The value can be:

#### **MQCFR\_YES**

Recovery is supported.

#### **MQCFR\_NO**

Recovery is not supported.

## **z/OS MQCMD\_INQUIRE\_CF\_STRUC\_NAMES (Inquire CF Structure Names) on z/OS**

The Inquire CF Structure Names (MQCMD\_INQUIRE\_CF\_STRUC\_NAMES) PCF command inquires for a list of CF application structure names that match the generic CF structure name specified.

**Note:** This command is supported only on z/OS when the queue manager is a member of a queue sharing group.

### **Required parameters**

#### **CFStrucName (MQCFST)**

CF Structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME).

Specifies the name of the CF application structure about which information is to be returned.

Generic CF structure names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all CF application structures having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length is MQ\_CF\_STRUC\_NAME\_LENGTH.

## **z/OS MQCMD\_INQUIRE\_CF\_STRUC\_NAMES (Inquire CF Structure Names)**

### **Response on z/OS**

The response to the Inquire CF Structure Names (MQCMD\_INQUIRE\_CF\_STRUC\_NAMES) PCF command consists of the response header followed by a single parameter structure giving zero or more names that match the specified CF application structure name.

#### **Always returned:**

*CFStrucNames*

#### **Returned if requested:**

None

## Response data

### CFStrucNames (MQCFSL)

List of CF application structure names (parameter identifier: MQCACF\_CF\_STRUC\_NAMES).

### **MQCMD\_INQUIRE\_CF\_STRUC\_STATUS (Inquire CF Structure Status) on z/OS**

The Inquire CF Structure Status (MQCMD\_INQUIRE\_CF\_STRUC\_STATUS) PCF command inquires about the status of a CF application structure.

**Note:** This command is supported only on z/OS when the queue manager is a member of a queue sharing group.

## Required parameters

### CFStrucName (MQCFST)

CF Structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME).

Specifies the name of the CF application structure for which status information is to be returned.

Generic CF structure names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all CF application structures having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length is MQ\_CF\_STRUC\_NAME\_LENGTH.

## Optional parameters

### CFStatusType (MQCFIN)

Status information type (parameter identifier: MQIACF\_CF\_STATUS\_TYPE).

Specifies the type of status information you want to be returned. You can specify one of the following:

#### **MQIACF\_CF\_STATUS\_SUMMARY**

Summary status information for the CF application structure. MQIACF\_CF\_STATUS\_SUMMARY is the default.

#### **MQIACF\_CF\_STATUS\_CONNECT**

Connection status information for each CF application structure for each active queue manager.

#### **MQIACF\_CF\_STATUS\_BACKUP**

Backup status information for each CF application structure.

#### **MQIACF\_CF\_STATUS\_SMDS**

Shared message data set information for each CF application structure.

### IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter in the response data except MQIACF\_CF\_STATUS\_TYPE. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

### StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter in the response data except MQCA\_CF\_STRUC\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

## **MQCMD\_INQUIRE\_CF\_STRUC\_STATUS (Inquire CF Structure Status)**

### **Response on z/OS**

The response to the Inquire CF Structure Status (MQCMD\_INQUIRE\_CF\_STRUC\_STATUS) PCF command consists of the response header followed by the *CFStrucName* and *CFStatusType* structures and a set of attribute parameter structures determined by the value of *CFStatusType* in the Inquire command.

#### **Always returned:**

*CFStrucName*, *CFStatusType*.

*CFStatusType* specifies the type of status information being returned. The value can be any of the following values:

#### **MQIACF\_CF\_STATUS\_SUMMARY**

Summary status information for the CF application structure. This is the default.

#### **MQIACF\_CF\_STATUS\_CONNECT**

Connection status information for each CF application structure for each active queue manager.

#### **MQIACF\_CF\_STATUS\_BACKUP**

Backup status information for each CF application structure.

#### **MQIACF\_CF\_STATUS\_SMDS**

Shared message data set information for each CF application structure.

#### **Returned if *CFStatusType* is MQIACF\_CF\_STATUS\_SUMMARY:**

*CFStrucStatus*, *CFStrucType*, *EntriesMax*, *EntriesUsed*, *FailDate*, *FailTime*, *OffLdUse*, *SizeMax*, *SizeUsed*

#### **Returned if *CFStatusType* is MQIACF\_CF\_STATUS\_CONNECT:**

*CFStrucStatus*, *FailDate*, *FailTime*, *QMgrName*, *SysName*

#### **Returned if *CFStatusType* is MQIACF\_CF\_STATUS\_BACKUP:**

*BackupDate*, *BackupEndRBA*, *BackupSize*, *BackupStartRBA*, *BackupTime*, *CFStrucStatus*, *FailDate*, *FailTime*, *LogQMgrNames*, *QmgrName*

#### **Returned if *CFStatusType* is MQIACF\_CF\_STATUS\_SMDS:**

*Access*, *FailDate*, *FailTime*, *RcvDate*, *RcvTime*, *CFStrucStatus*

### **Response data**

#### **Access (MQCFIN)**

Availability of the shared message data set (parameter identifier: MQIACF\_CF\_STRUC\_ACCESS).

#### **MQCFACCESS\_ENABLED**

The shared message data set is either available for use, or is to be enabled after previously being disabled, or access to the shared message data set is to be retried following an error.

#### **MQCFACCESS\_SUSPENDED**

The shared message data set is unavailable because of an error.

#### **MQCFACCESS\_DISABLED**

The shared message data set is either disabled, or is to be set as disabled.

#### **BackupDate (MQCFST)**

The date, in the form yyyy-mm-dd, on which the last successful backup was taken for this CF application structure (parameter identifier: MQCACF\_BACKUP\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

#### **BackupEndRBA (MQCFST)**

The backup data set end RBA for the end of the last successful backup taken for this CF application structure (parameter identifier: MQCACF\_CF\_STRUC\_BACKUP\_END).

The maximum length of the string is MQ\_RBA\_LENGTH.

#### **BackupSize (MQCFIN)**

The size, in megabytes, of the last successful backup taken for this CF application structure (parameter identifier: MQIACF\_CF\_STRUC\_BACKUP\_SIZE).

**BackupStartRBA (MQCFST)**

The backup data set start RBA for the start of the last successful backup taken for this CF application structure (parameter identifier: MQCACF\_CF\_STRUC\_BACKUP\_START).

The maximum length of the string is MQ\_RBA\_LENGTH.

**BackupTime (MQCFST)**

The end time, in the form hh.mm.ss, of the last successful backup taken for this CF application structure (parameter identifier: MQCACF\_BACKUP\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**CFStatusType (MQCFIN)**

Status information type (parameter identifier: MQIACF\_CF\_STATUS\_TYPE).

Specifies the type of status information being returned. The value can be any of the following values:

**MQIACF\_CF\_STATUS\_SUMMARY**

Summary status information for the CF application structure. MQIACF\_CF\_STATUS\_SUMMARY is the default.

**MQIACF\_CF\_STATUS\_CONNECT**

Connection status information for each CF application structure for each active queue manager.

**MQIACF\_CF\_STATUS\_BACKUP**

Back up status information for each CF application structure.

**MQIACF\_CF\_STATUS\_SMDS**

Shared message data set information for each CF application structure.

**CFStrucName (MQCFST)**

CF Structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length is MQ\_CF\_STRUC\_NAME\_LENGTH.

**CFStrucStatus (MQCFIN)**

CF Structure status (parameter identifier: MQIACF\_CF\_STRUC\_STATUS).

The status of the CF application structure.

If *CFStatusType* is MQIACF\_CF\_STATUS\_SUMMARY, the value can be:

**MQCFSTATUS\_ACTIVE**

The structure is active.

**MQCFSTATUS\_FAILED**

The structure has failed.

**MQCFSTATUS\_NOT\_FOUND**

The structure is not allocated in the CF, but has been defined to Db2.

**MQCFSTATUS\_IN\_BACKUP**

The structure is in the process of being backed up.

**MQCFSTATUS\_IN\_RECOVER**

The structure is in the process of being recovered.

**MQCFSTATUS\_UNKNOWN**

The status of the CF structure is unknown because, for example, Db2 might be unavailable.

If *CFStatusType* is MQIACF\_CF\_STATUS\_CONNECT, the value can be:

**MQCFSTATUS\_ACTIVE**

The structure is connected to this queue manager.

**MQCFSTATUS\_FAILED**

The queue manager connection to this structure has failed.

**MQCFSTATUS\_NONE**

The structure has never been connected to this queue manager.

If *CFStatusType* is MQIACF\_CF\_STATUS\_BACKUP, the value can be:

**MQCFSTATUS\_ACTIVE**

The structure is active.

**MQCFSTATUS\_FAILED**

The structure has failed.

**MQCFSTATUS\_NONE**

The structure has never been backed up.

**MQCFSTATUS\_IN\_BACKUP**

The structure is in the process of being backed up.

**MQCFSTATUS\_IN\_RECOVER**

The structure is in the process of being recovered.

If *CFStatusType* is MQIACF\_CF\_STATUS\_SMDS, the value can be:

**MQCFSTATUS\_ACTIVE**

The shared message data set is available for normal use

**MQCFSTATUS\_FAILED**

The shared message data set is in an unusable state and probably requires recovery.

**MQCFSTATUS\_IN\_RECOVER**

The shared message data set is in the process of being recovered (by way of a RECOVER CFSTRUCT command).

**MQCFSTATUS\_NOT\_FOUND**

The data set has never been used, or the attempt to open it for the first time failed.

**MQCFSTATUS\_RECOVERED**

The data set has been recovered or otherwise repaired, and is ready for use again, but requires some restart processing the next time it is opened. This restart processing ensures that obsolete references to any deleted messages have been removed from the coupling facility structure before the data set is made available again. The restart processing also rebuilds the data set space map.

**MQCFSTATUS\_EMPTY**

The data set contains no messages. The data set is put into this state if it is closed normally by the owning queue manager at a time when it does not contain any messages. It can also be put into EMPTY state when the previous data set contents are to be discarded because the application structure has been emptied (using **RECOVER CFSTRUCT** with TYPE PURGE or, for a nonrecoverable structure only, by deleting the previous instance of the structure). The next time the data set is opened by its owning queue manager, the space map is reset to empty, and the status is changed to ACTIVE. As the previous data set contents are no longer required, a data set in this state can be replaced with a newly allocated data set, for example to change the space allocation or move it to another volume.

**MQCFSTATUS\_NEW**

The data set is being opened and initialized for the first time, ready to be made active.

**CFStructType (MQCFIN)**

CF Structure type (parameter identifier: MQIACF\_CF\_STRUC\_TYPE).

The value can be:

**MQCFTYPE\_ADMIN**

MQCFTYPE\_ADMIN is the CF administration structure.

**MQCFTYPE\_APPL**

MQCFTYPE\_APPL is a CF application structure.

**EntriesMax (MQCFIN)**

Number of CF list entries defined for this CF application structure (parameter identifier: MQIACF\_CF\_STRUC\_ENTRIES\_MAX).

**EntriesUsed (MQCFIN)**

Number of CF list entries defined for this CF application structure that are in use (parameter identifier: MQIACF\_CF\_STRUC\_ENTRIES\_USED).

**FailDate (MQCFST)**

The date, in the form yyyy-mm-dd, on which this CF application structure failed (parameter identifier: MQCACF\_FAIL\_DATE).

If *CFStatusType* is MQIACF\_CF\_STATUS\_CONNECT, it is the date on which the queue manager lost connectivity to this application structure. For the other values of *CFStatusType*, it is the date on which this CF application structure failed. This parameter is only applicable when *CFStrucStatus* is MQCFSTATUS\_FAILED or MQCFSTATUS\_IN\_RECOVER.

The maximum length of the string is MQ\_DATE\_LENGTH.

**FailTime (MQCFST)**

The time, in the form hh.mm.ss, that this CF application structure failed (parameter identifier: MQCACF\_FAIL\_TIME).

If *CFStatusType* is MQIACF\_CF\_STATUS\_CONNECT, it is the time that the queue manager lost connectivity to this application structure. For the other values of *CFStatusType*, it is the time that this CF application structure failed. This parameter is only applicable when *CFStrucStatus* is MQCFSTATUS\_FAILED or MQCFSTATUS\_IN\_RECOVER.

The maximum length of the string is MQ\_TIME\_LENGTH.

**LogQMgrNames (MQCFSL)**

A list of queue managers, the logs of which are required to perform a recovery (parameter identifier: MQCACF\_CF\_STRUC\_LOG\_Q\_MGRS).

The maximum length of each name is MQ\_Q\_MGR\_NAME\_LENGTH.

**OffLdUse (MQCFIN)**

Offload usage (parameter identifier: MQIA\_CF\_OFFLDUSE).

Indicates whether any offloaded large message data might currently exist in shared message data sets, Db2, or both. The value can be any of the following values:

**MQCFOFFLD\_DB2**

Large shared messages are stored in Db2.

**MQCFOFFLD\_SMDS**

Large shared messages are stored in z/OS shared message data sets.

**MQCFOFFLD\_NONE**

Use on DISPLAY CFSTRUCT when the property has not been explicitly set.

**MQCFOFFLD\_BOTH**

There might be large shared messages stored in both Db2, and shared message data sets.

Value cannot be set unless CFLEVEL(5) is defined.

**QMgrName (MQCFST)**

Queue manager name (parameter identifier: MQCA\_Q\_MGR\_NAME).

This parameter is the name of the queue manager. If *CFStatusType* is MQIACF\_CF\_STATUS\_BACKUP, it is the name of the queue manager that took the last successful backup.

The maximum length is MQ\_Q\_MGR\_NAME\_LENGTH.

**RcvDate (MQCFST)**

The recovery start date (parameter identifier: MQCACF\_RECOVERY\_DATE).

If recovery is currently enabled for the data set, this indicates the date when it was activated, in the form yyyy-mm-dd.

**RcvTime (MQCFST)**

The recovery start time (parameter identifier: MQCACF\_RECOVERY\_TIME).

If recovery is currently enabled for the data set, this indicates the time when it was activated, in the form hh.mm.ss.

**SizeMax (MQCFIN)**

Size of the CF application structure (parameter identifier: MQIACF\_CF\_STRUC\_SIZE\_MAX).

This parameter is the size, in kilobytes, of the CF application structure.

**SizeUsed (MQCFIN)**

Percentage of the CF application structure that is in use (parameter identifier: MQIACF\_CF\_STRUC\_SIZE\_USED).

This parameter is the percentage of the size of the CF application structure that is in use.

**SysName (MQCFST)**

Queue manager name (parameter identifier: MQCACF\_SYSTEM\_NAME).

This parameter is the name of the z/OS image of the queue manager that last connected to the CF application structure.

The maximum length is MQ\_SYSTEM\_NAME\_LENGTH.

**SizeMax (MQCFIN)**

Size of the CF application structure (parameter identifier: MQIACF\_CF\_STRUC\_SIZE\_MAX).

This parameter is the size, in kilobytes, of the CF application structure.

**MQCMD\_INQUIRE\_CHANNEL (Inquire Channel)**

The Inquire Channel (MQCMD\_INQUIRE\_CHANNEL) PCF command inquires about the attributes of IBM MQ channel definitions.

**Required parameters****ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all channels having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**Optional parameters****ChannelAttrs (MQCFIL)**

Channel attributes (parameter identifier: MQIACF\_CHANNEL\_ATTRS).

The attribute list can specify the following value on its own. This is also the default value used if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

Alternatively, the attribute list can specify a combination of the parameters in the following table:

<i>Table 204. Optional parameters for ChannelAttrs</i>										
Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP	
<b>MQCA_ALTERATION_DATE</b> Date on which the definition was last altered	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 204. Optional parameters for ChannelAttrs (continued)

Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQCA_ALTERATION_TIME</b> Time at which the definition was last altered	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>MQCA_CERT_LABEL</b> Certificate label	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>MQCA_CLUSTER_NAME</b> Name of local queue manager							✓	✓	
<b>MQCA_CLUSTER_NAMELIST</b> Name of local queue manager							✓	✓	
<b>MQCA_Q_MGR_NAME</b> Name of local queue manager					✓				
<b>MQCACH_CHANNEL_NAME</b> Channel name. You cannot use this attribute as a filter keyword.	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>MQCACH_CONNECTION_NAME</b> Connection name	✓	✓		✓	✓		✓	✓	
<b>MQCACH_DESC</b> Description	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>MQCACH_LOCAL_ADDRESS</b> Local communications address for the channel	✓	✓		✓	✓		✓	✓	✓
Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQCACH_MCA_NAME</b> Message channel agent name	✓	✓		✓			✓		
<b>MQCACH_MCA_USER_ID</b> MCA user identifier	✓	✓	✓	✓		✓	✓	✓	✓
<b>MQCACH_MODE_NAME</b> Mode name	✓	✓		✓	✓		✓	✓	
<b>MQCACH_MR_EXIT_NAME</b> Message-retry exit name			✓	✓				✓	
<b>MQCACH_MR_EXIT_USER_DATA</b> Message-retry exit name			✓	✓				✓	
<b>MQCACH_MSG_EXIT_NAME</b> Message exit name	✓	✓	✓	✓			✓	✓	
<b>MQCACH_MSG_EXIT_USER_DATA</b> Message exit user data	✓	✓	✓	✓			✓	✓	
<b>MQCACH_PASSWORD</b> Password	✓	✓		✓	✓		✓		
<b>MQCACH_RCV_EXIT_NAME</b> Receive exit name	✓	✓	✓	✓	✓	✓	✓	✓	




Table 204. Optional parameters for ChannelAttrs (continued)

Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQCACH_RCV_EXIT_USER_DATA</b> Receive exit user data	✓	✓	✓	✓	✓	✓	✓	✓	
Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQCACH_SEC_EXIT_NAME</b> Security exit name	✓	✓	✓	✓	✓	✓	✓	✓	
<b>MQCACH_SEC_EXIT_USER_DATA</b> Security exit user data	✓	✓	✓	✓	✓	✓	✓	✓	
<b>MQCACH_SEND_EXIT_NAME</b> Send exit name	✓	✓	✓	✓	✓	✓	✓	✓	
<b>MQCACH_SEND_EXIT_USER_DATA</b> Send exit user data	✓	✓	✓	✓	✓	✓	✓	✓	
<b>MQCACH_SSL_CIPHER_SPEC</b> TLS cipher spec	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>MQCACH_SSL_PEER_NAME</b> TLS peer name	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>MQCACH_TP_NAME</b> Transaction program name	✓	✓		✓	✓	✓	✓	✓	
<b>MQCACH_TP_ROOT</b> Topic root for AMQP channel									✓
<b>MQCACH_USER_ID</b> User identifier	✓	✓		✓	✓		✓		
<b>MQCACH_XMIT_Q_NAME</b> Transmission queue name	✓	✓							
Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQIA_MONITORING_CHANNEL</b> Online monitoring data collection	✓	✓	✓	✓		✓	✓	✓	
<b>MQIA_PROPERTY_CONTROL</b> Property control attribute	✓	✓					✓	✓	
<b>MQIA_STATISTICS_CHANNEL</b> Online statistics collection	✓	✓	✓	✓			✓	✓	
<b>MQIA_USE_DEAD_LETTER_Q</b> Determines whether the dead-letter queue is used when messages cannot be delivered by channels.	✓	✓	✓	✓			✓	✓	
<b>MQIACH_AMQP_KEEP_ALIVE</b> AMQP channel keep alive interval									✓
<b>MQIACH_BATCH_HB</b> Value to use for batch heartbeating	✓	✓					✓	✓	
<b>MQIACH_BATCH_INTERVAL</b> Batch wait interval (seconds)	✓	✓					✓	✓	

Table 204. Optional parameters for ChannelAttrs (continued)

Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQIACH_BATCH_DATA_LIMIT</b> Batch data limit (kilobytes)	✓	✓					✓	✓	
<b>MQIACH_CHANNEL_TYPE</b> Channel type	✓	✓	✓	✓	✓	✓	✓	✓	✓
> <b>MQIACH_CLIENT_CHANNEL_WEIGHT</b> Client Channel Weight					✓				
Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQIACH_CLWL_CHANNEL_PRIORITY</b> Cluster workload channel priority							✓	✓	
<b>MQIACH_CLWL_CHANNEL_RANK</b> Cluster workload channel rank							✓	✓	
<b>MQIACH_CLWL_CHANNEL_WEIGHT</b> Cluster workload channel weight							✓	✓	
<b>MQIACH_CONNECTION_AFFINITY</b> Connection Affinity					✓				
<b>MQIACH_DATA_CONVERSION</b> Whether sender must convert application data	✓	✓					✓	✓	
<b>MQIACH_DEF_RECONNECT</b> Default reconnection option					✓				
<b>MQIACH_DISC_INTERVAL</b> Disconnection interval	✓	✓				✓	✓	✓	
<b>MQIACH_HB_INTERVAL</b> Heartbeat interval (seconds)	✓	✓	✓	✓	✓	✓	✓	✓	
<b>MQIACH_HDR_COMPRESSION</b> List of header data compression techniques supported by the channel	✓	✓	✓	✓	✓	✓	✓	✓	
<b>MQIACH_KEEP_ALIVE_INTERVAL</b> KeepAlive interval	✓	✓	✓	✓	✓	✓	✓	✓	
Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQIACH_LONG_RETRY</b> Long retry count	✓	✓					✓	✓	
<b>MQIACH_LONG_TIMER</b> Long timer	✓	✓					✓	✓	
> <b>MQIACH_MAX_INSTANCES</b> Maximum number of simultaneous instances of a server-connection channel that can be started.						✓			✓
> <b>MQIACH_MAX_INSTS_PER_CLIENT</b> Maximum number of simultaneous instances of a server-connection channel that can be started from a single client.						✓			

Table 204. Optional parameters for ChannelAttrs (continued)

Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQIACH_MAX_MSG_LENGTH</b> Maximum message length	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>MQIACH_MCA_TYPE</b> MCA type	✓	✓		✓			✓	✓	
<b>MQIACH_MR_COUNT</b> Message retry count			✓	✓				✓	
<b>MQIACH_MSG_COMPRESSION</b> List of message data compression techniques supported by the channel	✓	✓	✓	✓	✓	✓	✓	✓	
<b>MQIACH_MR_INTERVAL</b> Message retry interval (milliseconds)			✓	✓				✓	
<b>MQIACH_NPM_SPEED</b> Speed of nonpersistent messages	✓	✓	✓	✓			✓	✓	
Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP
<b>MQIACH_PORT</b> AMQP port number									✓
<b>MQIACH_PUT_AUTHORITY</b> Put authority			✓	✓		✓		✓	
<b>&gt;MQIACH_RESET_REQUESTED</b> Sequence number of outstanding request when a RESET CHANNEL command is used	✓	✓	✓	✓			✓	✓	
<b>MQIACH_SEQUENCE_NUMBER_WRAP</b> Sequence number wrap	✓	✓	✓	✓			✓	✓	
<b>MQIACH_SHARING_CONVERSATIONS</b> Value of Sharing Conversations						✓			
<b>MQIACH_SHORT_RETRY</b> Short retry count	✓	✓					✓	✓	
<b>MQIACH_SHORT_TIMER</b> Short timer	✓	✓					✓	✓	
 <b>MQIACH_SPL_PROTECTION</b> Security policy protection	✓	✓	✓	✓					
<b>MQIACH_SSL_CLIENT_AUTH</b> TLS client authentication	✓	✓	✓	✓		✓		✓	✓
<b>MQIACH_USE_CLIENT_ID</b> Specify that the client ID is used for authorization checks for an AMQP channel									✓
<b>MQIACH_XMIT_PROTOCOL_TYPE</b> Transport (transmission protocol) type	✓	✓	✓	✓	✓	✓	✓	✓	
Parameter	SDR	SVR	RCV	REQ	CLI CONN	SVR CONN	CLUS SDR	CLUS RCV	AMQP

Key:

- SDR - Sender
- SVR - Server
- RCV - Receiver
- REQ - Requester
- CLI CONN - Client Connection
- SVR CONN - Server Connection
- CLUS SDR - Cluster Sender
- CLUS RCV - Cluster receiver
- AMQP - AMQP

**Note:**

1. Only one of the following parameters can be specified:

- MQCACH\_JAAS\_CONFIG
- MQCACH\_MCA\_USER\_ID
- MQIACH\_USE\_CLIENT\_ID

If none of these parameters is specified, no authentication is performed. If MQCACH\_JAAS\_CONFIG is specified, the client flows a user name and password, in all other cases the flowed user name is ignored.

**ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

If this parameter is present, eligible channels are limited to the specified type. Any attribute selector specified in the *ChannelAttrs* list which is only valid for channels of a different type or types is ignored; no error is raised.

If this parameter is not present (or if MQCHT\_ALL is specified), channels of all types other than MQCHT\_MQTT are eligible. Each attribute specified must be a valid channel attribute selector (that is, it must be one from the following list), but it might not be applicable to all (or any) of the channels returned. Channel attribute selectors that are valid but not applicable to the channel are ignored, no error messages occur, and no attribute is returned.

The value can be:

**MQCHT\_SENDER**

Sender.

**MQCHT\_SERVER**

Server.

**MQCHT\_RECEIVER**

Receiver.

**MQCHT\_REQUESTER**

Requester.

**MQCHT\_SVRCONN**

Server-connection (for use by clients).

**MQCHT\_CLNTCONN**

Client connection.

**MQCHT\_CLUSRCVR**

Cluster-receiver.

**MQCHT\_CLUSSDR**

Cluster-sender.

**MQCHT\_AMQP**

AMQP channel.

**MQCHT\_MQTT**

Telemetry channel.

**MQCHT\_ALL**

All types other than MQCHT\_MQTT.

The default value if this parameter is not specified is MQCHT\_ALL.

**Note:** If this parameter is present, it must occur immediately after the **ChannelName** parameter on platforms other than z/OS otherwise resulting in a MQRCCF\_MSG\_LENGTH\_ERROR error message.

**CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

**IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ChannelAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter for channel type, you cannot also specify the **ChannelType** parameter.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

**QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

**MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

**MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_PRIVATE**

The object is defined as either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ChannelAttrs* except MQCACH\_CHANNEL\_NAME and MQCACH\_MCA\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CHANNEL\_NAME\_ERROR**

Channel name error.

**MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

**MQRCCF\_CHANNEL\_TYPE\_ERROR**

Channel type not valid.


**MQCMD\_INQUIRE\_CHANNEL (Inquire Channel)**
**MQTT on AIX, Linux, and Windows**

The Inquire Channel (MQCMD\_INQUIRE\_CHANNEL) PCF command inquires about the attributes of IBM MQ channel definitions.

**Required parameters****ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all channels having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

If this parameter is present, eligible channels are limited to the specified type. Any attribute selector specified in the *ChannelAttrs* list which is only valid for channels of a different type or types is ignored; no error is raised.

If this parameter is not present (or if MQCHT\_ALL is specified), channels of all types are eligible. Each attribute specified must be a valid channel attribute selector (that is, it must be one from the following list), but it might not be applicable to all (or any) of the channels returned. Channel attribute selectors that are valid but not applicable to the channel are ignored, no error messages occur, and no attribute is returned.

The value must be:

**MQCHT\_MQTT**

Telemetry channel.

## Optional parameters

### ChannelAttrs (MQCFIL)

Channel attributes (parameter identifier: MQIACF\_CHANNEL\_ATTRS).

The attribute list can specify the following value on its own - default value used if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

or a combination of the following parameters:

**MQCA\_SSL\_KEY\_REPOSITORY**

TLS Key Repository

**MQCACH\_CHANNEL\_NAME**

Channel name. You cannot use this attribute as a filter keyword.

**MQCACH\_JAAS\_CONFIG**

The file path of the JAAS configuration

**MQCACH\_LOCAL\_ADDRESS**

Local communications address for the channel

**MQCACH\_MCA\_USER\_ID**

MCA user identifier.

**MQCACH\_SSL\_CIPHER\_SPEC**

TLS cipher spec.

**MQCACH\_SSL\_KEY\_PASSPHRASE**

TLS key passphrase.

**MQIACH\_BACKLOG**

The number of concurrent connection requests that the channel supports.

**MQIACH\_CHANNEL\_TYPE**

Channel type

**MQIACH\_PORT**

Port number to use when *TransportType* is set to TCP.

**MQIACH\_PROTOCOL**

The communication protocol supported by the channel.

**MQIACH\_SSL\_CLIENT\_AUTH**

TLS client authentication.

**MQIACH\_USE\_CLIENT\_ID**

Specify whether to use the *clientID* of a new connection as the *userID* for that connection

**MQIACH\_XMIT\_PROTOCOL\_TYPE**

Transport (transmission protocol) type

**Note:**

1. Only one of the following parameters can be specified:

- MQCACH\_JAAS\_CONFIG

- MQCACH\_MCA\_USER\_ID
- MQIACH\_USE\_CLIENT\_ID

If none of these parameters are specified, no authentication is performed. If MQCACH\_JAAS\_CONFIG is specified, the client flows a user name and password, in all other cases the flowed user name is ignored.

## Error codes

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRCCF\_CHANNEL\_NAME\_ERROR**

Channel name error.

#### **MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

#### **MQRCCF\_CHANNEL\_TYPE\_ERROR**

Channel type not valid.

## **MQCMD\_INQUIRE\_CHANNEL (Inquire Channel) Response**


The response to the Inquire Channel (MQCMD\_INQUIRE\_CHANNEL) PCF command consists of the response header followed by the *ChannelName* and *ChannelType* structures (and on z/OS only, the *DefaultChannelDisposition*, and *QSGDisposition* structure), and the requested combination of attribute parameter structures (where applicable).

If a generic channel name was specified, one such message is generated for each channel found.

### Always returned:

*ChannelName*, *ChannelType*,  *DefaultChannelDisposition*,  *QSGDisposition*

### Returned if requested:

*AlterationDate*, *AlterationTime*, *BatchDataLimit*, *BatchHeartbeat*, *BatchInterval*, *BatchSize*, *CertificateLabel*, *ChannelDesc*, *ChannelMonitoring*, *ChannelStatistics*, *ClientChannelWeight*, *ClientIdentifier*, *ClusterName*, *ClusterNameList*, *CLWLChannelPriority*, *CLWLChannelRank*, *CLWLChannelWeight*, *ConnectionAffinity*, *ConnectionName*, *DataConversion*, *DefReconnect*, *DiscInterval*, *HeaderCompression*, *HeartbeatInterval*, *InDoubtInbound*, *InDoubtOutbound*, *KeepAliveInterval*, *LastMsgTime*, *LocalAddress*, *LongRetryCount*, *LongRetryInterval*, *MaxMsgLength*, *MCAName*, *MCAType*, *MCAUserIdentifier*, *MessageCompression*, *ModeName*, *MsgExit*, *MsgRetryCount*, *MsgRetryExit*, *MsgRetryInterval*, *MsgRetryUserData*, *MsgsReceived*, *MsgsSent*, *MsgUserData*, *NetworkPriority*, *NonPersistentMsgSpeed*, *Password*, *PendingOutbound*, *PropertyControl*, *PutAuthority*, *QMgrName*, *ReceiveExit*, *ReceiveUserData*, *ResetSeq*, *SecurityExit*, *SecurityUserData*, *SendExit*, *SendUserData*, *SeqNumberWrap*, *SharingConversations*, *ShortRetryCount*, *ShortRetryInterval*,  *SPLProtection*, *SSLCipherSpec*, *SSLCipherSuite*, *SSLClientAuth*, *SSLPeerName*, *TpName*, *TransportType*, *UseDLQ*, *UserIdentifier*, *XmitQName*

## Response data

### **AlterationDate (MQCFST)**

Alteration date, in the form yyyy-mm-dd (parameter identifier: MQCA\_ALTERATION\_DATE).

The date when the information was last altered.

### **AlterationTime (MQCFST)**

Alteration time, in the form hh.mm.ss (parameter identifier: MQCA\_ALTERATION\_TIME).

The time when the information was last altered.



**BatchDataLimit (MQCFIN)**

Batch data limit (parameter identifier: MQIACH\_BATCH\_DATA\_LIMIT).

The limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point. A sync point is taken after the message that caused the limit to be reached has flowed across the channel. A value of zero in this attribute means that no data limit is applied to batches over this channel.

This parameter only applies to channels with a *ChannelType* of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_CLUSRCVR, or MQCHT\_CLUSSDR.

**BatchHeartbeat (MQCFIN)**

The value being used for the batch heartbeating (parameter identifier: MQIACH\_BATCH\_HB).

The value can be 0 - 999999. A value of 0 indicates that heartbeating is not in use.

**BatchInterval (MQCFIN)**

Batch interval (parameter identifier: MQIACH\_BATCH\_INTERVAL).

**BatchSize (MQCFIN)**

Batch size (parameter identifier: MQIACH\_BATCH\_SIZE).

**CertificateLabel (MQCFST)**

Certificate label (parameter identifier: MQCA\_CERT\_LABEL).

Specifies the certificate label in use.

The maximum length is MQ\_CERT\_LABEL\_LENGTH.

**ChannelDesc (MQCFST)**

Channel description (parameter identifier: MQCACH\_DESC).

The maximum length of the string is MQ\_CHANNEL\_DESC\_LENGTH.

**ChannelMonitoring (MQCFIN)**

Online monitoring data collection (parameter identifier: MQIA\_MONITORING\_CHANNEL).

The value can be any of the following values:

**MQMON\_OFF**

Online monitoring data collection is turned off for this channel.

**MQMON\_Q\_MGR**

The value of the queue manager's **ChannelMonitoring** parameter is inherited by the channel.

**MQMON\_LOW**

Online monitoring data collection is turned on, with a low rate of data collection, for this channel unless the queue manager's **ChannelMonitoring** parameter is MQMON\_NONE.

**MQMON\_MEDIUM**

Online monitoring data collection is turned on, with a moderate rate of data collection, for this channel unless the queue manager's *ChannelMonitoring* parameter is MQMON\_NONE.

**MQMON\_HIGH**

Online monitoring data collection is turned on, with a high rate of data collection, for this channel unless the queue manager's **ChannelMonitoring** parameter is MQMON\_NONE.

**ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**ChannelStatistics (MQCFIN)**

Statistics data collection (parameter identifier: MQIA\_STATISTICS\_CHANNEL).

The value can be any of the following values:

**MQMON\_OFF**

Statistics data collection is turned off for this channel.

**MQMON\_Q\_MGR**

The value of the queue manager's **ChannelStatistics** parameter is inherited by the channel.

**MQMON\_LOW**


Statistics data collection is turned on, with a low rate of data collection, for this channel unless the queue manager's **ChannelStatistics** parameter is MQMON\_NONE.

**MQMON\_MEDIUM**

Statistics data collection is turned on, with a moderate rate of data collection, for this channel unless the queue manager's **ChannelStatistics** parameter is MQMON\_NONE.

**MQMON\_HIGH**

Statistics data collection is turned on, with a high rate of data collection, for this channel unless the queue manager's **ChannelStatistics** parameter is MQMON\_NONE.

 On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

**ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

The value can be any of the following values:

**MQCHT\_SENDER**

Sender.

**MQCHT\_SERVER**

Server.

**MQCHT\_RECEIVER**

Receiver.

**MQCHT\_REQUESTER**

Requester.

**MQCHT\_SVRCONN**

Server-connection (for use by clients).

**MQCHT\_CLNTCONN**

Client connection.

**MQCHT\_CLUSRCVR**

Cluster-receiver.

**MQCHT\_CLUSSDR**

Cluster-sender.

**MQCHT\_MQTT**

Telemetry channel.

**ClientChannelWeight (MQCFIN)**

Client Channel Weight (parameter identifier: MQIACH\_CLIENT\_CHANNEL\_WEIGHT).

The client channel weighting attribute is used so client channel definitions can be selected at random, with the larger weightings having a higher probability of selection, when more than one suitable definition is available.

The value can be 0 - 99. The default is 0.

This parameter is only valid for channels with a ChannelType of MQCHT\_CLNTCONN

**ClientIdentifier (MQCFST)**

the clientId of the client (parameter identifier: MQCACH\_CLIENT\_ID).

**ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

**ClusterNamelist (MQCFST)**

Cluster namelist (parameter identifier: MQCA\_CLUSTER\_NAMELIST).

**CLWLChannelPriority (MQCFIN)**

Channel priority (parameter identifier: MQIACH\_CLWL\_CHANNEL\_PRIORITY).

**CLWLChannelRank (MQCFIN)**

Channel rank (parameter identifier: MQIACH\_CLWL\_CHANNEL\_RANK).

**CLWLChannelWeight (MQCFIN)**

Channel weighting (parameter identifier: MQIACH\_CLWL\_CHANNEL\_WEIGHT).

**ConnectionAffinity (MQCFIN)**

Channel Affinity (parameter identifier: MQIACH\_CONNECTION\_AFFINITY)

The channel affinity attribute specifies whether client applications that connect multiple times using the same queue manager name, use the same client channel. The value can be any of the following values:

**MQCAFTY\_PREFERRED**

The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions based on the weighting with any zero ClientChannelWeight definitions first in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful nonzero ClientChannelWeight definitions are moved to the end of the list. Zero ClientChannelWeight definitions remain at the start of the list and are selected first for each connection. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT has been modified since the list was created. Each client process with the same host name creates the same list.

MQCAFTY\_PREFERRED is the default, and has the value of 1.

**MQCAFTY\_NONE**

The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process independently select an applicable definition based on the weighting with any applicable zero ClientChannelWeight definitions selected first in alphabetical order. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT has been modified since the list was created.

This parameter is only valid for channels with a ChannelType of MQCHT\_CLNTCONN.

**ConnectionName (MQCFST)**

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH. On z/OS, it is MQ\_LOCAL\_ADDRESS\_LENGTH.

The *ConnectionName* is a comma-separated list.

**DataConversion (MQCFIN)**

Whether sender must convert application data (parameter identifier: MQIACH\_DATA\_CONVERSION).

The value can be:

**MQCDC\_NO\_SENDER\_CONVERSION**

No conversion by sender.

**MQCDC\_SENDER\_CONVERSION**

Conversion by sender.

**z/OS DefaultChannelDisposition (MQCFIN)**

Default channel disposition (parameter identifier: MQIACH\_DEF\_CHANNEL\_DISP).

This parameter applies to z/OS only.

Specifies the intended disposition of the channel when active. The value can be any of the following values:

**MQCHLD\_PRIVATE**

The intended use of the object is as a private channel.

**MQCHLD\_FIXSHARED**

The intended use of the object is as a shared channel linked to a specific queue manager.

**MQCHLD\_SHARED**

The intended use of the object is as a shared channel.

**DiscInterval (MQCFIN)**

Disconnection interval (parameter identifier: MQIACH\_DISC\_INTERVAL).

**DefReconnect (MQCFIN)**

Client channel default reconnection option (parameter identifier: MQIACH\_DEF\_RECONNECT).

The returned values can be:

**MQRCN\_NO**

MQRCN\_NO is the default value.

Unless overridden by **MQCONN**, the client is not reconnected automatically.

**MQRCN\_YES**

Unless overridden by **MQCONN**, the client reconnects automatically.

**MQRCN\_Q\_MGR**

Unless overridden by **MQCONN**, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as MQCNO\_RECONNECT\_Q\_MGR.

**MQRCN\_DISABLED**

Reconnection is disabled, even if requested by the client program using the **MQCONN** MQI call.

**HeaderCompression (MQCFIL)**

Header data compression techniques supported by the channel (parameter identifier: MQIACH\_HDR\_COMPRESSION). For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference.

The value can be one, or more, of

**MQCOMPRESS\_NONE**

No header data compression is performed.

**MQCOMPRESS\_SYSTEM**

Header data compression is performed.

**HeartbeatInterval (MQCFIN)**

Heartbeat interval (parameter identifier: MQIACH\_HB\_INTERVAL).

**InDoubtInbound (MQCFIN)**

Number of inbound messages to the client that are in doubt (Parameter identifier: MQIACH\_IN\_DOUBT\_IN).

**InDoubtOutbound (MQCFIN)**

Number of outbound messages from the client that are in doubt (Parameter identifier: MQIACH\_IN\_DOUBT\_OUT).

**KeepAliveInterval (MQCFIN)**

KeepAlive interval (parameter identifier: MQIACH\_KEEP\_ALIVE\_INTERVAL).

**LastMsgTime (MQCFST)**

The time that the last message was sent or received (parameter identifier: MQCACH\_LAST\_MSG\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**LocalAddress (MQCFST)**

Local communications address for the channel (parameter identifier: MQCACH\_LOCAL\_ADDRESS).

The maximum length of the string is MQ\_LOCAL\_ADDRESS\_LENGTH.

**LongRetryCount (MQCFIN)**

Long retry count (parameter identifier: MQIACH\_LONG\_RETRY).

**LongRetryInterval (MQCFIN)**

Long timer (parameter identifier: MQIACH\_LONG\_TIMER).

**MaxInstances (MQCFIN)**

Maximum number of simultaneous instances of a server-connection channel (parameter identifier: MQIACH\_MAX\_INSTANCES).

This parameter is returned only for server-connection channels in response to an Inquire Channel call with ChannelAttrs including MQIACF\_ALL or MQIACH\_MAX\_INSTANCES.

**MaxInstancesPerClient (MQCFIN)**

Maximum number of simultaneous instances of a server-connection channel that can be started from a single client (parameter identifier: MQIACH\_MAX\_INSTS\_PER\_CLIENT).

This parameter is returned only for server-connection channels in response to an Inquire Channel call with ChannelAttrs including MQIACF\_ALL or MQIACH\_MAX\_INSTS\_PER\_CLIENT.

**MaxMsgLength (MQCFIN)**

Maximum message length (parameter identifier: MQIACH\_MAX\_MSG\_LENGTH).

**MCAName (MQCFST)**

Message channel agent name (parameter identifier: MQCACH\_MCA\_NAME).

The maximum length of the string is MQ\_MCA\_NAME\_LENGTH.

**MCAType (MQCFIN)**

Message channel agent type (parameter identifier: MQIACH\_MCA\_TYPE).

The value can be any of the following values:

**MQMCAT\_PROCESS**

Process.

**MQMCAT\_THREAD**

Thread (Windows only).

**MCAUserIdentifier (MQCFST)**

Message channel agent user identifier (parameter identifier: MQCACH\_MCA\_USER\_ID).

**Note:** An alternative way of providing a user ID for a channel to run under is to use channel authentication records. With channel authentication records, different connections can use the same channel while using different credentials. If both MCAUSER on the channel is set and channel authentication records are used to apply to the same channel, the channel authentication records take precedence. The MCAUSER on the channel definition is only used if the channel authentication record uses USERSRC(CHANNEL). For more details, see [Channel authentication records](#)

The maximum length of the MCA user identifier depends on the environment in which the MCA is running. MQ\_MCA\_USER\_ID\_LENGTH gives the maximum length for the environment for which your application is running. MQ\_MAX\_MCA\_USER\_ID\_LENGTH gives the maximum for all supported environments.

On Windows, the user identifier might be qualified with the domain name in the following format:

user@domain

**MessageCompression (MQCFIL)**

Message data compression techniques supported by the channel (parameter identifier: MQIACH\_MSG\_COMPRESSION). For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference.

The value can be one, or more, of:

**MQCOMPRESS\_NONE**

No message data compression is performed.

**MQCOMPRESS\_RLE**

Message data compression is performed using run-length encoding.

**MQCOMPRESS\_ZLIBFAST**

Message data compression is performed using ZLIB encoding with speed prioritized.

**MQCOMPRESS\_ZLIBHIGH**

Message data compression is performed using ZLIB encoding with compression prioritized.

**MQCOMPRESS\_ANY**

Any compression technique supported by the queue manager can be used. MQCOMPRESS\_ANY is only valid for receiver, requester, and server-connection channels.

**ModeName (MQCFST)**


Mode name (parameter identifier: MQCACH\_MODE\_NAME).


The maximum length of the string is MQ\_MODE\_NAME\_LENGTH.

**MsgExit (MQCFST)**

Message exit name (parameter identifier: MQCACH\_MSG\_EXIT\_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

 On Multiplatforms, if more than one message exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure.

 On z/OS, an MQCFSL structure is always used.

**MsgsReceived (MQCFIN64)**

The number of messages received by the client since it last connected (parameter identifier: MQIACH\_MSGS\_RECEIVED / MQIACH\_MSGS\_RCVD).

**MsgRetryCount (MQCFIN)**

Message retry count (parameter identifier: MQIACH\_MR\_COUNT).

**MsgRetryExit (MQCFST)**

Message retry exit name (parameter identifier: MQCACH\_MR\_EXIT\_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

**MsgRetryInterval (MQCFIN)**

Message retry interval (parameter identifier: MQIACH\_MR\_INTERVAL).

**MsgRetryUserData (MQCFST)**

Message retry exit user data (parameter identifier: MQCACH\_MR\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.


**MsgsSent (MQCFIN64)**

The number of messages sent by the client since it last connected (parameter identifier: MQIACH\_MSGS\_SENT).

**MsgUserData (MQCFST)**

Message exit user data (parameter identifier: MQCACH\_MSG\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

 On Multiplatforms, if more than one message exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure.

 On z/OS, an MQCFSL structure is always used.

**NetworkPriority (MQCFIN)**

Network priority (parameter identifier: MQIACH\_NETWORK\_PRIORITY).

**NonPersistentMsgSpeed (MQCFIN)**

Speed at which non-persistent messages are to be sent (parameter identifier: MQIACH\_NPM\_SPEED).

The value can be:

**MQNPMS\_NORMAL**

Normal speed.

**MQNPMS\_FAST**

Fast speed.

**Password (MQCFST)**

Password (parameter identifier: MQCACH\_PASSWORD).

If a nonblank password is defined, it is returned as asterisks. Otherwise, it is returned as blanks.

The maximum length of the string is MQ\_PASSWORD\_LENGTH. However, only the first 10 characters are used.

**PropertyControl (MQCFIN)**

Property control attribute (parameter identifier MQIA\_PROPERTY\_CONTROL).

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor). The value can be any of the following values:

**MQPROP\_COMPATIBILITY**

<i>Table 205. Range of results, depending on which message properties are set, when PropertyControl value is MQPROP_COMPATIBILITY</i>	
<b>Message properties</b>	<b>Result</b>
The message contains a property with a prefix of <b>mcd.</b> , <b>jms.</b> , <b>usr.</b> or <b>mqext.</b>	All optional message properties (where the <b>Support</b> value is MQPD_SUPPORT_OPTIONAL), except those properties in the message descriptor or extension, are placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.
The message does not contain a property with a prefix of <b>mcd.</b> , <b>jms.</b> , <b>usr.</b> or <b>mqext.</b>	All message properties, except those properties in the message descriptor or extension, are removed from the message before the message is sent to the remote queue manager.
The message contains a property where the <b>Support</b> field of the property descriptor is not set to MQPD_SUPPORT_OPTIONAL	The message is rejected with reason MQRC_UNSUPPORTED_PROPERTY and treated in accordance with its report options.
The message contains one or more properties where the <b>Support</b> field of the property descriptor is set to MQPD_SUPPORT_OPTIONAL but other fields of the property descriptor are set to non-default values	The properties with non-default values are removed from the message before the message is sent to the remote queue manager.
The MQRFH2 folder that would contain the message property needs to be assigned with the <i>content='properties'</i> attribute	The properties are removed to prevent MQRFH2 headers with unsupported syntax flowing to a V6 or prior queue manager.

**MQPROP\_NONE**

All properties of the message, except those properties in the message descriptor or extension, are removed from the message before the message is sent to the remote queue manager.

If the message contains a property where the **Support** field of the property descriptor is not set to MQPD\_SUPPORT\_OPTIONAL then the message is rejected with reason MQRC\_UNSUPPORTED\_PROPERTY and treated in accordance with its report options.

**MQPROP\_ALL**

All properties of the message are included with the message when it is sent to the remote queue manager. The properties, except those properties in the message descriptor (or extension), are placed in one or more MQRFH2 headers in the message data.

This attribute is applicable to Sender, Server, Cluster Sender, and Cluster Receiver channels.

**PutAuthority (MQCFIN)**

Put authority (parameter identifier: MQIACH\_PUT\_AUTHORITY).

The value can be any of the following values:

**MQPA\_DEFAULT**

Default user identifier is used.

**MQPA\_CONTEXT**

Context user identifier is used.

**QMgrName (MQCFST)**

Queue manager name (parameter identifier: MQCA\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**z/OS QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid only on z/OS. The value can be any of the following values:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**ReceiveExit (MQCFST)**

Receive exit name (parameter identifier: MQCACH\_RCV\_EXIT\_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

**Multi** On Multiplatforms, if more than one receive exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

**ReceiveUserData (MQCFST)**

Receive exit user data (parameter identifier: MQCACH\_RCV\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

**Multi** On Multiplatforms, if more than one receive exit user data string has been defined for the channel, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

**ResetSeq (MQCFIN)**

Pending reset sequence number (parameter identifier: MQIACH\_RESET\_REQUESTED).

This is the sequence number from an outstanding request and it indicates a user Reset Channel command request is outstanding.



A value of zero indicates that there is no outstanding Reset Channel. The value can be in the range 1 - 999999999.

Possible return values include MQCHRR\_RESET\_NOT\_REQUESTED.

This parameter is not applicable on z/OS.

### **SecurityExit (MQCFST)**

Security exit name (parameter identifier: MQCACH\_SEC\_EXIT\_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

### **SecurityUserData (MQCFST)**

Security exit user data (parameter identifier: MQCACH\_SEC\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

### **SendExit (MQCFST)**

Send exit name (parameter identifier: MQCACH\_SEND\_EXIT\_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

**Multi** On [Multiplatforms](#), if more than one send exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

### **SendUserData (MQCFST)**

Send exit user data (parameter identifier: MQCACH\_SEND\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

**Multi** On [Multiplatforms](#), if more than one send exit user data string has been defined for the channel, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

### **SeqNumberWrap (MQCFIN)**

Sequence wrap number (parameter identifier: MQIACH\_SEQUENCE\_NUMBER\_WRAP).

### **SharingConversations (MQCFIN)**

Number of sharing conversations (parameter identifier: MQIACH\_SHARING\_CONVERSATIONS).

This parameter is returned only for TCP/IP client-connection and server-connection channels.

### **ShortRetryCount (MQCFIN)**

Short retry count (parameter identifier: MQIACH\_SHORT\_RETRY).

### **ShortRetryInterval (MQCFIN)**

Short timer (parameter identifier: MQIACH\_SHORT\_TIMER).

### **z/OS SPLProtection (MQCFIN)**

SPLProtection (parameter identifier: MQIACH\_SPL\_PROTECTION). This parameter applies to z/OS only, from IBM MQ 9.1.3 onwards.

Security policy protection parameter. Specifies what happens to messages across the channel when Advanced Message Security is active and an applicable policy exists.

This parameter is valid for channel types MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_RECEIVER, and MQCHT\_REQUESTER only.

Possible values are:

**MQSPL\_PASSTHRU**

Pass through, unchanged, any messages sent or received by the message channel agent for this channel.

This value is valid only for *ChannelType* values of MQCHT\_SENDER, MQCHT\_SERVER, MQCHT\_RECEIVER, or MQCHT\_REQUESTER, and is the default value.

**MQSPL\_REMOVE**

Remove any AMS protection from messages retrieved from the transmission queue by the message channel agent, and send the messages to the partner.

When the MCA gets a message from the transmission queue, if an AMS policy is defined for the transmission queue, it is applied to remove any AMS protection from the message prior to sending the message across the channel. If an AMS policy is not defined for the transmission queue, the message is sent as is.

This value is valid only for *ChannelType* values of MQCHT\_SENDER or MQCHT\_SERVER.

**MQSPL\_AS\_POLICY**

Based on the policy defined for the target queue, apply AMS protection to inbound messages prior to putting them on to the target queue.

When the message channel agent receives an inbound message, if an AMS policy is defined for the target queue, AMS protection is applied to the message prior to the message being put to the target queue. If an AMS policy is not defined for the target queue, the message is put to the target queue as is.

This value is valid only for *ChannelType* values of MQCHT\_RECEIVER or MQCHT\_REQUESTER.

**SSLCipherSpec (MQCFST)**

CipherSpec (parameter identifier: MQCACH\_SSL\_CIPHER\_SPEC).

The length of the string is MQ\_SSL\_CIPHER\_SPEC\_LENGTH.

**SSLCipherSuite (MQCFST)**

CipherSuite (parameter identifier: MQCACH\_SSL\_CIPHER\_SUITE).

The length of the string is MQ\_SSL\_CIPHER\_SUITE\_LENGTH.

**SSLClientAuth (MQCFIN)**

Client authentication (parameter identifier: MQIACH\_SSL\_CLIENT\_AUTH).

The value can be

**MQSCA\_REQUIRED**

Client authentication required

**MQSCA\_OPTIONAL**

Client authentication is optional.

The following value is also valid for Channels of type MQCHT\_MQTT:

**MQSCA\_NEVER\_REQUIRED**

Client authentication is never required, and must not be provided.

Defines whether IBM MQ requires a certificate from the TLS client.

**SSLPeerName (MQCFST)**

Peer name (parameter identifier: MQCACH\_SSL\_PEER\_NAME).

**Note:** An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both SSLPEER on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns in order to connect. For more information, see [Channel authentication records](#).

The length of the string is MQ\_SSL\_PEER\_NAME\_LENGTH. On z/OS, it is MQ\_SSL\_SHORT\_PEER\_NAME\_LENGTH.

Specifies the filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel. (A Distinguished Name is the identifier of the TLS certificate.) If the Distinguished Name in the certificate received from the peer does not match the SSLPEER filter, the channel does not start.

#### **TPName (MQCFST)**

Transaction program name (parameter identifier: MQCACH\_TP\_NAME).

The maximum length of the string is MQ\_TP\_NAME\_LENGTH.

#### **TransportType (MQCFIN)**

Transmission protocol type (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

The value might be:

##### **MQXPT\_LU62**

LU 6.2.

##### **MQXPT\_TCP**

TCP.

##### **MQXPT\_NETBIOS**

NetBIOS.

##### **MQXPT\_SPX**

SPX.

##### **MQXPT\_DECNET**

DECnet.

#### **UseDLQ (MQCFIN)**

Whether the dead-letter queue (or undelivered message queue) should be used when messages cannot be delivered by channels (parameter identifier: MQIA\_USE\_DEAD\_LETTER\_Q).

The value might be:

##### **MQUSEDLQ\_NO**

Messages that cannot be delivered by a channel will be treated as a failure and either the channel will discard them, or the channel will end, in accordance with the setting of NPMSPEED.

##### **MQUSEDLQ\_YES**

If the queue manager DEADQ attribute provides the name of a dead-letter queue then it will be used, otherwise the behavior will be as for MQUSEDLQ\_NO.

#### **UserIdentifier (MQCFST)**

Task user identifier (parameter identifier: MQCACH\_USER\_ID).

The maximum length of the string is MQ\_USER\_ID\_LENGTH. However, only the first 10 characters are used.

#### **XmitQName (MQCFST)**

Transmission queue name (parameter identifier: MQCACH\_XMIT\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **MQCMD\_INQUIRE\_CHANNEL\_INIT (Inquire Channel Initiator) on z/OS**

The Inquire Channel Initiator (MQCMD\_INQUIRE\_CHANNEL\_INIT) PCF command returns information about the channel initiator.

## **Optional parameters**

#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## **MQCMD\_INQUIRE\_CHANNEL\_INIT (Inquire Channel Initiator) Response on z/OS**

The response to the Inquire Channel Initiator (MQCMD\_INQUIRE\_CHANNEL\_INIT) PCF command consists of one response with a series of attribute parameter structures showing the status of the channel initiator (shown by the *ChannelInitiatorStatus* parameter), and one response for each listener (shown by the **ListenerStatus** parameter).

### **Always returned (one message with channel initiator information):**

*ActiveChannels, ActiveChannelsMax, ActiveChannelsPaused, ActiveChannelsRetrying, ActiveChannelsStarted, ActiveChannelsStopped, AdaptersMax, AdaptersStarted, ChannelInitiatorStatus, CurrentChannels, CurrentChannelsLU62, CurrentChannelsMax, CurrentChannelsTCP, DispatchersMax, DispatchersStarted, SSLTasksStarted, TCPName*

### **Always returned (one message for each listener):**

*InboundDisposition, ListenerStatus, TransportType*

### **Returned if applicable for the listener:**

*IPAddress, LUName, Port*

## **Response data - channel initiator information**

### **ActiveChannels (MQCFIN)**

The number of active channel connections (parameter identifier: MQIACH\_ACTIVE\_CHL).

### **ActiveChannelsMax (MQCFIN)**

The requested number of active channel connections (parameter identifier: MQIACH\_ACTIVE\_CHL\_MAX).

### **ActiveChannelsPaused (MQCFIN)**

The number of active channel connections that have paused, waiting to become active, because the limit for active channels has been reached (parameter identifier: MQIACH\_ACTIVE\_CHL\_PAUSED).

### **ActiveChannelsRetrying (MQCFIN)**

The number of active channel connections that are attempting to reconnect following a temporary error (parameter identifier: MQIACH\_ACTIVE\_CHL\_RETRY).

### **ActiveChannelsStarted (MQCFIN)**

The number of active channel connections that have started (parameter identifier: MQIACH\_ACTIVE\_CHL\_STARTED).

### **ActiveChannelsStopped (MQCFIN)**

The number of active channel connections that have stopped, requiring manual intervention (parameter identifier: MQIACH\_ACTIVE\_CHL\_STOPPED).

### **AdaptersMax (MQCFIN)**

The requested number of adapter subtasks (parameter identifier: MQIACH\_ADAPS\_MAX).

### **AdaptersStarted (MQCFIN)**

The number of active adapter subtasks (parameter identifier: MQIACH\_ADAPS\_STARTED).

### **ChannelInitiatorStatus (MQCFIN)**

Status of the channel initiator (parameter identifier: MQIACF\_CHINIT\_STATUS).

The value can be:

**MQSVC\_STATUS\_STOPPED**

The channel initiator is not running.

**MQSVC\_STATUS\_RUNNING**

The channel initiator is fully initialized and is running.

**CurrentChannels (MQCFIN)**

The number of current channel connections (parameter identifier: MQIACH\_CURRENT\_CHL).

**CurrentChannelsLU62 (MQCFIN)**

The number of current LU 6.2 channel connections (parameter identifier: MQIACH\_CURRENT\_CHL\_LU62).

**CurrentChannelsMax (MQCFIN)**

The requested number of channel connections (parameter identifier: MQIACH\_CURRENT\_CHL\_MAX).

**CurrentChannelsTCP (MQCFIN)**

The number of current TCP/IP channel connections (parameter identifier: MQIACH\_CURRENT\_CHL\_TCP).

**DispatchersMax (MQCFIN)**

The requested number of dispatchers (parameter identifier: MQIACH\_DISPS\_MAX).

**DispatchersStarted (MQCFIN)**

The number of active dispatchers (parameter identifier: MQIACH\_DISPS\_STARTED).

**SSLTasksMax (MQCFIN)**

The requested number of TLS server subtasks (parameter identifier: MQIACH\_SSLTASKS\_MAX).

**SSLTasksStarted (MQCFIN)**

The number of active TLS server subtasks (parameter identifier: MQIACH\_SSLTASKS\_STARTED).

**TCPName (MQCFST)**

TCP system name (parameter identifier: MQCACH\_TCP\_NAME).

The maximum length is MQ\_TCP\_NAME\_LENGTH.

## Response data - listener information

**InboundDisposition (MQCFIN)**

Inbound transmission disposition (parameter identifier: MQIACH\_INBOUND\_DISP).

Specifies the disposition of the inbound transmissions that the listener handles. The value can be any of the following values:

**MQINBD\_Q\_MGR**

Handling for transmissions directed to the queue manager. MQINBD\_Q\_MGR is the default.

**MQINBD\_GROUP**

Handling for transmissions directed to the queue sharing group. MQINBD\_GROUP is permitted only if there is a shared queue manager environment.

**IPAddress (MQCFST)**

IP address on which the listener listens (parameter identifier: MQCACH\_IP\_ADDRESS).

**ListenerStatus (MQCFIN)**

Listener status (parameter identifier: MQIACH\_LISTENER\_STATUS).

The value can be:

**MQSVC\_STATUS\_RUNNING**

The listener has started.

**MQSVC\_STATUS\_STOPPED**

The listener has stopped.

**MQSVC\_STATUS\_RETRYING**

The listener is trying again.

**LUName (MQCFST)**

LU name on which the listener listens (parameter identifier: MQCACH\_LU\_NAME).

The maximum length is MQ\_LU\_NAME\_LENGTH.

**Port (MQCFIN)**

Port number on which the listener listens (parameter identifier: MQIACH\_PORT\_NUMBER).

**TransportType (MQCFIN)**

Transmission protocol type that the listener is using (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

The value can be:

**MQXPT\_LU62**

LU62.

**MQXPT\_TCP**

TCP.

**MQCMD\_INQUIRE\_CHANNEL\_NAMES (Inquire Channel Names)**

The Inquire Channel Names (MQCMD\_INQUIRE\_CHANNEL\_NAMES) PCF command inquires a list of IBM MQ channel names that match the generic channel name, and the optional channel type specified.

**Required parameters****ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**Optional parameters****ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

If present, this parameter limits the channel names returned to channels of the specified type.

The value can be any of the following values:

**MQCHT\_SENDER**

Sender.

**MQCHT\_SERVER**

Server.

**MQCHT\_RECEIVER**

Receiver.

**MQCHT\_REQUESTER**

Requester.

**MQCHT\_SVRCONN**

Server-connection (for use by clients).

**MQCHT\_CLNTCONN**

Client connection.

**MQCHT\_CLUSRCVR**

Cluster-receiver.

**MQCHT\_CLUSSDR**

Cluster-sender.

## **MQCHT\_ALL**

All types.

The default value if this parameter is not specified is MQCHT\_ALL, which means that channels of all types except MQCHT\_CLNTCONN are eligible.

z/OS

## **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

### **MQQSGD\_PRIVATE**

The object is defined with either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

## **Error code**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CHANNEL\_NAME\_ERROR**

Channel name error.

**MQRCCF\_CHANNEL\_TYPE\_ERROR**

Channel type not valid.

**MQCMD\_INQUIRE\_CHANNEL\_NAMES (Inquire Channel Names) Response**

The response to the Inquire Channel Names (MQCMD\_INQUIRE\_CHANNEL\_NAMES) PCF command consists of one response for each client connection channel (except for SYSTEM.DEF.CLNTCONN), and a final message with all the remaining channels.

**Always returned:**

*ChannelNames, ChannelTypes*

**Returned if requested:**

None

**z/OS** On z/OS only, one additional parameter structure (with the same number of entries as the *ChannelNames* structure), is returned. Each entry in the structure, *QSGDispositions*, indicates the disposition of the object with the corresponding entry in the *ChannelNames* structure.

**Response data****ChannelNames (MQCFSL)**

List of channel names (parameter identifier: MQCACH\_CHANNEL\_NAMES).

**ChannelTypes (MQCFIL)**

List of channel types (parameter identifier: MQIACH\_CHANNEL\_TYPES). Possible values for fields in this structure are those values permitted for the **ChannelType** parameter, except MQCHT\_ALL.

**z/OS****QSGDispositions (MQCFIL)**

List of queue sharing group dispositions (parameter identifier: MQIACF\_QSG\_DISPS). This parameter is valid only on z/OS. The value can be:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQCMD\_INQUIRE\_CHANNEL\_STATUS (Inquire Channel Status)**

The Inquire Channel Status (MQCMD\_INQUIRE\_CHANNEL\_STATUS) PCF command inquires about the status of one or more channel instances.

You must specify the name of the channel for which you want to inquire status information. This name can be a specific channel name or a generic channel name. By using a generic channel name, you can inquire either:

- Status information for all channels, or
- Status information for one or more channels that match the specified name.

You must also specify whether you want:

- The status data (of current channels only), or
- The saved status data of all channels, or



- On z/OS only, the short status data of the channel.

Status for all channels that meet the selection criteria is returned, whether the channels were defined manually or automatically.

## Selection

The way to make a selection, is to use one of the following options:

- **XmitQname** (MQCACH\_XMIT\_Q\_NAME)
- **ConnectionName** (MQCACH\_CONNECTION\_NAME)
- **z/OS ChannelType** (MQIACH\_CHANNEL\_TYPE)
- **ChannelInstanceType** (MQIACH\_CHANNEL\_INSTANCE\_TYPE)
- **ChannelSummaryAttrs** (MQIACH\_CHANNEL\_SUMMARY\_ATTRS)
- **ClientID** (MQCACH\_CLIENT\_ID)

**Multi** This command includes a check on the current depth of the transmission queue for the channel, if the channel is a CLUSSDR channel. To issue this command, you must be authorized to inquire the queue depth, and to do this requires *+inq* authority on the transmission queue. Note that another name for this authority is MQZAO\_INQUIRE.

**Multi** Without this authority this command runs without error, but a value of zero is output for the **MsgsAvailable** parameter of the “MQCMD\_INQUIRE\_CHANNEL\_STATUS (Inquire Channel Status) Response” on page 1231 command. If you have the correct authority, the command provides the correct value for **MsgsAvailable**.

There are three classes of data available for channel status. These classes are **saved**, **current**, and **short**. The status fields available for saved data are a subset of the fields available for current data and are called **common** status fields. Although the common data *fields* are the same, the data *values* might be different for saved and current status. The rest of the fields available for current data are called **current-only** status fields.

- **Saved** data consists of the common status fields. This data is reset at the following times:
  - For all channels:
    - When the channel enters or leaves STOPPED or RETRY state
  - For a sending channel:
    - Before requesting confirmation that a batch of messages has been received
    - When confirmation has been received
  - For a receiving channel:
    - Just before confirming that a batch of messages has been received
  - For a server connection channel:
    - No data is saved

Therefore, a channel which has never been current does not have any saved status.

- **Current** data consists of the common status fields and current-only status fields. The data fields are continually updated as messages are sent or received.
- **Short** data consists of the queue manager name that owns the channel instance. This class of data is available only on z/OS.

This method of operation has the following consequences:

- An inactive channel might not have any saved status if it has never been current or has not yet reached a point where saved status is reset.
- The "common" data fields might have different values for saved and current status.

- A current channel always has current status and might have saved status.

Channels can be current or inactive:

### Current channels

These are channels that have been started, or on which a client has connected, and that have not finished or disconnected normally. They might not yet have reached the point of transferring messages, or data, or even of establishing contact with the partner. Current channels have **current** status and can also have **saved** or **short** status.

The term **Active** is used to describe the set of current channels which are not stopped.

### Inactive channels

These are channels that have either not been started or on which a client has not connected, or that have finished or disconnected normally. (If a channel is stopped, it is not yet considered to have finished normally and is, therefore, still current.) Inactive channels have either **saved** status or no status at all.

There can be more than one instance of a receiver, requester, cluster-sender, cluster-receiver, or server-connection channel current at the same time (the requester is acting as a receiver). This situation occurs if several senders, at different queue managers, each initiate a session with this receiver, using the same channel name. For channels of other types, there can only be one instance current at any time.

For all channel types, however, there can be more than one set of saved status information available for a particular channel name. At most one of these sets relates to a current instance of the channel, the rest relate to previously current instances. Multiple instances arise if different transmission queue names or connection names have been used with the same channel. This situation can happen in the following cases:

- At a sender or server:
  - If the same channel has been connected to by different requesters (servers only),
  - If the transmission queue name has been changed in the definition, or
  - If the connection name has been changed in the definition.
- At a receiver or requester:
  - If the same channel has been connected to by different senders or servers, or
  - If the connection name has been changed in the definition (for requester channels initiating connection).

The number of sets returned for a particular channel can be limited by using the **XmitQName**, **ConnectionName** and **ChannelInstanceType** parameters.

## Required parameters

### ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The channel name is always returned, regardless of the instance attributes requested.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

## Optional parameters

### ChannelDisposition (MQCFIN)

Channel disposition (parameter identifier: MQIACH\_CHANNEL\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels for which information is to be returned. The value can be any of the following values:

**MQCHLD\_ALL**

Returns requested status information for private channels.

In a shared queue environment where the command is being executed on the queue manager where it was issued, or if *ChannelInstanceType* has a value of MQOT\_CURRENT\_CHANNEL, this option also displays the requested status information for shared channels.

**MQCHLD\_PRIVATE**

Returns requested status information for private channels.

**MQCHLD\_SHARED**

Returns requested status information for shared channels.

The status information that is returned for various combinations of *ChannelDisposition*, *CommandScope*, and status type, is summarized in [Table 206 on page 1219](#), [Table 207 on page 1219](#), and [Table 208 on page 1220](#).

*Table 206. ChannelDisposition and CommandScope for Inquire Channel Status, Current*

<b>ChannelDisposition</b>	<b>CommandScope blank or local queue manager</b>	<b>CommandScope (qmgr-name)</b>	<b>CommandScope (*)</b>
MQCHLD_PRIVATE	Common and current-only status for current private channels on the local queue manager	Common and current-only status for current private channels on the named queue manager	Common and current-only status for current private channels on all queue managers
MQCHLD_SHARED	Common and current-only status for current shared channels on the local queue manager	Common and current-only status for current shared channels on the named queue manager	Common and current-only status for current shared channels on all queue managers
MQCHLD_ALL	Common and current-only status for current private and shared channels on the local queue manager	Common and current-only status for current private and shared channels on the named queue manager	Common and current-only status for current private and shared channels on all active queue managers

*Table 207. ChannelDisposition and CommandScope for Inquire Channel Status, Short*

<b>ChannelDisposition</b>	<b>CommandScope blank or local queue manager</b>	<b>CommandScope (qmgr-name)</b>	<b>CommandScope (*)</b>
MQCHLD_PRIVATE	<i>ChannelStatus</i> and short status for current private channels on the local queue manager	<i>ChannelStatus</i> and short status for current private channels on the named queue manager	<i>ChannelStatus</i> and short status for current private channels on all active queue managers
MQCHLD_SHARED	<i>ChannelStatus</i> and short status for current shared channels on all active queue managers in the queue sharing group	Not permitted	Not permitted
MQCHLD_ALL	<i>ChannelStatus</i> and short status for current private channels on the local queue manager and current shared channels in the queue sharing group( “1” on page 1220 )	<i>ChannelStatus</i> and short status for current private channels on the named queue manager	<i>ChannelStatus</i> and short status for current private, and shared, channels on all active queue managers in the queue sharing group( “1” on page 1220 )

**Note:**

1. In this case you get two separate sets of responses to the command on the queue manager where it was entered; one for MQCHLD\_PRIVATE and one for MQCHLD\_SHARED.

<b>ChannelDisposition</b>	<b>CommandScope blank or local queue manager</b>	<b>CommandScope (qmgr-name)</b>	<b>CommandScope (*)</b>
MQCHLD_PRIVATE	Common status for saved private channels on the local queue manager	Common status for saved private channels on the named queue manager	Common status for saved private channels on all active queue managers
MQCHLD_SHARED	Common status for saved shared channels on all active queue managers in the queue sharing group	Not permitted	Not permitted
MQCHLD_ALL	Common status for saved private channels on the local queue manager and saved shared channels in the queue sharing group	Common status for saved private channels on the named queue manager	Common status for saved private, and shared, channels on all active queue managers in the queue sharing group

You cannot use this parameter as a filter keyword.

**ChannelInstanceAttrs (MQCFIL)**

Channel instance attributes (parameter identifier: MQIACH\_CHANNEL\_INSTANCE\_ATTRS).

The **ChannelInstanceAttrs** parameter names the list of attributes to be returned. This parameter does not provide any way to select, based upon the value of the items in that list of attributes.

If status information is requested which is not relevant for the particular channel type, it is not an error. Similarly, it is not an error to request status information that is applicable only to active channels for saved channel instances. In both of these cases, no structure is returned in the response for the information concerned.

For a saved channel instance, the MQCACH\_CURRENT\_LUWID, MQIACH\_CURRENT\_MSGS, and MQIACH\_CURRENT\_SEQ\_NUMBER attributes have meaningful information only if the channel instance is in doubt. However, the attribute values are still returned when requested, even if the channel instance is not in-doubt.

The attribute list might specify the following value on its own:

**MQIACF\_ALL**

All attributes.

MQIACF\_ALL is the default value used if the parameter is not specified or it can specify a combination of the following:

- Relevant for common status:

The following information applies to all sets of channel status, whether the set is current.

**MQCACH\_CHANNEL\_NAME**

Channel name.

**MQCACH\_CONNECTION\_NAME**

Connection name.

**MQCACH\_CURRENT\_LUWID**

Logical unit of work identifier for current batch.

**MQCACH\_LAST\_LUWID**

Logical unit of work identifier for last committed batch.

**MQCACH\_XMIT\_Q\_NAME**

Transmission queue name.

**MQIACH\_CHANNEL\_INSTANCE\_TYPE**

Channel instance type.

**MQIACH\_CHANNEL\_TYPE**

Channel type.

**MQIACH\_CURRENT\_MSGS**

Number of messages sent or received in current batch.

**MQIACH\_CURRENT\_SEQ\_NUMBER**

Sequence number of last message sent or received.

**MQIACH\_INDOUBT\_STATUS**

Whether the channel is currently in-doubt.

**MQIACH\_LAST\_SEQ\_NUMBER**

Sequence number of last message in last committed batch.

MQCACH\_CURRENT\_LUWID, MQCACH\_LAST\_LUWID, MQIACH\_CURRENT\_MSGS, MQIACH\_CURRENT\_SEQ\_NUMBER, MQIACH\_INDOUBT\_STATUS and MQIACH\_LAST\_SEQ\_NUMBER do not apply to server-connection channels, and no values are returned. If specified on the command, they are ignored.

- Relevant for current-only status:

The following information applies only to current channel instances. The information applies to all channel types, except where stated.

**MQCA\_Q\_MGR\_NAME**

Name of the queue manager that owns the channel instance. This parameter is valid only on z/OS.

**MQCA\_REMOTE\_Q\_MGR\_NAME**

Queue manager name, or queue sharing group name of the remote system. The remote queue manager name is always returned regardless of the instance attributes requested.

**MQCACH\_CHANNEL\_START\_DATE**

Date channel was started.

**MQCACH\_CHANNEL\_START\_TIME**

Time channel was started.

**MQCACH\_LAST\_MSG\_DATE**

Date last message was sent, or MQI call was handled.

**MQCACH\_LAST\_MSG\_TIME**

Time last message was sent, or MQI call was handled.

**MQCACH\_LOCAL\_ADDRESS**

Local communications address for the channel.

**MQCACH\_MCA\_JOB\_NAME**

Name of MCA job.

This parameter is not valid on z/OS.

You cannot use MQCACH\_MCA\_JOB\_NAME as a parameter to filter on.

**MQCACH\_MCA\_USER\_ID**

The user ID used by the MCA.

**MQCACH\_REMOTE\_APPL\_TAG**

Remote partner application name. MQCACH\_REMOTE\_APPL\_TAG is the name of the client application at the remote end of the channel. This parameter applies only to server-connection channels.

**MQCACH\_REMOTE\_PRODUCT**

Remote partner product identifier. This is the product identifier of the IBM MQ code running at the remote end of the channel.

**MQCACH\_REMOTE\_VERSION**

Remote partner version. This is the version of the IBM MQ code running at the remote end of the channel.

**MQCACH\_SSL\_CIPHER\_SPEC**

CipherSpec in use on the connection.

**MQCACH\_SSL\_SHORT\_PEER\_NAME**

TLS short peer name.

**MQCACH\_SSL\_CERT\_ISSUER\_NAME**

The full Distinguished Name of the issuer of the remote certificate.

**z/OS MQCACH\_SSL\_CERT\_USER\_ID**

User ID associated with the remote certificate; valid on z/OS only.

**MQCACH\_TOPIC\_ROOT**

Topic root for AMQP channel.

**MQIA\_MONITORING\_CHANNEL**

The level of monitoring data collection.

**z/OS MQIA\_STATISTICS\_CHANNEL**

The level of statistics data collection; valid on z/OS only.

**MQIACF\_MONITORING**

All channel status monitoring attributes. These attributes are:

**MQIA\_MONITORING\_CHANNEL**

The level of monitoring data collection.

**MQIACH\_BATCH\_SIZE\_INDICATOR**

Batch size.

**MQIACH\_COMPRESSION\_RATE**

The compression rate achieved displayed to the nearest percentage.

**MQIACH\_COMPRESSION\_TIME**

The amount of time per message, displayed in microseconds, spent during compression or decompression.

**MQIACH\_EXIT\_TIME\_INDICATOR**

Exit time.

**MQIACH\_NETWORK\_TIME\_INDICATOR**

Network time.

**MQIACH\_XMITQ\_MSGS\_AVAILABLE**

Number of messages available to the channel on the transmission queue.

**MQIACH\_XMITQ\_TIME\_INDICATOR**

Time on transmission queue.

You cannot use MQIACF\_MONITORING as a parameter to filter on.

**MQIACH\_BATCH\_SIZE\_INDICATOR**

Batch size.

You cannot use MQIACH\_BATCH\_SIZE\_INDICATOR as a parameter to filter on.

**MQIACH\_BATCHES**

Number of completed batches.

**MQIACH\_BUFFERS\_RCVD**

Number of buffers received.

**MQIACH\_BUFFERS\_SENT**

Number of buffers sent.

**MQIACH\_BYTES\_RCVD**

Number of bytes received.

**MQIACH\_BYTES\_SENT**

Number of bytes sent.

**MQIACH\_CHANNEL\_SUBSTATE**

The channel substate.

**MQIACH\_COMPRESSION\_RATE**

The compression rate achieved displayed to the nearest percentage.

You cannot use MQIACH\_COMPRESSION\_RATE as a parameter to filter on.

**MQIACH\_COMPRESSION\_TIME**

The amount of time per message, displayed in microseconds, spent during compression or decompression.

You cannot use MQIACH\_COMPRESSION\_TIME as a parameter to filter on.

**MQIACH\_CURRENT\_SHARING\_CONVS**

Requests information about the current number of conversations on this channel instance.

This attribute applies only to TCP/IP server-connection channels.

**MQIACH\_EXIT\_TIME\_INDICATOR**

Exit time.

You cannot use MQIACH\_EXIT\_TIME\_INDICATOR as a parameter to filter on.

**MQIACH\_HDR\_COMPRESSION**

Technique used to compress the header data sent by the channel.

**MQIACH\_KEEP\_ALIVE\_INTERVAL**

The KeepAlive interval in use for this session. This parameter is significant only for z/OS.

**MQIACH\_LONG\_RETRIES\_LEFT**

Number of long retry attempts remaining.

**MQIACH\_MAX\_MSG\_LENGTH**

Maximum message length. MQIACH\_MAX\_MSG\_LENGTH is valid only on z/OS.

**MQIACH\_MAX\_SHARING\_CONVS**

Requests information about the maximum number of conversations on this channel instance.

This attribute applies only to TCP/IP server-connection channels.

**MQIACH\_MCA\_STATUS**

MCA status.

You cannot use MQIACH\_MCA\_STATUS as a parameter to filter on.

**MQIACH\_MSG\_COMPRESSION**

Technique used to compress the message data sent by the channel.

**MQIACH\_MSGS**

Number of messages sent or received, or number of MQI calls handled.

**MQIACH\_NETWORK\_TIME\_INDICATOR**


Network time.

You cannot use MQIACH\_NETWORK\_TIME\_INDICATOR as a parameter on which to filter.

**MQIACH\_SECURITY\_PROTOCOL**

Security protocol currently in use.

This parameter does not apply to client-connection channels.

 From IBM MQ 9.1.1, this parameter is supported on z/OS.

**MQIACH\_SHORT\_RETRIES\_LEFT**

Number of short retry attempts remaining.

**MQIACH\_SSL\_KEY\_RESETS**

Number of successful TLS key resets.

**MQIACH\_SSL\_RESET\_DATE**

Date of previous successful TLS secret key reset.

**MQIACH\_SSL\_RESET\_TIME**

Time of previous successful TLS secret key reset.

**MQIACH\_STOP\_REQUESTED**

Whether user stop request has been received.

**MQIACH\_XMITQ\_MSGS\_AVAILABLE**

Number of messages available to the channel on the transmission queue.

**MQIACH\_XMITQ\_TIME\_INDICATOR**

Time on transmission queue.

You cannot use MQIACH\_XMITQ\_TIME\_INDICATOR as a parameter to filter on.

The following value is supported on all platforms:

**MQIACH\_BATCH\_SIZE**

Batch size.

The following value is supported on all platforms:

**MQIACH\_HB\_INTERVAL**

Heartbeat interval (seconds).

**MQIACH\_NPM\_SPEED**

Speed of nonpersistent messages.

The following attributes do not apply to server-connection channels, and no values are returned. If specified on the command they are ignored:

- MQIACH\_BATCH\_SIZE\_INDICATOR
- MQIACH\_BATCH\_SIZE
- MQIACH\_BATCHES
- MQIACH\_LONG\_RETRIES\_LEFT
- MQIACH\_NETWORK\_TIME
- MQIACH\_NPM\_SPEED
- MQCA\_REMOTE\_Q\_MGR\_NAME
- MQIACH\_SHORT\_RETRIES\_LEFT
- MQIACH\_XMITQ\_MSGS\_AVAILABLE
- MQIACH\_XMITQ\_TIME\_INDICATOR

The following attributes apply only to server-connection channels. If specified on the command for other types of channel the attribute is ignored and no value is returned:

- MQIACH\_CURRENT\_SHARING\_CONVS
- MQIACH\_MAX\_SHARING\_CONVS

-  Relevant for short status:

The following parameter applies to current channels on z/OS:

**MQCACH\_Q\_MGR\_NAME**

Name of the queue manager that owns the channel instance.

**ChannelInstanceType (MQCFIN)**

Channel instance type (parameter identifier: MQIACH\_CHANNEL\_INSTANCE\_TYPE).



It is always returned regardless of the channel instance attributes requested.

The value can be:

#### **MQOT\_CURRENT\_CHANNEL**

The channel status.

MQOT\_CURRENT\_CHANNEL is the default, and indicates that only current status information for active channels is to be returned.

Both common status information and active-only status information can be requested for current channels.

#### **MQOT\_SAVED\_CHANNEL**

Saved channel status.

Specify MQOT\_SAVED\_CHANNEL to cause saved status information for both active and inactive channels to be returned.

Only common status information can be returned. Active-only status information is not returned for active channels if this keyword is specified.

#### **z/OS MQOT\_SHORT\_CHANNEL**

Short channel status (valid on z/OS only).

Specify MQOT\_SHORT\_CHANNEL to cause short status information for current channels to be returned.

Other common status and current-only status information are not returned for current channels if this keyword is specified.

You cannot use MQIACH\_CHANNEL\_INSTANCE\_TYPE as a parameter to filter on.

#### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

#### **ConnectionName (MQCFST)**

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).

If this parameter is present, eligible channel instances are limited to those using this connection name. If it is not specified, eligible channel instances are not limited in this way.

The connection name is always returned, regardless of the instance attributes requested.

The value returned for *ConnectionName* might not be the same as in the channel definition, and might differ between the current channel status and the saved channel status. (Using *ConnectionName* for limiting the number of sets of status is therefore not recommended.)

For example, when using TCP, if *ConnectionName* in the channel definition:

- Is blank or is in *host name* format, the channel status value has the resolved IP address.
- Includes the port number, the current channel status value includes the port number (except on z/OS), but the saved channel status value does not.

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH.

#### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ChannelInstanceAttrs* except MQIACF\_ALL and others as noted. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter”](#) on page 1523 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

#### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ChannelInstanceAttrs* except MQCACH\_CHANNEL\_NAME and others as noted. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter”](#) on page 1530 for information about using this filter condition.

If you specify a string filter for **ConnectionName** or **XmitQName**, you cannot also specify the **ConnectionName** or **XmitQName** parameter.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

#### **XmitQName (MQCFST)**

Transmission queue name (parameter identifier: MQCACH\_XMIT\_Q\_NAME).

If this parameter is present, eligible channel instances are limited to those using this transmission queue. If it is not specified, eligible channel instances are not limited in this way.

The transmission queue name is always returned, regardless of the instance attributes requested.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **Error code**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

#### **Reason (MQLONG)**

The value can be any of the following values:

##### **MQRCCF\_CHANNEL\_NAME\_ERROR**

Channel name error.

##### **MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

##### **MQRCCF\_CHL\_INST\_TYPE\_ERROR**

Channel instance type not valid.

##### **MQRCCF\_CHL\_STATUS\_NOT\_FOUND**

Channel status not found.

##### **MQRCCF\_NONE\_FOUND**

Channel status not found.

##### **MQRCCF\_XMIT\_Q\_NAME\_ERROR**

Transmission queue name error.

## **MQCMD\_INQUIRE\_CHANNEL\_STATUS (inquire channel status) AMQP on AIX, Linux, and Windows**

The Inquire Channel Status (MQCMD\_INQUIRE\_CHANNEL\_STATUS) (AMQP) PCF command inquires about the status of one or more AMQP channel instances.

You must specify the name of the channel for which you want to inquire status information. This name can be a specific channel name or a generic channel name. By using a generic channel name, you can inquire either:

- Status information for all channels, or
- Status information for one or more channels that match the specified name.

If the **ClientIdentifier** parameter is not specified, the output of the **Inquire Channel Status** command is a summary of statuses of all clients connected to the channel. One PCF response message is returned per channel.

If the **ClientIdentifier** parameter is specified, separate PCF response messages are returned for each client connection. The **ClientIdentifier** parameter can be a wildcard, in which the status for all clients that match the **ClientIdentifier** string is returned.

### **Required parameters**

#### **ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects which have names that start with the selected character string. An asterisk on its own matches all possible names.

The channel name is always returned, regardless of the instance attributes requested.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

#### **ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

The value must be:

**MQCHT\_AMQP**  
AMQP

### **Optional parameters**

#### **ChannelInstanceAttrs (MQCFIL)**

Channel instance attributes (parameter identifier: MQIACH\_CHANNEL\_INSTANCE\_ATTRS).

The **ChannelInstanceAttrs** parameter names the list of attributes to be returned. This parameter does not provide any way to select, based upon the value of the items in that list of attributes.

The attribute list might specify the following value on its own:

**MQIACF\_ALL**  
All attributes.

MQIACF\_ALL is the default value used if the parameter is not specified or it can specify a combination of the following:

- Relevant for summary status, applicable when you do not specify a **ClientIdentifier** parameter.

The following information applies:

**MQCACH\_CHANNEL\_NAME**  
Channel name

**MQIACH\_CHANNEL\_TYPE**  
Channel type

**MQIACF\_CONNECTION\_COUNT**

Number of connections described in the summary

**MQIACH\_CHANNEL\_STATUS**

Current status of the client

- Relevant for client details mode, applicable when you do specify a **ClientIdentifier** parameter.

The following information applies:

**MQCACH\_CHANNEL\_NAME**

Channel name

**MQIACH\_CHANNEL\_STATUS**

Current status of the client

**MQIACH\_CHANNEL\_TYPE**

Channel type

**MQCACH\_CONNECTION\_NAME**

Name of the remote connection (IP address)

**MQIACH\_AMQP\_KEEP\_ALIVE**

Keep alive interval of the client

**MQCACH\_MCA\_USER\_ID**

Message channel agent user Id

**MQIACH\_MSGS\_SENT**

Number of messages sent by the client since it last connected

**MQIACH\_MSGS\_RECEIVED or MQIACH\_MSGS\_RCVD**

Number of messages received by the client since it last connected

**MQCACH\_LAST\_MSG\_DATE**

Date last message was received or sent

**MQCACH\_LAST\_MSG\_TIME**

Time last message was received or sent

**MQCACH\_CHANNEL\_START\_DATE**

Date channel started

**MQCACH\_CHANNEL\_START\_TIME**

Time channel started

**ClientIdentifier (MQCFST)**

The Client Id of the client (parameter identifier: MQCACH\_CLIENT\_ID).

The maximum length of the string is MQ\_CLIENT\_ID\_LENGTH.

**Summary mode**

If you do not specify a **ClientIdentifier** parameter, the following fields are returned:

**MQCACH\_CHANNEL\_NAME**

The channel name.

**MQIACH\_CHANNEL\_TYPE**

The channel type of AMQP.

**MQIACF\_CONNECTION\_COUNT**

The number of connections described in the summary.

**MQIACH\_CHANNEL\_STATUS**

The current status of the client.

**Client details mode**

If you specify a **ClientIdentifier** parameter, the following fields are returned:

**MQIACH\_CHANNEL\_STATUS**

The current status of the client.

**MQCACH\_CONNECTION\_NAME**

The name of the remote connection, that is, the IP address.

**MQIACH\_AMQP\_KEEP\_ALIVE**

The keep alive interval of the client.

**MQCACH\_MCA\_USER\_ID**

Message channel agent user Id.

**MQIACH\_MSGS\_SENT**

Number of messages sent by the client since it last connected.

**MQIACH\_MSGS\_RECEIVED or MQIACH\_MSGS\_RCVD**

Number of messages received by the client since it last connected.

**MQCACH\_LAST\_MSG\_DATE**

Date last message was received or sent.

**MQCACH\_LAST\_MSG\_TIME**

Time last message was received or sent.

**MQCACH\_CHANNEL\_START\_DATE**

Date channel started.

**MQCACH\_CHANNEL\_START\_TIME**

Time channel was started.

**MQIACH\_PROTOCOL**

AMQP protocol supported by this channel.

**Error code**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CHANNEL\_NAME\_ERROR**

Channel name error.

**MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

**MQRCCF\_CHL\_INST\_TYPE\_ERROR**

Channel instance type not valid.

**MQRCCF\_CHL\_STATUS\_NOT\_FOUND**

Channel status not found.

**MQRCCF\_XMIT\_Q\_NAME\_ERROR**

Transmission queue name error.

**ALW MQCMD\_INQUIRE\_CHANNEL\_STATUS (Inquire Channel Status) MQTT on AIX, Linux, and Windows**

The Inquire Channel Status (MQCMD\_INQUIRE\_CHANNEL\_STATUS) (MQTT) PCF command inquires about the status of one or more Telemetry channel instances.

You must specify the name of the channel for which you want to inquire status information. This name can be a specific channel name or a generic channel name. By using a generic channel name, you can inquire either:

- Status information for all channels, or
- Status information for one or more channels that match the specified name.

**Note:** The **Inquire Channel Status** command for MQ Telemetry has the potential to return a far larger number of responses than if the command was run for an IBM MQ channel. For this reason, the MQ Telemetry server does not return more responses than fit on the reply-to queue. The number of responses is limited to the value of `MAXDEPTH` parameter of the `SYSTEM.MQSC.REPLY.QUEUE` queue. When an MQ Telemetry command is truncated by the MQ Telemetry server, the `AMQ8492` message is displayed specifying how many responses are returned based on the size of `MAXDEPTH`.

If the **ClientIdentifier** parameter is not specified, the output of the **Inquire Channel Status** command is a summary of statuses of all clients connected to the channel. One PCF response message is returned per channel.

If the **ClientIdentifier** parameter is specified, separate PCF response messages are returned for each client connection. The **ClientIdentifier** parameter may be a wildcard, in which the status for all clients that match the **ClientIdentifier** string is returned (within the limits of **MaxResponses** and **ResponseRestartPoint** if they are set).

## Required parameters

### ChannelName (MQCFST)

Channel name (parameter identifier: `MQCACH_CHANNEL_NAME`).

Generic channel names are supported. A generic name is a character string followed by an asterisk (\*), for example `ABC*`, and it selects all objects which have names that start with the selected character string. An asterisk on its own matches all possible names.

This parameter is allowed for only when the **ResponseType** parameter is set to `MQRESP_TOTAL`.

The channel name is always returned, regardless of the instance attributes requested.

The maximum length of the string is `MQ_CHANNEL_NAME_LENGTH`.

### ChannelType (MQCFIN)

Channel type (parameter identifier: `MQIACH_CHANNEL_TYPE`).

The value must be:

**MQCHT\_MQTT**  
Telemetry.

## Optional parameters

### ClientIdentifier (MQCFST)

The ClientId of the client (parameter identifier: `MQCACH_CLIENT_ID`).

### MaxResponses (MQCFIN)

The maximum number of clients to return status for (parameter identifier: `MQIA_MAX_RESPONSES`).

This parameter is only allowed when the **ClientIdentifier** parameter is specified.

### ResponseRestartPoint (MQCFIN)

The first client to return status for (parameter identifier: `MQIA_RESPONSE_RESTART_POINT`). The combination of this parameter with **MaxResponses** enables the range of clients to be specified.

This parameter is only allowed when the **ClientIdentifier** parameter is specified.

## Client details mode

### STATUS

The current status of the client (parameter identifier: `MQIACH_CHANNEL_STATUS`).

### CONNAME

The name of the remote connection (ip address) (parameter identifier: `MQCACH_CONNECTION_NAME`).

### KAINTE

The keep alive interval of the client (parameter identifier: `MQIACH_KEEP_ALIVE_INTERVAL`).

**MCANAME**

Message channel agent name (parameter identifier: MQCACH\_MCA\_USER\_ID).

**MSGSENT**

Number of messages sent by the client since it last connected (parameter identifier: MQIACH\_MSGS\_SENT).

**MSGRCVD**

Number of messages received by the client since it last connected (parameter identifier: MQIACH\_MSGS\_RECEIVED / MQIACH\_MSGS\_RCVD).

**INDOUBTIN**

Number of in doubt, inbound messages to the client (parameter identifier: MQIACH\_IN\_DOUBT\_IN).

**INDOUBTOUT**

Number of in doubt, outbound messages to the client (parameter identifier: MQIACH\_IN\_DOUBT\_OUT).

**PENDING**

Number of outbound pending messages (parameter identifier: MQIACH\_PENDING\_OUT).

**LMSGDATE**

Date last message was received or sent (parameter identifier: MQCACH\_LAST\_MSG\_DATE).

**LMSGTIME**

Time last message was received or sent (parameter identifier: MQCACH\_LAST\_MSG\_TIME).

**CHLSDATE**

Date channel started (parameter identifier: MQCACH\_CHANNEL\_START\_DATE).

**CHLSTIME**

Time channel was started (parameter identifier: MQCACH\_CHANNEL\_START\_TIME).

**Error code**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CHANNEL\_NAME\_ERROR**

Channel name error.

**MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

**MQRCCF\_CHL\_INST\_TYPE\_ERROR**

Channel instance type not valid.

**MQRCCF\_CHL\_STATUS\_NOT\_FOUND**

Channel status not found.


**MQRCCF\_XMIT\_Q\_NAME\_ERROR**

Transmission queue name error.

**MQCMD\_INQUIRE\_CHANNEL\_STATUS (Inquire Channel Status) Response**

The response to the Inquire Channel Status (MQCMD\_INQUIRE\_CHANNEL\_STATUS) PCF command consists of the response header followed by several structures.

These structures are

- The *ChannelName* structure
-  The *ChannelDisposition* structure (on z/OS only),
- The *ChannelInstanceType* structure
- The *ChannelStatus* structure (except on z/OS channels whose **ChannelInstanceType** parameter has a value of MQOT\_SAVED\_CHANNEL).

- The **ChannelType** structure
- The **ConnectionName** structure
- The **RemoteAppTag** structure
- The **RemoteQMGrName** structure
- The **StopRequested** structure
- The **XmitQName** structure

which are then followed by the requested combination of status attribute parameter structures. One such message is generated for each channel instance found that matches the criteria specified on the command.

**z/OS** On z/OS, before IBM MQ 9.3.0, if the value for *BytesSent* or *BytesReceived* exceeds 999999999, it is wrapped.

On all platforms, from IBM MQ 9.3.0, if the value for *BuffersSent* or *BytesReceived* exceeds 999999999, it is wrapped.

#### Always returned:

**z/OS** *ChannelDisposition, ChannelInstanceType, ChannelName, ChannelStatus, ChannelType, ConnectionName, RemoteAppTag, RemoteQMGrName, StopRequested, SubState, XmitQName*

#### Returned if requested:

*Batches, BatchSize, BatchSizeIndicator, BuffersReceived, BuffersSent, BytesReceived, BytesSent, ChannelMonitoring, ChannelStartDate, ChannelStartTime, CompressionRate, CompressionTime, CurrentLUWID, CurrentMsgs, CurrentSequenceNumber, CurrentSharingConversations, ExitTime, HeaderCompression, HeartbeatInterval, InDoubtStatus, KeepAliveInterval, LastLUWID, LastMsgDate, LastMsgTime, LastSequenceNumber, LocalAddress, LongRetriesLeft, MaxMsgLength, MaxSharingConversations, MCAJobName, MCAStatus, MCAUserIdentifier, MessageCompression, Msgs, MsgsAvailable, NetTime, NonPersistentMsgSpeed, QMGrName, RemoteVersion, RemoteProduct, SecurityProtocol, ShortRetriesLeft, SSLCertRemoteIssuerName, SSLCertUserId, SSLKeyResetDate, SSLKeyResets, SSLKeyResetTime, SSLShortPeerName, XQTime*

## Response data

### Batches (MQCFIN)

Number of completed batches (parameter identifier: MQIACH\_BATCHES).

### BatchSize (MQCFIN)

Negotiated batch size (parameter identifier: MQIACH\_BATCH\_SIZE).

### BatchSizeIndicator (MQCFIL)

Indicator of the number of messages in a batch (parameter identifier: MQIACH\_BATCH\_SIZE\_INDICATOR). Two values are returned:

- A value based on recent activity over a short period.
- A value based on activity over a longer period.

Where no measurement is available, the value MQMON\_NOT\_AVAILABLE is returned.

### BuffersReceived (MQCFIN)

Number of buffers received (parameter identifier: MQIACH\_BUFFERS\_RCVD).

### BuffersSent (MQCFIN)

Number of buffers sent (parameter identifier: MQIACH\_BUFFERS\_SENT).

### BytesReceived (MQCFIN)

Number of bytes received (parameter identifier: MQIACH\_BYTES\_RCVD).

If the value for *BytesSent* or *BytesReceived* exceeds 999999999, it wraps to zero.



**BytesSent (MQCFIN)**

Number of bytes sent (parameter identifier: MQIACH\_BYTES\_SENT).

If the value for BytesSent or BytesReceived exceeds 999999999, it wraps to zero.

**z/OS ChannelDisposition (MQCFIN)**

Channel disposition (parameter identifier: MQIACH\_CHANNEL\_DISP). This parameter is valid only on z/OS.

The value can be any of the following values:

**MQCHLD\_PRIVATE**

Status information for a private channel.

**MQCHLD\_SHARED**

Status information for a shared channel.

**MQCHLD\_FIXSHARED**

Status information for a shared channel, tied to a specific queue manager.

**ChannelInstanceType (MQCFIN)**

Channel instance type (parameter identifier: MQIACH\_CHANNEL\_INSTANCE\_TYPE).

The value can be any of the following values:

**MQOT\_CURRENT\_CHANNEL**

Current channel status.

**MQOT\_SAVED\_CHANNEL**

Saved channel status.

**z/OS MQOT\_SHORT\_CHANNEL**

Short channel status, only on z/OS.

**ChannelMonitoring (MQCFIN)**

Current level of monitoring data collection for the channel (parameter identifier: MQIA\_MONITORING\_CHANNEL).

The value can be any of the following values:

**MQMON\_OFF**

Monitoring for the channel is disabled.

**MQMON\_LOW**

Low rate of data collection.

**MQMON\_MEDIUM**

Medium rate of data collection.

**MQMON\_HIGH**

High rate of data collection.

**ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**ChannelStartDate (MQCFST)**

Date channel started, in the form yyyy-mm-dd (parameter identifier: MQCACH\_CHANNEL\_START\_DATE).

The maximum length of the string is MQ\_CHANNEL\_DATE\_LENGTH.

**ChannelStartTime (MQCFST)**

Time channel started, in the form hh.mm.ss (parameter identifier: MQCACH\_CHANNEL\_START\_TIME).

The maximum length of the string is MQ\_CHANNEL\_TIME\_LENGTH.

**ChannelStatistics (MQCFIN)**

Specifies whether statistics data is to be collected for channels (parameter identifier: MQIA\_STATISTICS\_CHANNEL).

The value can be:

**MQMON\_OFF**

Statistics data collection is turned off.

**MQMON\_LOW**

Statistics data collection is turned on, with a low ratio of data collection.

**MQMON\_MEDIUM**

Statistics data collection is turned on, with a moderate ratio of data collection.

**MQMON\_HIGH**

Statistics data collection is turned on, with a high ratio of data collection.

On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

This parameter is valid only on z/OS.

**ChannelStatus (MQCFIN)**

Channel status (parameter identifier: MQIACH\_CHANNEL\_STATUS).

Channel status has the following values defined:

**MQCHS\_BINDING**

Channel is negotiating with the partner.

**MQCHS\_STARTING**

Channel is waiting to become active.

**MQCHS\_RUNNING**

Channel is transferring or waiting for messages.

**MQCHS\_PAUSED**

Channel is paused.

**MQCHS\_STOPPING**

Channel is in process of stopping.

**MQCHS\_RETRYING**

Channel is reattempting to establish connection.

**MQCHS\_STOPPED**

Channel is stopped.

**MQCHS\_REQUESTING**

Requester channel is requesting connection.

**MQCHS\_SWITCHING**

Channel is switching transmission queues.

**MQCHS\_INITIALIZING**

Channel is initializing.

**ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

The value can be any of the following values:

**MQCHT\_SENDER**

Sender.

**MQCHT\_SERVER**

Server.

**MQCHT\_RECEIVER**

Receiver.

**MQCHT\_REQUESTER**

Requester.

**MQCHT\_SVRCONN**

Server-connection (for use by clients).

**MQCHT\_CLNTCONN**

Client connection.

**MQCHT\_CLUSRCVR**

Cluster-receiver.

**MQCHT\_CLUSSDR**

Cluster-sender.

**CompressionRate (MQCFIL)**

The compression rate achieved displayed to the nearest percentage (parameter identifier: MQIACH\_COMPRESSION\_RATE). Two values are returned:

- A value based on recent activity over a short period.
- A value based on activity over a longer period.

Where no measurement is available, the value MQMON\_NOT\_AVAILABLE is returned.

**CompressionTime (MQCFIL)**


The amount of time per message, displayed in microseconds, spent during compression or decompression (parameter identifier: MQIACH\_COMPRESSION\_TIME). Two values are returned:


- A value based on recent activity over a short period.
- A value based on activity over a longer period.

Where no measurement is available, the value MQMON\_NOT\_AVAILABLE is returned.

**ConnectionName (MQCFST)**

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).

 On Multiplatforms, the maximum length of the string is 264.

 On z/OS, the maximum length of the string is 48.

**CurrentLUWID (MQCFST)**

Logical unit of work identifier for in-doubt batch (parameter identifier: MQCACH\_CURRENT\_LUWID).

The logical unit of work identifier associated with the current batch, for a sending or a receiving channel.

For a sending channel, when the channel is in-doubt it is the LUWID of the in-doubt batch.

It is updated with the LUWID of the next batch when it is known.

The maximum length is MQ\_LUWID\_LENGTH.

**CurrentMsgs (MQCFIN)**

Number of messages in-doubt (parameter identifier: MQIACH\_CURRENT\_MSGS).

For a sending channel, this parameter is the number of messages that have been sent in the current batch. It is incremented as each message is sent, and when the channel becomes in-doubt it is the number of messages that are in-doubt.

For a receiving channel, it is the number of messages that have been received in the current batch. It is incremented as each message is received.

The value is reset to zero, for both sending and receiving channels, when the batch is committed.

**CurrentSequenceNumber (MQCFIN)**

Sequence number of last message in in-doubt batch (parameter identifier: MQIACH\_CURRENT\_SEQ\_NUMBER).

For a sending channel, this parameter is the message sequence number of the last message sent. It is updated as each message is sent, and when the channel becomes in-doubt it is the message sequence number of the last message in the in-doubt batch.

For a receiving channel, it is the message sequence number of the last message that was received. It is updated as each message is received.

### **CurrentSharingConversations (MQCFIN)**

Number of conversations currently active on this channel instance (parameter identifier: MQIACH\_CURRENT\_SHARING\_CONVS).

This parameter is returned only for TCP/IP server-connection channels.

A value of zero indicates that the channel instance is running in a mode before IBM WebSphere MQ 7.0, regarding:

- Administrator stop-quietce
- Heartbeating
- Read ahead
- Client asynchronous consumption

### **ExitTime (MQCFIL)**

Indicator of the time taken executing user exits per message (parameter identifier: MQIACH\_EXIT\_TIME\_INDICATOR). Amount of time, in microseconds, spent processing user exits per message. Where more than one exit is executed per message, the value is the sum of all the user exit times for a single message. Two values are returned:

- A value based on recent activity over a short period.
- A value based on activity over a longer period.

Where no measurement is available, the value MQMON\_NOT\_AVAILABLE is returned.

### **HeaderCompression (MQCFIL)**

Whether the header data sent by the channel is compressed (parameter identifier: MQIACH\_HDR\_COMPRESSION). Two values are returned:

- The default header data compression value negotiated for this channel.
- The header data compression value used for the last message sent. The header data compression value can be altered in a sending channels message exit. If no message has been sent, the second value is MQCOMPRESS\_NOT\_AVAILABLE.

The values can be:

#### **MQCOMPRESS\_NONE**

No header data compression is performed. MQCOMPRESS\_NONE is the default value.

#### **MQCOMPRESS\_SYSTEM**

Header data compression is performed.

#### **MQCOMPRESS\_NOT\_AVAILABLE**

No message has been sent by the channel.

### **HeartbeatInterval (MQCFIN)**

Heartbeat interval (parameter identifier: MQIACH\_HB\_INTERVAL).

### **InDoubtStatus (MQCFIN)**

Whether the channel is currently in doubt (parameter identifier: MQIACH\_INDOUBT\_STATUS).

A sending channel is only in doubt while the sending Message Channel Agent is waiting for an acknowledgment that a batch of messages, which it has sent, has been successfully received. It is not in doubt at all other times, including the period during which messages are being sent, but before an acknowledgment has been requested.

A receiving channel is never in doubt.

The value can be any of the following values:

**MQCHIDS\_NOT\_INDOUBT**

Channel is not in-doubt.

**MQCHIDS\_INDOUBT**

Channel is in-doubt.

**KeepAliveInterval (MQCFIN)**

KeepAlive interval (parameter identifier: MQIACH\_KEEP\_ALIVE\_INTERVAL). This parameter is valid only on z/OS.

**LastLUWID (MQCFST)**

Logical unit of work identifier for last committed batch (parameter identifier: MQCACH\_LAST\_LUWID).

The maximum length is MQ\_LUWID\_LENGTH.

**LastMsgDate (MQCFST)**

Date last message was sent, or MQI call was handled, in the form yyyy-mm-dd (parameter identifier: MQCACH\_LAST\_MSG\_DATE).

The maximum length of the string is MQ\_CHANNEL\_DATE\_LENGTH.

**LastMsgTime (MQCFST)**

Time last message was sent, or MQI call was handled, in the form hh.mm.ss (parameter identifier: MQCACH\_LAST\_MSG\_TIME).

The maximum length of the string is MQ\_CHANNEL\_TIME\_LENGTH.

**LastSequenceNumber (MQCFIN)**

Sequence number of last message in last committed batch (parameter identifier: MQIACH\_LAST\_SEQ\_NUMBER).

**LocalAddress (MQCFST)**

Local communications address for the channel (parameter identifier: MQCACH\_LOCAL\_ADDRESS).

The maximum length of the string is MQ\_LOCAL\_ADDRESS\_LENGTH.

**LongRetriesLeft (MQCFIN)**

Number of long retry attempts remaining (parameter identifier: MQIACH\_LONG\_RETRIES\_LEFT).

**MaxMsgLength (MQCFIN)**

Maximum message length (parameter identifier: MQIACH\_MAX\_MSG\_LENGTH). This parameter is valid only on z/OS.

**MaxSharingConversations (MQCFIN)**

Maximum number of conversations permitted on this channel instance. (parameter identifier: MQIACH\_MAX\_SHARING\_CONVS)

This parameter is returned only for TCP/IP server-connection channels.

A value of zero indicates that the channel instance is running in a mode before IBM WebSphere MQ 7.0, regarding:

- Administrator stop-quiesce
- Heartbeating
- Read ahead
- Client asynchronous consumption

**MCAJobName (MQCFST)**

Name of MCA job (parameter identifier: MQCACH\_MCA\_JOB\_NAME).

The maximum length of the string is MQ\_MCA\_JOB\_NAME\_LENGTH.

**MCAStatus (MQCFIN)**

MCA status (parameter identifier: MQIACH\_MCA\_STATUS).

The value can be any of the following values:

**MQMCAS\_STOPPED**

Message channel agent stopped.

**MQMCAS\_RUNNING**

Message channel agent running.

**MCAUserIdentifier (MQCFST)**

The user ID used by the MCA (parameter identifier: MQCACH\_MCA\_USER\_ID).

This parameter applies only to server-connection, receiver, requester, and cluster-receiver channels.

The maximum length of the string is MQ\_MCA\_USER\_ID\_LENGTH.

**MessageCompression (MQCFIL)**

Whether the message data sent by the channel is compressed (parameter identifier: MQIACH\_MSG\_COMPRESSION). Two values are returned:

- The default message data compression value negotiated for this channel.
- The message data compression value used for the last message sent. The message data compression value can be altered in a sending channels message exit. If no message has been sent, the second value is MQCOMPRESS\_NOT\_AVAILABLE.

The values can be:

**MQCOMPRESS\_NONE**

No message data compression is performed. MQCOMPRESS\_NONE is the default value.

**MQCOMPRESS\_RLE**

Message data compression is performed using run-length encoding.

**MQCOMPRESS\_ZLIBFAST**

Message data compression is performed using ZLIB encoding with speed prioritized.

**MQCOMPRESS\_ZLIBHIGH**

Message data compression is performed using ZLIB encoding with compression prioritized.

**MQCOMPRESS\_NOT\_AVAILABLE**

No message has been sent by the channel.

**Msgs (MQCFIN)**

Number of messages sent or received, or number of MQI calls handled (parameter identifier: MQIACH\_MSGS).

**MsgsAvailable (MQCFIN)**

Number of messages available (parameter identifier: MQIACH\_XMITQ\_MSGS\_AVAILABLE). Number of messages queued on the transmission queue available to the channel for MQGETs.

Where no measurement is available, the value MQMON\_NOT\_AVAILABLE is returned.

This parameter applies to cluster sender channels only.

**NetTime (MQCFIL)**

Indicator of the time of a network operation (parameter identifier: MQIACH\_NETWORK\_TIME\_INDICATOR). Amount of time, in microseconds, to send a request to the remote end of the channel and receive a response. This time only measures the network time for such an operation. Two values are returned:

- A value based on recent activity over a short period.
- A value based on activity over a longer period.

Where no measurement is available, the value MQMON\_NOT\_AVAILABLE is returned.

**NonPersistentMsgSpeed (MQCFIN)**

Speed at which nonpersistent messages are to be sent (parameter identifier: MQIACH\_NPM\_SPEED).

The value can be any of the following values:

**MQNPMS\_NORMAL**

Normal speed.

**MQNPMS\_FAST**

Fast speed.

**QMgrName (MQCFST)**

Name of the queue manager that owns the channel instance (parameter identifier: MQCA\_Q\_MGR\_NAME). This parameter is valid only on z/OS.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**RemoteApplTag (MQCFST)**

The remote partner application name. This parameter is the name of the client application at the remote end of the channel. This parameter applies only to server-connection channels (parameter identifier: MQCACH\_REMOTE\_APPL\_TAG).

**RemoteProduct (MQCFST)**

The remote partner product identifier. This parameter is the product identifier of the IBM MQ code running at the remote end of the channel (parameter identifier: MQCACH\_REMOTE\_PRODUCT).

The possible values are shown in the following table:

<i>Table 209. Product Identifier values</i>	
<b>Product Identifier</b>	<b>Description</b>
MQMM	Queue Manager (non z/OS Platform)
MQMV	Queue Manager on z/OS
MQCC	IBM MQ C client
MQNM	IBM MQ .NET fully managed client
MQJB	IBM MQ Classes for JAVA
MQJM	IBM MQ Classes for JMS (normal mode)
MQJN	IBM MQ Classes for JMS (migration mode)
MQJU	Common Java interface to the MQI
MQXC	XMS client C/C++ (normal mode)
MQXD	XMS client C/C++ (migration mode)
MQXN	XMS client .NET (normal mode)
MQXM	XMS client .NET (migration mode)
MQXU	IBM MQ .NET XMS client (unmanaged/XA)
MQNU	IBM MQ .NET unmanaged client

**RemoteVersion (MQCFST)**

The remote partner version. This parameter is the version of the IBM MQ code running at the remote end of the channel (parameter identifier: MQCACH\_REMOTE\_VERSION).

The remote version is displayed as **VVRRMMFF**, where

**VV**

Version

**RR**

Release

**MM**

Maintenance level

**FF**

Fix level

**RemoteQMgrName (MQCFST)**

Name of the remote queue manager, or queue sharing group (parameter identifier: MQCA\_REMOTE\_Q\_MGR\_NAME).

**ShortRetriesLeft (MQCFIN)**

Number of short retry attempts remaining (parameter identifier: MQIACH\_SHORT\_RETRIES\_LEFT).

**SecurityProtocol (MQCFIN)**

Defines the security protocol currently in use (parameter identifier: MQIACH\_SECURITY\_PROTOCOL).

Does not apply to client-connection channels.

Set automatically, based on the value you set for [SSLCipherSpecification](#).

Possible values are:

**MQSECPROT\_NONE**

No security protocol

**Deprecated MQSECPROT\_SSLV30**

SSL 3.0

This protocol is deprecated. See [Deprecated CipherSpecs](#)

**Deprecated MQSECPROT\_TLSV10**

TLS 1.0

This protocol is deprecated. See [Deprecated CipherSpecs](#)

**MQSECPROT\_TLSV12**

TLS 1.2

**ALW MQSECPROT\_TLSV13**

TLS 1.3

From IBM MQ 9.2.0, the product supports the TLS 1.3 security protocol on all platforms.

**z/OS** On IBM MQ for z/OS, TLS 1.3 is supported only on z/OS 2.4 or later.

**SSLCertRemoteIssuerName (MQCFST)**

The full Distinguished Name of the issuer of the remote certificate. The issuer is the certificate authority that issued the certificate (parameter identifier: MQCACH\_SSL\_CERT\_ISSUER\_NAME).

The maximum length of the string is MQ\_SHORT\_DNAME\_LENGTH.

**SSLCertUserId (MQCFST)**

The local user ID associated with the remote certificate (parameter identifier: MQCACH\_SSL\_CERT\_USER\_ID).

This parameter is valid only on z/OS.

The maximum length of the string is MQ\_USER\_ID\_LENGTH.

**SSLCipherSpecification (MQCFST)**

The CipherSpec that is being used by the connection (parameter identifier: MQCACH\_SSL\_CIPHER\_SPEC).

The maximum length of the string is MQ\_SSL\_CIPHER\_SPEC\_LENGTH.

For more information, see [the SSLCipherSpec property in Change, Copy, and Create Channel](#).

The value for this parameter is also used to set the value of [SecurityProtocol](#)

**SSLKeyResetDate (MQCFST)**

Date of the previous successful TLS secret key reset, in the form yyyy-mm-dd (parameter identifier: MQCACH\_SSL\_KEY\_RESET\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

**SSLKeyResets (MQCFIN)**

TLS secret key resets (parameter identifier: MQIACH\_SSL\_KEY\_RESETS).



The number of successful TLS secret key resets that have occurred for this channel instance since the channel started. If TLS secret key negotiation is enabled, the count is incremented whenever a secret key reset is performed.

**SSLKeyResetTime (MQCFST)**

Time of the previous successful TLS secret key reset, in the form hh.mm.ss (parameter identifier: MQCACH\_SSL\_KEY\_RESET\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**SSLShortPeerName (MQCFST)**

Distinguished Name of the peer queue manager or client at the other end of the channel (parameter identifier: MQCACH\_SSL\_SHORT\_PEER\_NAME).

The maximum length is MQ\_SHORT\_DNAME\_LENGTH, so longer Distinguished Names are truncated.

**StopRequested (MQCFIN)**

Whether user stop request is outstanding (parameter identifier: MQIACH\_STOP\_REQUESTED).

The value can be any of the following values:

**MQCHSR\_STOP\_NOT\_REQUESTED**

User stop request has not been received.

**MQCHSR\_STOP\_REQUESTED**

User stop request has been received.

**SubState (MQCFIN)**

Current action being performed by the channel (parameter identifier: MQIACH\_CHANNEL\_SUBSTATE).

The value can be any of the following values:

**MQCHSSTATE\_CHADEXIT**

Running channel auto-definition exit.

**MQCHSSTATE\_COMPRESSING**

Compressing or decompressing data.

**MQCHSSTATE\_END\_OF\_BATCH**

End of batch processing.

**MQCHSSTATE\_HANDSHAKING**

TLS handshaking.

**MQCHSSTATE\_HEARTBEATING**

Heartbeating with partner.

**MQCHSSTATE\_IN\_MQGET**

Performing MQGET.

**MQCHSSTATE\_IN\_MQI\_CALL**

Executing an IBM MQ API call, other than an MQPUT or MQGET.

**MQCHSSTATE\_IN\_MQPUT**

Performing MQPUT.

**MQCHSSTATE\_MREXIT**

Running retry exit.

**MQCHSSTATE\_MSGEXIT**

Running message exit.

**MQCHSSTATE\_NAME\_SERVER**

Name server request.

**MQCHSSTATE\_NET\_CONNECTING**

Network connect.

**MQCHSSTATE\_OTHER**

Undefined state.

**MQCHSSTATE\_RCVEXIT**

Running receive exit.

**MQCHSSTATE\_RECEIVING**

Network receive.

**MQCHSSTATE\_RESYNCHING**

Resynching with partner.

**MQCHSSTATE\_SCYEXIT**

Running security exit.

**MQCHSSTATE\_SENDEXIT**

Running send exit.

**MQCHSSTATE\_SENDING**

Network send.

**MQCHSSTATE\_SERIALIZING**

Serialized on queue manager access.

**XmitQName (MQCFST)**

Transmission queue name (parameter identifier: MQCACH\_XMIT\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**XQTime (MQCFIL)**

This parameter applies to only sender, server, and cluster-sender channels.

Transmission queue time indicator (parameter identifier: MQIACH\_XMITQ\_TIME\_INDICATOR). The time, in microseconds, that messages remained on the transmission queue before being retrieved. The time is measured from when the message is put onto the transmission queue until it is retrieved to be sent on the channel and, therefore, includes any interval caused by a delay in the putting application.

Two values are returned:

- A value based on recent activity over a short period.
- A value based on activity over a longer period.

Where no measurement is available, the value MQMON\_NOT\_AVAILABLE is returned.

**Related reference**

[“DISPLAY CHSTATUS \(display channel status\)” on page 685](#)

Use the MQSC command **DISPLAY CHSTATUS** to display the status of one or more channels.

## **MQCMD\_INQUIRE\_CHANNEL\_STATUS (inquire channel status) Response AMQP on AIX, Linux, and Windows**

The response to the Inquire Channel Status (MQCMD\_INQUIRE\_CHANNEL\_STATUS) PCF command consists of the response header followed by the *ChannelName* structure and the requested combination of attribute parameter structures.

One PCF response message is generated for each channel instance found that matches the criteria that are specified on the command.

If the **ClientIdentifier** parameter is not specified, the output of the Inquire Channel Status command is a summary of statuses of all clients that are connected to the channel. One PCF response message is returned per channel.

**Always returned:**

*ChannelName, ChannelStatus, ChannelType,*

If the **ClientIdentifier** parameter is specified, separate PCF response messages are returned for each client connection. The **ClientIdentifier** parameter might be a wildcard, in which the status for all clients that match the **ClientIdentifier** string is returned.

**Always returned:**

*ChannelName, ChannelStatus, ChannelType, ClientIdentifier*

**Returned if requested:**

*ChannelStartDate, ChannelStartTime, ClientUser, ConnectionName, Connections, KeepAliveInterval, LastMsgDate, LastMsgTime, MCAUser, MsgsReceived, MsgsSent, Protocol*

**Response data****ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**ChannelStartDate (MQCFST)**

Date on which the channel started, in the form yyyy-mm-dd (parameter identifier: MQCACH\_CHANNEL\_START\_DATE).

The maximum length of the string is MQ\_CHANNEL\_DATE\_LENGTH.

**ChannelStartTime (MQCFST)**

Time at which the channel started, in the form hh.mm.ss (parameter identifier: MQCACH\_CHANNEL\_START\_TIME).

The maximum length of the string is MQ\_CHANNEL\_TIME\_LENGTH.

**ChannelStatus (MQCFIN)**

Channel status (parameter identifier: MQIACH\_CHANNEL\_STATUS).

The value can be:

**MQCHS\_DISCONNECTED**

Channel is disconnected.

**MQCHS\_RUNNING**

Channel is transferring or waiting for messages.

**ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

The value must be:

**MQCHT\_AMQP**

AMQP

**ClientUser (MQCFST)**

Client Id of the client (parameter identifier: MQCACH\_CLIENT\_USER\_ID).

The maximum length of the string is MQ\_CLIENT\_USER\_ID\_LENGTH.

**ConnectionName (MQCFST)**

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH.

**Connections (MQCFIN)**

Current number of AMQP connections connected to this channel (parameter identifier: MQIACF\_NAME\_LENGTH).

**KeepAliveInterval (MQCFIN)**

Keep alive interval (parameter identifier: MQIACH\_KEEP\_ALIVE\_INTERVAL).

The interval in milliseconds after which the client is disconnected because of inactivity.

**LastMsgDate (MQCFST)**

Date on which the last message was sent, or the MQI call was handled, in the form yyyy-mm-dd (parameter identifier: MQCACH\_LAST\_MSG\_DATE).

The maximum length of the string is MQ\_CHANNEL\_DATE\_LENGTH.

**LastMsgTime (MQCFST)**

Time at which the last message was sent, or the MQI call was handled, in the form hh.mm.ss (parameter identifier: MQCACH\_LAST\_MSG\_TIME).

The maximum length of the string is MQ\_CHANNEL\_TIME\_LENGTH.

**MCAUser (MQCFST)**

Message channel agent user identifier (parameter identifier: MQCACH\_MCA\_USER\_ID).

The maximum length of the MCA user identifier is MQ\_MCA\_USER\_ID\_LENGTH.

**MsgsReceived (MQCFIN64)**

Number of messages received by the client since it last connected (parameter identifier: MQIACH\_MSGS\_RECEIVED or MQIACH\_MSGS\_RCVD).

**MsgsSent (MQCFIN64)**

Number of messages sent by the client since it last connected (parameter identifier: MQIACH\_MSGS\_SENT).

**Protocol (MQCFST)**

AMQP protocol supported by this channel (parameter identifier: MQIACH\_PROTOCOL).

The value will be:

**MQPROTO\_AMQP**  
AMQP

**ALW MQCMD\_INQUIRE\_CHANNEL\_STATUS (Inquire Channel Status) Response**  
**MQTT on AIX, Linux, and Windows**

The response to the Inquire Channel Status (MQCMD\_INQUIRE\_CHANNEL\_STATUS) PCF command consists of the response header followed by the *ChannelName* structure and the requested combination of attribute parameter structures.

One PCF response message is generated for each channel instance found that matches the criteria that are specified on the command.

If the **ClientIdentifier** parameter is not specified, the output of the Inquire Channel Status command is a summary of statuses of all clients that are connected to the channel. One PCF response message is returned per channel.

**Always returned:**

*ChannelName, ChannelStatus, ChannelType, Connections,*

If the **ClientIdentifier** parameter is specified, separate PCF response messages are returned for each client connection. The **ClientIdentifier** parameter might be a wildcard, in which the status for all clients that match the **ClientIdentifier** string is returned (within the limits of **MaxResponses** and **ResponseRestartPoint** if they are set).

**Always returned:**

*ChannelName, ChannelStatus, ChannelType, ClientId*

**Returned if requested:**

*ChannelStatusDate, ChannelStatusTime, ClientUser, InDoubtInput, InDoubtOutput, KeepAliveInterval, LastMessageSentDate, LastMessageSentTime, MCAUser, MessagesReceived, MessagesSent, PendingOutbound, Protocol*

**Response data**

**ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**ChannelStartDate (MQCFST)**

Date on which the channel started, in the form yyyy-mm-dd (parameter identifier: MQCACH\_CHANNEL\_START\_DATE).

The maximum length of the string is MQ\_CHANNEL\_DATE\_LENGTH.

**ChannelStartTime (MQCFST)**

Time at which the channel started, in the form hh.mm.ss (parameter identifier: MQCACH\_CHANNEL\_START\_TIME).

The maximum length of the string is MQ\_CHANNEL\_TIME\_LENGTH.

**ChannelStatus (MQCFIN)**

Channel status (parameter identifier: MQIACH\_CHANNEL\_STATUS).

The value can be:

**MQCHS\_DISCONNECTED**

Channel is disconnected.

**MQCHS\_RUNNING**

Channel is transferring or waiting for messages.

**ChannelType (MQCFIN)**

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

The value must be:

**MQCHT\_MQTT**

Telemetry.

**ClientUser (MQCFST)**

ClientID of the client (parameter identifier: MQCACH\_CLIENT\_USER\_ID).

The maximum length of the string is MQ\_CLIENT\_USER\_ID\_LENGTH.

**ConnectionName (MQCFST)**

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH.

**Connections (MQCFIN)**

Current number of MQTT connections connected to this channel (parameter identifier: MQIACH\_NAME\_LENGTH).

**InDoubtInput (MQCFIN)**

The number of inbound messages to the client that are in doubt (parameter identifier: MQIACH\_IN\_DOUBT\_IN).

**InDoubtOutput (MQCFIN)**

The number of outbound messages from the client that are in doubt (parameter identifier: MQIACH\_IN\_DOUBT\_OUT).

**KeepAliveInterval (MQCFIN)**

KeepAlive interval (parameter identifier: MQIACH\_KEEP\_ALIVE\_INTERVAL).

The interval in milliseconds after which the client is disconnected because of inactivity. If the MQXR service does not receive any communication from the client within the keep alive interval, it disconnects from the client. This interval is calculated based on the MQTT keep alive time sent by the client when it connects. The maximum size is MQ\_MQTT\_MAX\_KEEP\_ALIVE.

**LastMsgDate (MQCFST)**

Date on which the last message was sent, or the MQI call was handled, in the form yyyy-mm-dd (parameter identifier: MQCACH\_LAST\_MSG\_DATE).

The maximum length of the string is MQ\_CHANNEL\_DATE\_LENGTH.

**LastMsgTime (MQCFST)**

Time at which the last message was sent, or the MQI call was handled, in the form hh.mm.ss (parameter identifier: MQCACH\_LAST\_MSG\_TIME).

The maximum length of the string is MQ\_CHANNEL\_TIME\_LENGTH.

**MCAUser (MQCFST)**

Message channel agent user identifier (parameter identifier: MQCACH\_MCA\_USER\_ID).

The maximum length of the MCA user identifier is MQ\_MCA\_USER\_ID\_LENGTH.

**MsgsReceived (MQCFIN64)**

Number of messages received by the client since it last connected (parameter identifier: MQIACH\_MSGS\_RECEIVED / MQIACH\_MSGS\_RCVD).

**MsgsSent (MQCFIN64)**

Number of messages sent by the client since it last connected (parameter identifier: MQIACH\_MSGS\_SENT).

**PendingOutbound (MQCFIN)**

The number of outbound messages pending (parameter identifier: MQIACH\_PENDING\_OUT).

**Protocol (MQCFST)**

MQTT protocol supported by this channel (parameter identifier: MQIACH\_PROTOCOL).

Specify one or more of the following options. To specify more than one option, either add the values together (do not add the same constant more than once), or combine the values using the bitwise OR operation (if the programming language supports bit operations).

MQTTv311 (constant: MQPROTO\_MQTTV311)

MQTTv3 (constant: MQPROTO\_MQTTV3)

HTTP (constant: MQPROTO\_HTTP)

***MQCMD\_INQUIRE\_CHLAUTH\_RECS (Inquire Channel Authentication Records)***

The Inquire Channel Authentication Records (MQCMD\_INQUIRE\_CHLAUTH\_RECS) PCF command retrieves the allowed partner details and mappings to MCAUSER for a channel or set of channels.

**Required parameters****generic-channel-name (MQCFST)**

The name of the channel or set of channels on which you are inquiring (parameter identifier: MQCACH\_CHANNEL\_NAME).

You can use the asterisk (\*) as a wildcard to specify a set of channels, unless you set Match to MQMATCH\_RUNCHECK. If you set Type to BLOCKADDR, you must set the generic channel name to a single asterisk, which matches all channel names.

**Optional parameters****Address (MQCFST)**

The IP address to be mapped (parameter identifier: MQCACH\_CONNECTION\_NAME).

This parameter is valid only when **Match** is MQMATCH\_RUNCHECK and must not be generic.

**ByteStringFilterCommand (MQCFBF)**

Byte string filter command descriptor. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFBF - PCF byte string filter parameter”](#) on page 1518 for information about using this filter condition.

If you specify a byte string filter, you cannot also specify an integer filter using the

**IntegerFilterCommand** parameter, or a string filter using the **StringFilterCommand** parameter.

**ChannelAuthAttrs (MQCFIL)**

Authority record attributes (parameter identifier: MQIACF\_CHLAUTH\_ATTRS).

You can specify the following value in the attribute list on its own. This is the default value if the parameter is not specified.

**MQIACF\_ALL**

All attributes.

If MQIACF\_ALL is not specified, specify a combination of the following values:

**MQCA\_ALTERATION\_DATE**

Alteration Date.

**MQCA\_ALTERATION\_TIME**

Alteration Time.

**MQCA\_CHLAUTH\_DESC**

Description.

**MQCA\_CUSTOM**

Custom.

**MQCACH\_CONNECTION\_NAME**

IP address filter.

**MQCACH\_MCA\_USER\_ID**

MCA User ID mapped on the record.

**MQIACH\_USER\_SOURCE**

The source of the user ID for this record.

**MQIACH\_WARNING**

Warning mode.

**CheckClient (MQCFIN)**

The user ID and password requirements for the client connection to be successful. The following values are valid:

**MQCHK\_REQUIRED\_ADMIN**

A valid user ID and password are required for the connection to be allowed if you are using a privileged user ID.

Any connections using a non-privileged user ID are not required to provide a user ID and password.

The user ID and password are checked against the user repository details provided in an authentication information object, and supplied on ALTER QMGR in the CONNAUTH field.

If no user repository details are provided, so that user ID and password checking are not enabled on the queue manager, the connection is not successful.

A privileged user is one that has full administrative authorities for IBM MQ. See [Privileged users](#) for more information.

This option is not valid on z/OS platforms.

**MQCHK\_REQUIRED**

A valid user ID and password are required for the connection to be allowed.

The user ID and password are checked against the user repository details provided in an authentication information object and supplied on ALTER QMGR in the CONNAUTH field.

If no user repository details are provided, so that user ID and password checking are not enabled on the queue manager, the connection is not successful.

**MQCHK\_AS\_Q\_MGR**

In order for the connection to be allowed, it must meet the connection authentication requirements defined on the queue manager.

If the CONNAUTH field provides an authentication information object, and the value of CHCKCLNT is REQUIRED, the connection fails unless a valid user ID and password are supplied.

If the CONNAUTH field does not provide an authentication information object, or the value of CHCKCLNT is not REQUIRED, the user ID and password are not required.



**Attention:** If you select MQCHK\_REQUIRED or MQCHK\_REQUIRED\_ADMIN on [Multiplatforms](#) and you have not set the **Connauth** field on the queue manager, or if the value of **CheckClient** is None, the connection fails. On Multiplatforms, you receive message AMQ9793. On z/OS, you receive message CSQX793E.

### ClntUser (MQCFST)

The client asserted user ID to be mapped to a new user ID, allowed through unchanged, or blocked (parameter identifier: MQCACH\_CLIENT\_USER\_ID).

This can be the user ID flowed from the client indicating the user ID the client side process is running under, or the user ID presented by the client on an MQCONN call using MQCSP.

This parameter is valid only with TYPE(USERMAP) and when **Match** is MQMATCH\_RUNCHECK.

z/OS

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following values:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which the command was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

### IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 1523 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a byte string filter using the **ByteStringFilterCommand** parameter or a string filter using the **StringFilterCommand** parameter.

### Match (MQCFIN)

Indicates the type of matching to be applied (parameter identifier MQIACH\_MATCH). You can specify any one of the following values:

#### MQMATCH\_RUNCHECK

A specific match is made against the supplied channel name and optionally supplied **Address**, **SSLPeer**, **QMName**, and **ClntUser** attributes to find the channel authentication record that will be matched by the channel at runtime if it connects into this queue manager. If the record discovered has **Warn** set to MQWARN\_YES, a second record might also be displayed to show the actual record the channel will use at runtime. The channel name supplied in this case cannot be generic. This option must be combined with **Type** MQCAUT\_ALL.

#### MQMATCH\_EXACT

Return only those records which exactly match the channel profile name supplied. If there are no asterisks in the channel profile name, this option returns the same output as MQMATCH\_GENERIC.

#### MQMATCH\_GENERIC

Any asterisks in the channel profile name are treated as wildcards. If there are no asterisks in the channel profile name, this returns the same output as MQMATCH\_EXACT. For example, a profile of ABC\* could result in records for ABC, ABC\*, and ABCD being returned.



## **MQMATCH\_ALL**

Return all possible records that match the channel profile name supplied. If the channel name is generic in this case, all records that match the channel name are returned even if more specific matches exist. For example, a profile of SYSTEM.\*.SVRCONN could result in records for SYSTEM.\*, SYSTEM.DEF.\*, SYSTEM.DEF.SVRCONN, and SYSTEM.ADMIN.SVRCONN being returned.

## **QMName (MQCFST)**

The name of the remote partner queue manager to be matched (parameter identifier: MQCA\_REMOTE\_Q\_MGR\_NAME).

This parameter is valid only when **Match** is MQMATCH\_RUNCHECK. The value cannot be generic.

## **SSLCertIssuer (MQCFST)**

This parameter is additional to the **SSLPeer** parameter.

**SSLCertIssuer** restricts matches to being within certificates issued by a particular Certificate Authority.

## **SSLPeer (MQCFST)**

The Distinguished Name of the certificate to be matched (parameter identifier: MQCACH\_SSL\_PEER\_NAME).

This parameter is valid only when **Match** is MQMATCH\_RUNCHECK.

The **SSLPeer** value is specified in the standard form used to specify a Distinguished Name and cannot be a generic value.

The maximum length of the parameter is MQ\_SSL\_PEER\_NAME\_LENGTH.

## **StringFilterCommand (MQCFSF)**

String filter command descriptor. Use this parameter to restrict the output from the command by specifying a filter condition. See “[MQCFSF - PCF string filter parameter](#)” on page 1530 for information about using this filter condition.

If you specify a string filter, you cannot also specify a byte string filter using the **ByteStringFilterCommand** parameter or an integer filter using the **IntegerFilterCommand** parameter.

## **Type (MQCFIN)**

The type of channel authentication record for which to set allowed partner details or mappings to MCAUSER (parameter identifier: MQIACF\_CHLAUTH\_TYPE). The following values are valid:

### **MQCAUT\_BLOCKUSER**

This channel authentication record prevents a specified user or users from connecting.

### **MQCAUT\_BLOCKADDR**

This channel authentication record prevents connections from a specified IP address or addresses.

### **MQCAUT\_SSLPEERMAP**

This channel authentication record maps TLS Distinguished Names (DNs) to MCAUSER values.

### **MQCAUT\_ADDRESSMAP**

This channel authentication record maps IP addresses to MCAUSER values.

### **MQCAUT\_USERMAP**

This channel authentication record maps asserted user IDs to MCAUSER values.

### **MQCAUT\_QMGRMAP**

This channel authentication record maps remote queue manager names to MCAUSER values.

### **MQCAUT\_ALL**

Inquire on all types of record. This is the default value.

## **Related concepts**

[Channel authentication records](#)

## **MQCMD\_INQUIRE\_CHLAUTH\_RECS (Inquire Channel Authentication Records)**

### **Response**

The response to the Inquire Channel Authentication Records (MQCMD\_INQUIRE\_CHLAUTH\_RECS) PCF command consists of the response header followed by the requested combination of attribute parameter structures.

#### **Always returned:**

*ChlAuth, Type, Warn (yes)*

#### **Always returned if type is MQCAUT\_BLOCKUSER:**

*UserList*

#### **Always returned if type is MQCAUT\_BLOCKADDR:**

*AddrList*

#### **Always returned if type is MQCAUT\_SSLPEERMAP:**

*Address (unless blanks), MCAUser (unless blanks), SSLCertIssuer, SSLPeer, UserSrc*

#### **Always returned if type is MQCAUT\_ADDRESSMAP:**

*Address (unless blanks), MCAUser (unless blanks), UserSrc*

#### **Always returned if type is MQCAUT\_USERMAP:**

*Address (unless blanks), ClntUser, MCAUser (unless blanks), UserSrc*

#### **Always returned if type is MQCAUT\_QMGRMAP:**

*Address (unless blanks), MCAUser (unless blanks), QMName, UserSrc*

#### **Returned if requested:**

*Address, AlterationDate, AlterationTime, Custom, Description, MCAUser, SSLPeer, UserSrc, Warn*

### **Response data**

#### **AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date when the information was last altered, in the form yyyy-mm-dd.

#### **AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time when the information was last altered, in the form hh.mm.ss.

#### **Address (MQCFST)**

The filter used to compare with the IP address, or host name, of the partner queue manager or client at the other end of the channel (parameter identifier: MQCACH\_CONNECTION\_NAME).

#### **AddrList (MQCFSL)**

A list of up to 100 IP address patterns which are banned from accessing this queue manager on any channel (parameter identifier: MQCACH\_CONNECTION\_NAME\_LIST).

#### **Chlauth (MQCFST)**

The name of the channel, or pattern that matches a set of channels, to which the channel authentication record applies (parameter identifier: MQCACH\_CHANNEL\_NAME).

#### **CheckClient (MQCFIN)**

The user ID and password requirements for the client connection to be successful (parameter identifier: MQIA\_CHECK\_CLIENT\_BINDING).

#### **ClntUser (MQCFST)**

The client asserted user ID to be mapped to a new user ID, allowed through unchanged, or blocked (parameter identifier: MQCACH\_CLIENT\_USER\_ID).

#### **Description (MQCFST)**

Descriptive information about the channel authentication record (parameter identifier: MQCA\_CHLAUTH\_DESC).

**MCAUser (MQCFST)**

The user identifier to be used when the inbound connection matches the TLS DN, IP address, client asserted user ID or remote queue manager name supplied (parameter identifier: MQCACH\_MCA\_USER\_ID).

**QMName (MQCFST)**

The name of the remote partner queue manager to be mapped to a user ID, allowed through unchanged, or blocked (parameter identifier: MQCA\_REMOTE\_Q\_MGR\_NAME).

**SSLCertIssuer (MQCFST)**

This parameter is additional to the **SSLPeer** parameter.

**SSLCertIssuer** restricts matches to being within certificates issued by a particular Certificate Authority (parameter identifier: MQCA\_SSL\_CERT\_ISSUER\_NAME).

**SSLPeer (MQCFST)**

The filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel (parameter identifier: MQCACH\_SSL\_PEER\_NAME).

**Type (MQCFIN)**

The type of channel authentication record for which to set allowed partner details or mappings to MCAUSER (parameter identifier: MQIACF\_CHLAUTH\_TYPE). The following values can be returned:

**MQCAUT\_BLOCKUSER**

This channel authentication record prevents a specified user or users from connecting.

**MQCAUT\_BLOCKADDR**

This channel authentication record prevents connections from a specified IP address or addresses.

**MQCAUT\_SSLPEERMAP**

This channel authentication record maps TLS Distinguished Names (DNs) to MCAUSER values.

**MQCAUT\_ADDRESSMAP**

This channel authentication record maps IP addresses to MCAUSER values.

**MQCAUT\_USERMAP**

This channel authentication record maps asserted user IDs to MCAUSER values.

**MQCAUT\_QMGRMAP**

This channel authentication record maps remote queue manager names to MCAUSER values.

**UserList (MQCFSL)**

A list of up to 100 user IDs which are banned from use of this channel or set of channels (parameter identifier: MQCACH\_MCA\_USER\_ID\_LIST). Use the special value \*MQADMIN to mean privileged or administrative users. The definition of this value depends on the operating system, as follows:

- On AIX and Linux, all members of the mqm group.
- On Windows, all members of the mqm group, the Administrators group and SYSTEM.
- On IBM i, the profiles (users) qmqm and qmqmadm and all members of the qmqmadm group, and any user defined with the \*ALLOBJ special setting.
- On z/OS, the user ID that the channel initiator, queue manager and advanced message security address spaces are running under.

**UserSrc (MQCFIN)**

The source of the user ID to be used for MCAUSER at run time (parameter identifier: MQIACH\_USER\_SOURCE).

The following values can be returned:

**MQUSRC\_MAP**

Inbound connections that match this mapping use the user ID specified in the **MCAUser** attribute.

**MQUSRC\_NOACCESS**

Inbound connections that match this mapping have no access to the queue manager and the channel ends immediately.

**MQUSRC\_CHANNEL**

Inbound connections that match this mapping use the flowed user ID or any user defined on the channel object in the MCAUSER field.

**Warn (MQCFIN)**

Indicates whether this record operates in warning mode (parameter identifier: MQIACH\_WARNING).

**MQWARN\_NO**

This record does not operate in warning mode. Any inbound connection that matches this record is blocked. This is the default value.

**MQWARN\_YES**

This record operates in warning mode. Any inbound connection that matches this record and would therefore be blocked is allowed access. An error message is written and, if events are configured, an event message is created showing the details of what would have been blocked. The connection is allowed to continue.

**MQCMD\_INQUIRE\_CLUSTER\_Q\_MGR (Inquire Cluster Queue Manager)**

The Inquire Cluster Queue Manager (MQCMD\_INQUIRE\_CLUSTER\_Q\_MGR) PCF command inquires about the attributes of IBM MQ queue managers in a cluster.

**Required parameters****ClusterQMgrName (MQCFST)**

Queue manager name (parameter identifier: MQCA\_CLUSTER\_Q\_MGR\_NAME).

Generic queue manager names are supported. A generic name is a character string followed by an asterisk "\*", for example ABC\*. It selects all queue managers having names that start with the selected character string. An asterisk on its own matches all possible names.

The queue manager name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**Optional parameters****Channel (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

Specifies that eligible cluster queue managers are limited to those having the specified channel name.

Generic channel names are supported. A generic name is a character string followed by an asterisk "\*", for example ABC\*. It selects all queue managers having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

If you do not specify a value for this parameter, channel information about *all* queue managers in the cluster is returned.

**ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

Specifies that eligible cluster queue managers are limited to those having the specified cluster name.

Generic cluster names are supported. A generic name is a character string followed by an asterisk "\*", for example ABC\*. It selects all queue managers having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

If you do not specify a value for this parameter, cluster information about *all* queue managers inquired is returned.

**ClusterQMgrAttrs (MQCFIL)**

Attributes (parameter identifier: MQIACF\_CLUSTER\_Q\_MGR\_ATTRS).

Some parameters are relevant only for cluster channels of a particular type or types. Attributes that are not relevant for a particular type of channel cause no output, and do not cause an error. To check which attributes apply to which channel types; see [Channel attributes and channel types](#).

The attribute list might specify the following value on its own. If the parameter is not specified, a default value is used.

**MQIACF\_ALL**

All attributes.

Alternative, supply a combination of the following values:

**MQCA\_ALTERATION\_DATE**

The date on which the information was last altered.

**MQCA\_ALTERATION\_TIME**

The time at which the information was last altered.

**MQCA\_CLUSTER\_DATE**

The date on which the information became available to the local queue manager.

**MQCA\_CLUSTER\_NAME**

The name of the cluster to which the channel belongs.

**MQCA\_CLUSTER\_Q\_MGR\_NAME**

The name of the cluster to which the channel belongs.

**MQCA\_CLUSTER\_TIME**

The time at which the information became available to the local queue manager.

**MQCA\_Q\_MGR\_IDENTIFIER**

The unique identifier of the queue manager.

**MQCA\_VERSION**

The version of the IBM MQ installation that the cluster queue manager is associated with.

**MQCA\_XMIT\_Q\_NAME**

The cluster transmission queue used by the queue manager.

**MQCACH\_CONNECTION\_NAME**

Connection name.

**MQCACH\_DESCRIPTION**

Description.

**MQCACH\_LOCAL\_ADDRESS**

Local communications address for the channel.

**MQCACH\_MCA\_NAME**

Message channel agent name.

You cannot use MQCACH\_MCA\_NAME as a parameter to filter on.

**MQCACH\_MCA\_USER\_ID**

MCA user identifier.

**MQCACH\_MODE\_NAME**

Mode name.

**MQCACH\_MR\_EXIT\_NAME**

Message-retry exit name.

**MQCACH\_MR\_EXIT\_USER\_DATA**

Message-retry exit user data.

**MQCACH\_MSG\_EXIT\_NAME**

Message exit name.

**MQCACH\_MSG\_EXIT\_USER\_DATA**

Message exit user data.

**MQCACH\_PASSWORD**

Password.

This parameter is not valid on z/OS.

**MQCACH\_RCV\_EXIT\_NAME**

Receive exit name.

**MQCACH\_RCV\_EXIT\_USER\_DATA**

Receive exit user data.

**MQCACH\_SEC\_EXIT\_NAME**

Security exit name.

**MQCACH\_SEC\_EXIT\_USER\_DATA**

Security exit user data.

**MQCACH\_SEND\_EXIT\_NAME**

Send exit name.

**MQCACH\_SEND\_EXIT\_USER\_DATA**

Send exit user data.

**MQCACH\_SSL\_CIPHER\_SPEC**

TLS cipher spec.

**MQIACH\_SSL\_CLIENT\_AUTH**

TLS client authentication.

**MQCACH\_SSL\_PEER\_NAME**

TLS peer name.

**MQCACH\_TP\_NAME**

Transaction program name.

**MQCACH\_USER\_ID**

User identifier.

This parameter is not valid on z/OS.

**MQIA\_MONITORING\_CHANNEL**

Online monitoring data collection.

**MQIA\_USE\_DEAD\_LETTER\_Q**

Determines whether the dead-letter queue is used when messages cannot be delivered by channels.

**MQIACF\_Q\_MGR\_DEFINITION\_TYPE**

How the cluster queue manager was defined.

**MQIACF\_Q\_MGR\_TYPE**

The function of the queue manager in the cluster.

**MQIACF\_SUSPEND**

Specifies whether the queue manager is suspended from the cluster.

**MQIACH\_BATCH\_HB**

The value being used for the batch heartbeat.

**MQIACH\_BATCH\_INTERVAL**

Batch wait interval (seconds).

**MQIACH\_BATCH\_DATA\_LIMIT**

Batch data limit (kilobytes).

**MQIACH\_BATCH\_SIZE**

Batch size.

**MQIACH\_CHANNEL\_STATUS**

Channel status.

**MQIACH\_CLWL\_CHANNEL\_PRIORITY**

Cluster workload channel priority.

**MQIACH\_CLWL\_CHANNEL\_RANK**

Cluster workload channel rank.

**MQIACH\_CLWL\_CHANNEL\_WEIGHT**

Cluster workload channel weight.

**MQIACH\_DATA\_CONVERSION**

Specifies whether sender must convert application data.

**MQIACH\_DISC\_INTERVAL**

Disconnection interval.

**MQIACH\_HB\_INTERVAL**

Heartbeat interval (seconds).

**MQIACH\_HDR\_COMPRESSION**

The list of header data compression techniques supported by the channel.

**MQIACH\_KEEP\_ALIVE\_INTERVAL**

KeepAlive interval (valid on z/OS only).

**MQIACH\_LONG\_RETRY**

Count of long duration attempts.

**MQIACH\_LONG\_TIMER**

Long duration timer.

**MQIACH\_MAX\_MSG\_LENGTH**

Maximum message length.

**MQIACH\_MCA\_TYPE**

MCA type.

**MQIACH\_MR\_COUNT**

Count of send message attempts.

**MQIACH\_MR\_INTERVAL**

Interval between attempting to resend a message in milliseconds.

**MQIACH\_MSG\_COMPRESSION**

List of message data compression techniques supported by the channel.

**MQIACH\_NETWORK\_PRIORITY**

Network priority.

**MQIACH\_NPM\_SPEED**

Speed of nonpersistent messages.

**MQIACH\_PUT\_AUTHORITY**

Put authority.

**MQIACH\_SEQUENCE\_NUMBER\_WRAP**

Sequence number wrap.

**MQIACH\_SHORT\_RETRY**

Count of short duration attempts.

**MQIACH\_SHORT\_TIMER**

Short duration timer.

**MQIACH\_XMIT\_PROTOCOL\_TYPE**

Transmission protocol type.

**z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following values:

- Blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- A queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the

queue manager on which it was entered, you must be using a queue sharing group environment. The command server must be enabled.

- An asterisk " \* ". The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ClusterQMGrAttrs* except MQIACF\_ALL and others as noted. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 1523 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ClusterQMGrAttrs* except MQCA\_CLUSTER\_Q\_MGR\_NAME and others as noted. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 1530 for information about using this filter condition.

If you specify a string filter for *Channel* or *ClusterName*, you cannot also specify the *Channel* or *ClusterName* parameter.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.


## **MQCMD\_INQUIRE\_CLUSTER\_Q\_MGR (Inquire Cluster Queue Manager) Response**

The response to the Inquire Cluster Queue Manager (MQCMD\_INQUIRE\_CLUSTER\_Q\_MGR) PCF command consists of three parts. The response header is followed by the *QMGrName* structure and the requested combination of attribute parameter structures.

### **Always returned:**

*ChannelName*, *ClusterName*, *QMGrName*,

### **Returned if requested:**

*AlterationDate*, *AlterationTime*, *BatchHeartbeat*, *BatchInterval*, *BatchSize*, *ChannelDesc*, *ChannelMonitoring*, *ChannelStatus*, *ClusterDate*, *ClusterInfo*, *ClusterTime*, *CLWLChannelPriority*, *CLWLChannelRank*, *CLWLChannelWeight*, *ConnectionName*, *DataConversion*, *DiscInterval*, *HeaderCompression*, *HeartbeatInterval*,  *KeepAliveInterval*, *LocalAddress*, *LongRetryCount*, *LongRetryInterval*, *MaxMsgLength*, *MCAName*, *MCAType*, *MCAUserIdentifier*, *MessageCompression*, *ModeName*, *MsgExit*, *MsgRetryCount*, *MsgRetryExit*, *MsgRetryInterval*, *MsgRetryUserData*, *MsgUserData*, *NetworkPriority*, *NonPersistentMsgSpeed*, *Password*, *PutAuthority*, *QMGrDefinitionType*, *QMGrIdentifier*, *QMGrType*, *ReceiveExit*, *ReceiveUserData*, *SecurityExit*, *SecurityUserData*, *SendExit*, *SendUserData*, *SeqNumberWrap*, *ShortRetryCount*, *ShortRetryInterval*, *SSLCipherSpec*, *SSLClientAuth*, *SSLPeerName*, *Suspend*, *TpName*, *TransmissionQName*, *TransportType*, *UseDLQ*, *UserIdentifier*, *Version*

## **Response data**

### **AlterationDate (MQCFST)**

Alteration date, in the form yyyy-mm-dd (parameter identifier: MQCA\_ALTERATION\_DATE).

The date at which the information was last altered.

### **AlterationTime (MQCFST)**

Alteration time, in the form hh.mm.ss (parameter identifier: MQCA\_ALTERATION\_TIME).



The time at which the information was last altered.

**BatchHeartbeat (MQCFIN)**

The value being used for the batch heartbeat (parameter identifier: MQIACH\_BATCH\_HB).

The value can be 0 - 999,999. A value of 0 indicates that the batch heartbeat is not being used.

**BatchInterval (MQCFIN)**

Batch interval (parameter identifier: MQIACH\_BATCH\_INTERVAL).

**BatchSize (MQCFIN)**

Batch size (parameter identifier: MQIACH\_BATCH\_SIZE).

**ChannelDesc (MQCFST)**

Channel description (parameter identifier: MQCACH\_DESC).

The maximum length of the string is MQ\_CHANNEL\_DESC\_LENGTH.

**ChannelMonitoring (MQCFIN)**

Online monitoring data collection (parameter identifier: MQIA\_MONITORING\_CHANNEL).

The value can be:

**MQMON\_OFF**

Online monitoring data collection is turned off for this channel.

**MQMON\_Q\_MGR**

The value of the queue manager's **ChannelMonitoring** parameter is inherited by the channel. MQMON\_Q\_MGR is the default value.

**MQMON\_LOW**

Online monitoring data collection is turned on, with a low rate of data collection, for this channel unless the queue manager's **ChannelMonitoring** parameter is MQMON\_NONE.

**MQMON\_MEDIUM**

Online monitoring data collection is turned on, with a moderate rate of data collection, for this channel unless the queue manager's **ChannelMonitoring** parameter is MQMON\_NONE.

**MQMON\_HIGH**

Online monitoring data collection is turned on, with a high rate of data collection, for this channel unless the queue manager's **ChannelMonitoring** parameter is MQMON\_NONE.

**ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**ChannelStatus (MQCFIN)**

Channel status (parameter identifier: MQIACH\_CHANNEL\_STATUS).

The value can be:

**MQCHS\_BINDING**

Channel is negotiating with the partner.

**MQCHS\_INACTIVE**

Channel is not active.

**MQCHS\_STARTING**

Channel is waiting to become active.

**MQCHS\_RUNNING**

Channel is transferring or waiting for messages.

**MQCHS\_PAUSED**

Channel is paused.

**MQCHS\_STOPPING**

Channel is in process of stopping.

**MQCHS\_RETRYING**

Channel is reattempting to establish connection.

**MQCHS\_STOPPED**

Channel is stopped.

**MQCHS\_REQUESTING**

Requester channel is requesting connection.

**MQCHS\_INITIALIZING**

Channel is initializing.

**ClusterDate (MQCFST)**

Cluster date, in the form yyyy-mm-dd (parameter identifier: MQCA\_CLUSTER\_DATE).

The date at which the information became available to the local queue manager.

**ClusterInfo (MQCFIN)**

Cluster information (parameter identifier: MQIACF\_CLUSTER\_INFO).

The cluster information available to the local queue manager.

**ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

**ClusterTime (MQCFST)**

Cluster time, in the form hh.mm.ss (parameter identifier: MQCA\_CLUSTER\_TIME).

The time at which the information became available to the local queue manager.

**CLWLChannelPriority (MQCFIN)**

Channel priority (parameter identifier: MQIACH\_CLWL\_CHANNEL\_PRIORITY).

**CLWLChannelRank (MQCFIN)**

Channel rank (parameter identifier: MQIACH\_CLWL\_CHANNEL\_RANK).

**CLWLChannelWeight (MQCFIN)**

Channel weighting (parameter identifier: MQIACH\_CLWL\_CHANNEL\_WEIGHT).

**ConnectionName (MQCFST)**

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH. On z/OS, it is MQ\_LOCAL\_ADDRESS\_LENGTH.

**DataConversion (MQCFIN)**

Specifies whether sender must convert application data (parameter identifier: MQIACH\_DATA\_CONVERSION).

The value can be:

**MQCDC\_NO\_SENDER\_CONVERSION**

No conversion by sender.

**MQCDC\_SENDER\_CONVERSION**

Conversion by sender.

**DiscInterval (MQCFIN)**

Disconnection interval (parameter identifier: MQIACH\_DISC\_INTERVAL).

**HeaderCompression (MQCFIL)**

Header data compression techniques supported by the channel (parameter identifier: MQIACH\_HDR\_COMPRESSION). The values specified are in order of preference.

The value can be one, or more, of

**MQCOMPRESS\_NONE**

No header data compression is performed.

**MQCOMPRESS\_SYSTEM**

Header data compression is performed.

**HeartbeatInterval (MQCFIN)**

Heartbeat interval (parameter identifier: MQIACH\_HB\_INTERVAL).



### **KeepAliveInterval (MQCFIN)**

KeepAlive interval (parameter identifier: MQIACH\_KEEP\_ALIVE\_INTERVAL). This parameter applies to z/OS only.

### **LocalAddress (MQCFST)**

Local communications address for the channel (parameter identifier: MQCACH\_LOCAL\_ADDRESS).

The maximum length of the string is MQ\_LOCAL\_ADDRESS\_LENGTH.

### **LongRetryCount (MQCFIN)**

Long retry count (parameter identifier: MQIACH\_LONG\_RETRY).

### **LongRetryInterval (MQCFIN)**

Long timer (parameter identifier: MQIACH\_LONG\_TIMER).

### **MaxMsgLength (MQCFIN)**

Maximum message length (parameter identifier: MQIACH\_MAX\_MSG\_LENGTH).

### **MCAName (MQCFST)**

Message channel agent name (parameter identifier: MQCACH\_MCA\_NAME).

The maximum length of the string is MQ\_MCA\_NAME\_LENGTH.

### **MCAType (MQCFIN)**

Message channel agent type (parameter identifier: MQIACH\_MCA\_TYPE).

The value can be:

#### **MQMCAT\_PROCESS**

Process.

#### **MQMCAT\_THREAD**

Thread (Windows only).

### **MCAUserIdentifier (MQCFST)**

Message channel agent user identifier (parameter identifier: MQCACH\_MCA\_USER\_ID).

The maximum length of the string is MQ\_USER\_ID\_LENGTH.

### **MessageCompression (MQCFIL)**

Message data compression techniques supported by the channel (parameter identifier: MQIACH\_MSG\_COMPRESSION). The values specified are in order of preference.

The value can be one, or more, of:

#### **MQCOMPRESS\_NONE**

No message data compression is performed.

#### **MQCOMPRESS\_RLE**

Message data compression is performed using run-length encoding.

#### **MQCOMPRESS\_ZLIBFAST**

Message data compression is performed using ZLIB encoding with speed prioritized.

#### **MQCOMPRESS\_ZLIBHIGH**

Message data compression is performed using ZLIB encoding with compression prioritized.

### **ModeName (MQCFST)**

Mode name (parameter identifier: MQCACH\_MODE\_NAME).

The maximum length of the string is MQ\_MODE\_NAME\_LENGTH.

### **MsgExit (MQCFST)**

Message exit name (parameter identifier: MQCACH\_MSG\_EXIT\_NAME).

The maximum length of the string is MQ\_EXIT\_NAME\_LENGTH.



On Multiplatforms, more than one message exit can be defined for a channel. If more than one message exit is defined, the list of names is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

**MsgRetryCount (MQCFIN)**

Message retry count (parameter identifier: MQIACH\_MR\_COUNT).

**MsgRetryExit (MQCFST)**

Message retry exit name (parameter identifier: MQCACH\_MR\_EXIT\_NAME).

The maximum length of the string is MQ\_EXIT\_NAME\_LENGTH.

**MsgRetryInterval (MQCFIN)**

Message retry interval (parameter identifier: MQIACH\_MR\_INTERVAL).

**MsgRetryUserData (MQCFST)**

Message retry exit user data (parameter identifier: MQCACH\_MR\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

**MsgUserData (MQCFST)**

Message exit user data (parameter identifier: MQCACH\_MSG\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

**Multi** On Multiplatforms, more than one message exit user data string can be defined for a channel. If more than one string is defined, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

**NetworkPriority (MQCFIN)**

Network priority (parameter identifier: MQIACH\_NETWORK\_PRIORITY).

**NonPersistentMsgSpeed (MQCFIN)**

Speed at which non-persistent messages are to be sent (parameter identifier: MQIACH\_NPM\_SPEED).

The value can be:

**MQNPMS\_NORMAL**

Normal speed.

**MQNPMS\_FAST**

Fast speed.

**Password (MQCFST)**

Password (parameter identifier: MQCACH\_PASSWORD). This parameter is not available on z/OS.

If a nonblank password is defined, it is returned as asterisks. Otherwise, it is returned as blanks.

The maximum length of the string is MQ\_PASSWORD\_LENGTH. However, only the first 10 characters are used.

**PutAuthority (MQCFIN)**

Put authority (parameter identifier: MQIACH\_PUT\_AUTHORITY).

The value can be:

**MQPA\_DEFAULT**

Default user identifier is used.

**MQPA\_CONTEXT**

Context user identifier is used.

**MQPA\_ALTERNATE\_OR\_MCA**

The user identifier from the *UserIdentifier* field of the message descriptor is used. Any user ID received from the network is not used. This value is valid only on z/OS.

**MQPA\_ONLY\_MCA**

The default user identifier is used. Any user ID received from the network is not used. This value is valid only on z/OS.

**QMgrDefinitionType (MQCFIN)**

Queue manager definition type (parameter identifier: MQIACF\_Q\_MGR\_DEFINITION\_TYPE).

The value can be:

**MQQMDT\_EXPLICIT\_CLUSTER\_SENDER**

A cluster-sender channel from an explicit definition.

**MQQMDT\_AUTO\_CLUSTER\_SENDER**

A cluster-sender channel by auto-definition.

**MQQMDT\_CLUSTER\_RECEIVER**

A cluster-receiver channel.

**MQQMDT\_AUTO\_EXP\_CLUSTER\_SENDER**

A cluster-sender channel, both from an explicit definition and by auto-definition.

**QMgrIdentifier (MQCFST)**

Queue manager identifier (parameter identifier: MQCA\_Q\_MGR\_IDENTIFIER).

The unique identifier of the queue manager.

**QMgrName (MQCFST)**

Queue manager name (parameter identifier: MQCA\_CLUSTER\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**QMgrType (MQCFIN)**

Queue manager type (parameter identifier: MQIACF\_Q\_MGR\_TYPE).

The value can be:

**MQQMT\_NORMAL**

A normal queue manager.

**MQQMT\_REPOSITORY**

A repository queue manager.

**ReceiveExit (MQCFST)**

Receive exit name (parameter identifier: MQCACH\_RCV\_EXIT\_NAME).

The maximum length of the string is MQ\_EXIT\_NAME\_LENGTH.

**Multi** On Multiplatforms, more than one receive exit can be defined for a channel. If more than one receive exit is defined, the list of names is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

**ReceiveUserData (MQCFST)**

Receive exit user data (parameter identifier: MQCACH\_RCV\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

**Multi** On Multiplatforms, more than one receive exit user data string can be defined for a channel. If more than one string is defined, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

**SecurityExit (MQCFST)**

Security exit name (parameter identifier: MQCACH\_SEC\_EXIT\_NAME).

The maximum length of the string is MQ\_EXIT\_NAME\_LENGTH.

**SecurityUserData (MQCFST)**

Security exit user data (parameter identifier: MQCACH\_SEC\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

### **SendExit (MQCFST)**

Send exit name (parameter identifier: MQCACH\_SEND\_EXIT\_NAME).

The maximum length of the string is MQ\_EXIT\_NAME\_LENGTH.

**Multi** On Multiplatforms, more than one send exit can be defined for a channel. If more than one send exit is defined, the list of names is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

### **SendUserData (MQCFST)**

Send exit user data (parameter identifier: MQCACH\_SEND\_EXIT\_USER\_DATA).

The maximum length of the string is MQ\_EXIT\_DATA\_LENGTH.

**Multi** On Multiplatforms, more than one send exit user data string can be defined for a channel. If more than one string is defined, the list of names is returned in an MQCFSL structure instead of an MQCFST structure.

**z/OS** On z/OS, an MQCFSL structure is always used.

### **SeqNumberWrap (MQCFIN)**

Sequence wrap number (parameter identifier: MQIACH\_SEQUENCE\_NUMBER\_WRAP).

### **ShortRetryCount (MQCFIN)**

Short retry count (parameter identifier: MQIACH\_SHORT\_RETRY).

### **ShortRetryInterval (MQCFIN)**

Short timer (parameter identifier: MQIACH\_SHORT\_TIMER).

### **SSLCipherSpec (MQCFST)**

CipherSpec (parameter identifier: MQCACH\_SSL\_CIPHER\_SPEC).

The length of the string is MQ\_SSL\_CIPHER\_SPEC\_LENGTH.

### **SSLClientAuth (MQCFIN)**

Client authentication (parameter identifier: MQIACH\_SSL\_CLIENT\_AUTH).

The value can be:

#### **MQSCA\_REQUIRED**

Client authentication required

#### **MQSCA\_OPTIONAL**

Client authentication is optional.

Defines whether IBM MQ requires a certificate from the TLS client.

### **SSLPeerName (MQCFST)**

Peer name (parameter identifier: MQCACH\_SSL\_PEER\_NAME).

The length of the string is MQ\_SSL\_PEER\_NAME\_LENGTH. On z/OS, it is MQ\_SHORT\_PEER\_NAME\_LENGTH.

Specifies the filter to use to compare with the distinguished name of the certificate from the peer queue manager or client at the other end of the channel. (A distinguished name is the identifier of the TLS certificate.) If the distinguished name in the certificate received from the peer does not match the SSLPEER filter, the channel does not start.

### **Suspend (MQCFIN)**

Specifies whether the queue manager is suspended (parameter identifier: MQIACF\_SUSPEND).

The value can be:

#### **MQSUS\_NO**

The queue manager is not suspended from the cluster.

**MQSUS\_YES**

The queue manager is suspended from the cluster.

**TpName (MQCFST)**

Transaction program name (parameter identifier: MQCACH\_TP\_NAME).

The maximum length of the string is MQ\_TP\_NAME\_LENGTH.

**TranmissionQName (MQCFST)**

Transmission queue name (parameter identifier: MQCA\_XMIT\_Q\_NAME). The cluster transmission queue used by the queue manager.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**TransportType (MQCFIN)**

Transmission protocol type (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

The value can be:

**MQXPT\_LU62**

LU 6.2.

**MQXPT\_TCP**

TCP.

**MQXPT\_NETBIOS**

NetBIOS.

**MQXPT\_SPX**

SPX.

**MQXPT\_DECNET**

DECnet.

**UseDLQ (MQCFIN)**

Determines whether the dead-letter queue is used when publication messages cannot be delivered to their correct subscriber queue (parameter identifier: MQIA\_USE\_DEAD\_LETTER\_Q).

**UserIdentifier (MQCFST)**

Task user identifier (parameter identifier: MQCACH\_USER\_ID). This parameter is not available on z/OS.

The maximum length of the string is MQ\_USER\_ID\_LENGTH. However, only the first 10 characters are used.

**Version (MQCFST)**

The version of the IBM MQ installation that the cluster queue manager is associated with. (parameter identifier: MQCA\_VERSION). The version has the format VVRRMMFF:

VV: Version

RR: Release

MM: Maintenance level

FF: Fix level

**Multi** **MQCMD\_INQUIRE\_COMM\_INFO (Inquire Communication Information Object) on Multiplatforms**

The Inquire Communication Information Object (MQCMD\_INQUIRE\_COMM\_INFO) PCF command inquires about the attributes of existing IBM MQ communication information objects.

**Required parameters:**

*CommInfoName*

**Optional parameters:**

*CommInfoAttrs, IntegerFilterCommand, StringFilterCommand*

## Required parameters

### ComminfoName (MQCFST)

The name of the communication information definition about which information is to be returned (parameter identifier: MQCA\_COMM\_INFO\_NAME).

The communication information name is always returned regardless of the attributes requested.

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

## Optional parameters

### ComminfoAttrs (MQCFIL)

Comminfo attributes (parameter identifier: MQIACF\_COMM\_INFO\_ATTRS).

The attribute list might specify the following value on its own - default value if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes.

or a combination of the following:

#### **MQIA\_CODED\_CHAR\_SET\_ID**

CCSID for transmitted messages.

#### **MQIA\_COMM\_EVENT**

Comminfo event control.

#### **MQIA\_MCAST\_BRIDGE**

Multicast bridging.

#### **MQIA\_MONITOR\_INTERVAL**

Frequency of update for monitoring information.

#### **MQIACF\_ENCODING**

Encoding for transmitted messages.

#### **MQIACH\_MC\_HB\_INTERVAL**

Multicast heartbeat interval.

#### **MQIACH\_MSG\_HISTORY**

Amount of message history being kept.

#### **MQIACH\_MULTICAST\_PROPERTIES**

Multicast properties control.

#### **MQIACH\_NEW\_SUBSCRIBER\_HISTORY**

New subscriber history.

#### **MQIACH\_PORT**

Port Number.

#### **MQCA\_ALTERATION\_DATE**

The date on which the information was last altered.

#### **MQCA\_ALTERATION\_TIME**

The time at which the information was last altered.

#### **MQCA\_COMM\_INFO\_DESC**

Comminfo description.

#### **MQCA\_COMM\_INFO\_TYPE**

Comminfo type

#### **MQCACH\_GROUP\_ADDRESS**

Group Address.

### IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ComminfoAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the



command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter for *CommInfoType* (MQIA\_COMM\_INFO\_TYPE), you cannot also specify the **CommInfoType** parameter.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

#### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *CommInfoAttrs* except MQCA\_COMM\_INFO\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

### **Multi MQCMD\_INQUIRE\_COMM\_INFO (Inquire Communication Information Object) Response on Multiplatforms**

The response to the Inquire Communication Information Object (MQCMD\_INQUIRE\_COMM\_INFO) PCF command consists of the response header followed by the *CommInfoName* structure, and the requested combination of attribute parameter structures (where applicable).

If a generic communication information name was specified, one such message is generated for each object found.

#### **Always returned:**

*CommInfoName*

#### **Returned if requested:**

*AlterationDate, AlterationTime, Bridge, CCSID, CommEvent, Description, Encoding, GrpAddress, MonitorInterval, MulticastHeartbeat, MulticastPropControl, MsgHistory, NewSubHistory, PortNumber, Type*

### **Response data**

#### **AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date when the information was last altered, in the form yyyy-mm-dd.

#### **AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time when the information was last altered, in the form hh.mm.ss.

#### **Bridge (MQCFIN)**

Multicast Bridging (parameter identifier: MQIA\_MCAST\_BRIDGE).

Controls whether publications from applications not using Multicast are bridged to applications using multicast.

#### **CCSID (MQCFIN)**

CCSID that messages are transmitted in (parameter identifier: MQIA\_CODED\_CHAR\_SET\_ID).

The coded character set identifier that messages are transmitted in.

#### **CommEvent (MQCFIN)**

Event Control (parameter identifier: MQIA\_COMM\_EVENT).

Controls whether event messages are generated for multicast handles that are created using this COMMINFO object. The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**MQEVR\_EXCEPTION**

Reporting of events for message reliability below the reliability threshold enabled.

**CommInfoName (MQCFST)**

The name of the communication information definition (parameter identifier: MQCA\_COMM\_INFO\_NAME).

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

**Description (MQCFST)**

Description of the communication information definition (parameter identifier: MQCA\_COMM\_INFO\_DESC).

The maximum length of the string is MQ\_COMM\_INFO\_DESC\_LENGTH.

**Encoding (MQCFIN)**

Encoding that messages are transmitted in (parameter identifier: MQIACF\_ENCODING).

The encoding that messages are transmitted in. The value can be any of the following values:

**MQENC\_AS\_PUBLISHED**

Encoding taken from published message.

**MQENC\_NORMAL****MQENC\_REVERSED****MQENC\_S390****MQENC\_TNS****GrpAddress (MQCFST)**

The group IP address or DNS name (parameter identifier: MQCACH\_GROUP\_ADDRESS).

The maximum length of the string is MQ\_GROUP\_ADDRESS\_LENGTH.

**MonitorInterval (MQCFIN)**

Frequency of monitoring (parameter identifier: MQIA\_MONITOR\_INTERVAL).

How frequently, in seconds, monitoring information is updated and event messages are generated.

**MulticastHeartbeat (MQCFIN)**

Heartbeat Interval for multicast (parameter identifier: MQIACH\_MC\_HB\_INTERVAL).

The heartbeat interval, in milliseconds, for multicast transmitters.

**MulticastPropControl (MQCFIN)**

Multicast property control (parameter identifier: MQIACH\_MULTICAST\_PROPERTIES).

Control which MQMD properties and user properties flow with the message. The value can be any of the following values:

**MQMCP\_ALL**

All MQMD and user properties.

**MQMAP\_REPLY**

Properties related to replying to messages.

**MQMAP\_USER**

Only user properties.

**MQMAP\_NONE**

No MQMD or user properties.

**MQMAP\_COMPAT**

Properties are transmitted in a format compatible with previous Multicast clients.

**MsgHistory (MQCFIN)**

Message History (parameter identifier: MQIACH\_MSG\_HISTORY).

The amount of message history, in kilobytes, that is kept by the system to handle retransmissions in the case of NACKS.

### **NewSubHistory (MQCFIN)**

New Subscriber History (parameter identifier: MQIACH\_NEW\_SUBSCRIBER\_HISTORY).

Controls how much historical data a new subscriber receives. The value can be any of the following values:

#### **MQNSH\_NONE**

Only publications from the time of the subscription are sent.

#### **MQNSH\_ALL**

As much history as is known is retransmitted.

### **PortNumber (MQCFIN)**

Port Number (parameter identifier: MQIACH\_PORT).

The port number to transmit on.

### **Type (MQCFIN)**

The type of the communications information definition (parameter identifier: MQIA\_COMM\_INFO\_TYPE).

The value can be:

#### **MQCIT\_MULTICAST**

Multicast.

## ***MQCMD\_INQUIRE\_CONNECTION (Inquire Connection)***

The Inquire connection (MQCMD\_INQUIRE\_CONNECTION) PCF command inquires about the applications which are connected to the queue manager, the status of any transactions that those applications are running, and the objects which the application has open.

## **Required parameters**

### **ConnectionId (MQCFBS)**

Connection identifier (parameter identifier: MQBACF\_CONNECTION\_ID).

This parameter is the unique connection identifier associated with an application that is connected to the queue manager. Specify either this parameter **or** *GenericConnectionId*.

All connections are assigned a unique identifier by the queue manager regardless of how the connection is established.

If you need to specify a generic connection identifier, use the **GenericConnectionId** parameter instead.

The length of the string is MQ\_CONNECTION\_ID\_LENGTH.

### **GenericConnectionId (MQCFBS)**

Generic specification of a connection identifier (parameter identifier: MQBACF\_GENERIC\_CONNECTION\_ID).

Specify either this parameter **or** *ConnectionId*.

If you specify a byte string of zero length, or one which contains only null bytes, information about all connection identifiers is returned. This value is the only value permitted for *GenericConnectionId*.

The length of the string is MQ\_CONNECTION\_ID\_LENGTH.

## **Optional parameters**

### **ByteStringFilterCommand (MQCFBF)**

Byte string filter command descriptor. The parameter identifier must be MQBACF\_EXTERNAL\_UOW\_ID, MQBACF\_ORIGIN\_UOW\_ID, or MQBACF\_Q\_MGR\_UOW\_ID. Use this

parameter to restrict the output from the command by specifying a filter condition. See “MQCFBF - PCF byte string filter parameter” on page 1518 for information about using this filter condition.

If you specify a byte string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter, or a string filter using the **StringFilterCommand** parameter.

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_Q\_MGR\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

### **ConnectionAttrs (MQCFIL)**

Connection attributes (parameter identifier: MQIACF\_CONNECTION\_ATTRS).

The attribute list can specify the following value on its own - default value if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes of the selected *ConnInfoType*.

or, if you select a value of MQIACF\_CONN\_INFO\_CONN for *ConnInfoType*, a combination of the following:

#### **MQBACF\_CONNECTION\_ID**

Connection identifier.

#### **MQBACF\_CONN\_TAG**

Connection tag.

#### **MQBACF\_EXTERNAL\_UOW\_ID**

External unit of recovery identifier associated with the connection.

#### **MQBACF\_ORIGIN\_UOW\_ID**

Unit of recovery identifier assigned by the originator (valid on z/OS only).

#### **MQBACF\_Q\_MGR\_UOW\_ID**

Unit of recovery identifier assigned by the queue manager.

#### **MQCACF\_APPL\_TAG**

Name of an application that is connected to the queue manager.

#### **MQCACF\_ASID**

The 4-character address-space identifier of the application identified in MQCACF\_APPL\_TAG (valid on z/OS only).

#### **MQCACF\_ORIGIN\_NAME**

Originator of the unit of recovery (valid on z/OS only).

#### **MQCACF\_PSB\_NAME**

The 8-character name of the program specification block (PSB) associated with the running IMS transaction (valid on z/OS only).

**MQCACF\_PST\_ID**

The 4-character IMS program specification table (PST) region identifier for the connected IMS region (valid on z/OS only).

**MQCACF\_TASK\_NUMBER**

A 7-digit CICS task number (valid on z/OS only).

**MQCACF\_TRANSACTION\_ID**

A 4-character CICS transaction identifier (valid on z/OS only).

**MQCACF\_UOW\_LOG\_EXTENT\_NAME**

Name of the first extent required to recover the transaction. MQCACF\_UOW\_LOG\_EXTENT\_NAME is not valid on z/OS.

**MQCACF\_UOW\_LOG\_START\_DATE**

Date on which the transaction associated with the current connection first wrote to the log.

**MQCACF\_UOW\_LOG\_START\_TIME**

Time at which the transaction associated with the current connection first wrote to the log.

**MQCACF\_UOW\_START\_DATE**

Date on which the transaction associated with the current connection was started.

**MQCACF\_UOW\_START\_TIME**

Time at which the transaction associated with the current connection was started.

**MQCACF\_USER\_IDENTIFIER**

User identifier of the application that is connected to the queue manager.

**MQCACH\_CHANNEL\_NAME**

Name of the channel associated with the connected application.

**MQCACH\_CONNECTION\_NAME**

Connection name of the channel associated with the application.

**MQIA\_APPL\_TYPE**

Type of the application that is connected to the queue manager.

**MQIACF\_CONNECT\_OPTIONS**

Connect options currently in force for this application connection.

You cannot use the value MQCNO\_STANDARD\_BINDING as a filter value.

**MQIACF\_PROCESS\_ID**

Process identifier of the application that is currently connected to the queue manager.

This parameter is not valid on z/OS.

**MQIACF\_THREAD\_ID**

Thread identifier of the application that is currently connected to the queue manager.

This parameter is not valid on z/OS.

**MQIACF\_UOW\_STATE**

State of the unit of work.

**MQIACF\_UOW\_TYPE**

Type of external unit of recovery identifier as understood by the queue manager.

or, if you select a value of MQIACF\_CONN\_INFO\_HANDLE for *ConnInfoType*, a combination of the following:

**MQCACF\_OBJECT\_NAME**

Name of each object that the connection has open.

**MQCACH\_CONNECTION\_NAME**

Connection name of the channel associated with the application.

 **MQIA\_QSG\_DISP**

Disposition of the object (valid on z/OS only).

You cannot use MQIA\_QSG\_DISP as a parameter to filter on.

**MQIA\_READ\_AHEAD**

The read ahead connection status.

**MQIA\_UR\_DISP**

The unit of recovery disposition associated with the connection (valid on z/OS only).

**MQIACF\_HANDLE\_STATE**

Whether an API call is in progress.

**MQIACF\_OBJECT\_TYPE**

Type of each object that the connection has open.

**MQIACF\_OPEN\_OPTIONS**

Options used by the connection to open each object.

or, if you select a value of MQIACF\_CONN\_INFO\_ALL for *ConnInfoType*, any of the previous values.

**ConnInfoType (MQCFIN)**

Type of connection information to be returned (parameter identifier: MQIACF\_CONN\_INFO\_TYPE).

The value can be any of the following values:

**MQIACF\_CONN\_INFO\_CONN**

Connection information. On z/OS, MQIACF\_CONN\_INFO\_CONN includes threads which might be logically or actually disassociated from a connection, together with those threads that are in-doubt and for which external intervention is needed to resolve them. MQIACF\_CONN\_INFO\_CONN is the default value used if the parameter is not specified.

**MQIACF\_CONN\_INFO\_HANDLE**

Information pertaining only to those objects opened by the specified connection.

**MQIACF\_CONN\_INFO\_ALL**

Connection information and information about those objects that the connection has open.

You cannot use *ConnInfoType* as a parameter to filter on.

**IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ConnectionAttrs* except as noted and MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. You cannot use the value MQCNO\_STANDARD\_BINDING on the MQIACF\_CONNECT\_OPTIONS parameter with either the MQCFOP\_CONTAINS or MQCFOP\_EXCLUDES operator. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you filter on MQIACF\_CONNECT\_OPTIONS or MQIACF\_OPEN\_OPTIONS, in each case the filter value must have only 1 bit set.

If you specify an integer filter, you cannot also specify a byte string filter using the

**ByteStringFilterCommand** parameter or a string filter using the **StringFilterCommand** parameter.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ConnectionAttrs*. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify a byte string filter using the

**ByteStringFilterCommand** parameter or an integer filter using the **IntegerFilterCommand** parameter.

**URDisposition (MQCFIN)**

The unit of recovery disposition associated with the connection (parameter identifier: MQI\_UR\_DISP). This parameter is valid only on z/OS.

The value can be any of the following values:

**MQQSGD\_ALL**

Specifies that all connections must be returned.

**MQQSGD\_GROUP**

Specifies that only connections with a GROUP unit of recovery disposition must be returned.

**MQQSGD\_Q\_MGR**

Specifies that only connections with a QMGR unit of recovery disposition must be returned.

**Error code**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CONNECTION\_ID\_ERROR**

Connection identifier not valid.

**MQCMD\_INQUIRE\_CONNECTION (Inquire Connection) Response**

The response to the Inquire Connection (MQCMD\_INQUIRE\_CONNECTION) PCF command consists of the response header followed by the *ConnectionId* structure and a set of attribute parameter structures determined by the value of *ConnInfoType* in the Inquire command.

If the value of *ConnInfoType* was MQIACF\_CONN\_INFO\_ALL, there is one message for each connection found with MQIACF\_CONN\_INFO\_CONN, and *n* more messages per connection with MQIACF\_CONN\_INFO\_HANDLE (where *n* is the number of objects that the connection has open).

**Always returned:**

*ConnectionId, ConnInfoType*

**Always returned if ConnInfoType is MQIACF\_CONN\_INFO\_HANDLE:**

*ObjectName, ObjectType, z/OS QSGDisposition*

**Returned if requested and ConnInfoType is MQIACF\_CONN\_INFO\_CONN:**

*ApplDesc, ApplTag, ApplType, z/OS ASID, AsynchronousState, ChannelName, ClientIdentifier, ConnectionName, ConnectionOptions, Multi ConnectionTag, z/OS OriginName, z/OS OriginUOWId, z/OS ProcessId, PSBName, z/OS PSTId, QMgrUOWId, StartUOWLogExtent, TaskNumber, ThreadId, z/OS TransactionId, UOWIdentifier, UOWLogStartDate, UOWLogStartTime, UOWStartDate, UOWStartTime, UOWState, UOWType, z/OS URDisposition, UserId*

**Returned if requested and ConnInfoType is MQIACF\_CONN\_INFO\_HANDLE:**

*AsynchronousState, Destination, DestinationQueueManager, HandleState, OpenOptions, ReadAhead, SubscriptionID, SubscriptionName, TopicString*

**Response data****ApplDesc (MQCFST)**

Application description (parameter identifier: MQCACF\_APPL\_DESC).

The maximum length is MQ\_APPL\_DESC\_LENGTH.

**ApplTag (MQCFST)**

Application tag (parameter identifier: MQCACF\_APPL\_TAG).

The maximum length is MQ\_APPL\_TAG\_LENGTH.

**ApplType (MQCFIN)**

Application type (parameter identifier: MQIA\_APPL\_TYPE).

The value can be any of the following values:

**MQAT\_QMGR**

Queue manager process.

**MQAT\_CHANNEL\_INITIATOR**

Channel initiator.

**MQAT\_USER**

User application.

**MQAT\_BATCH**

Application using a batch connection (only on z/OS).

**MQAT\_RRS\_BATCH**

RRS-coordinated application using a batch connection (only on z/OS).

**MQAT\_CICS**

CICS transaction (only on z/OS).

**MQAT\_IMS**

IMS transaction (only on z/OS).

**MQAT\_SYSTEM\_EXTENSION**

Application performing an extension of function that is provided by the queue manager.

**z/OS ASID (MQCFST)**

Address space identifier (parameter identifier: MQCACF\_ASID).

The four character address-space identifier of the application identified by *AppLTag*. It distinguishes duplicate values of *AppLTag*.

This parameter is valid only on z/OS.

The length of the string is MQ\_ASID\_LENGTH.

**AsynchronousState (MQCFIN)**

The state of asynchronous consumption on this handle (parameter identifier: MQIACF\_ASYNC\_STATE).

The value can be:

**MQAS\_NONE**

If *ConnInfoType* is MQIACF\_CONN\_INFO\_CONN, an MQCTL call has not been issued against the handle. Asynchronous message consumption cannot currently proceed on this connection. If *ConnInfoType* is MQIACF\_CONN\_INFO\_HANDLE, an MQCB call has not been issued against this handle, so no asynchronous message consumption is configured on this handle.

**MQAS\_SUSPENDED**

The asynchronous consumption callback has been suspended so that asynchronous message consumption cannot currently proceed on this handle. This situation can be either because an MQCB or MQCTL call with *Operation* MQOP\_SUSPEND has been issued against this object handle by the application, or because it has been suspended by the system. If it has been suspended by the system, as part of the process of suspending asynchronous message consumption the callback function is called with the reason code that describes the problem resulting in suspension. This reason code is reported in the *Reason* field in the MQCBC structure passed to the callback. In order for asynchronous message consumption to proceed, the application must issue an MQCB or MQCTL call with *Operation* MQOP\_RESUME. This reason code can be returned if *ConnInfoType* is MQIACF\_CONN\_INFO\_CONN or MQIACF\_CONN\_INFO\_HANDLE.

**MQAS\_SUSPENDED\_TEMPORARY**

The asynchronous consumption callback has been temporarily suspended by the system so that asynchronous message consumption cannot currently proceed on this object handle. As part of the process of suspending asynchronous message consumption, the callback function is called with the reason code that describes the problem resulting in suspension. MQAS\_SUSPENDED\_TEMPORARY is reported in the *Reason* field in the MQCBC structure passed to the callback. The callback function is called again when asynchronous message consumption is resumed by the system when the temporary condition



has been resolved. MQAS\_SUSPENDED\_TEMPORARY is returned only if ConnInfoType is MQIACF\_CONN\_INFO\_HANDLE.

**MQAS\_STARTED**

An MQCTL call with *Operation* MQOP\_START has been issued against the connection handle so that asynchronous message consumption can proceed on this connection. MQAS\_STARTED is returned only if *ConnInfoType* is MQIACF\_CONN\_INFO\_CONN.

**MQAS\_START\_WAIT**

An MQCTL call with *Operation* MQOP\_START\_WAIT has been issued against the connection handle so that asynchronous message consumption can proceed on this connection. MQAS\_START\_WAIT is returned only if *ConnInfoType* is MQIACF\_CONN\_INFO\_CONN.

**MQAS\_STOPPED**

An MQCTL call with *Operation* MQOP\_STOP has been issued against the connection handle so that asynchronous message consumption cannot currently proceed on this connection. MQAS\_STOPPED is returned only if *ConnInfoType* is MQIACF\_CONN\_INFO\_CONN.

**MQAS\_ACTIVE**

An MQCB call has set up a function to call back to process messages asynchronously and the connection handle has been started so that asynchronous message consumption can proceed. MQAS\_ACTIVE is returned only if *ConnInfoType* is MQIACF\_CONN\_INFO\_HANDLE.

**MQAS\_INACTIVE**

An MQCB call has set up a function to call back to process messages asynchronously but the connection handle has not yet been started, or has been stopped or suspended, so that asynchronous message consumption cannot currently proceed. MQAS\_INACTIVE is returned only if *ConnInfoType* is MQIACF\_CONN\_INFO\_HANDLE.

**ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

**ClientId (MQCFST)**

Client identifier (parameter identifier: MQCACH\_CLIENT\_ID). The client identifier of the client that is using the connection. If there is no client identifier associated with the connection, this attribute is blank.

The maximum length of the string is MQ\_CLIENT\_ID\_LENGTH.

**ConnectionId (MQCFBS)**

Connection identifier (parameter identifier: MQBACF\_CONNECTION\_ID).

The length of the string is MQ\_CONNECTION\_ID\_LENGTH.

**ConnectionName (MQCFST)**

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH.

**ConnectionOptions (MQCFIL)**

Connect options currently in force for the connection (parameter identifier: MQIACF\_CONNECT\_OPTIONS).

**Multi ConnectionTag (MQCFBS)**

Connection tag (parameter identifier: MQBACF\_CONN\_TAG).

Identifies related connections, which collectively represent a single instance of an application. The length of the string is MQ\_CONN\_TAG\_LENGTH.

**ConnInfoType (MQCFIN)**

Type of information returned (parameter identifier: MQIACF\_CONN\_INFO\_TYPE).

The value can be any of the following values:

**MQIACF\_CONN\_INFO\_CONN**

Generic information for the specified connection.

**MQIACF\_CONN\_INFO\_HANDLE**

Information pertinent only to those objects opened by the specified connection.

**Destination (MQCFST)**

The destination queue for messages published to this subscription (parameter identifier MQCACF\_DESTINATION).

This parameter is relevant only for handles of subscriptions to topics.

**DestinationQueueManager (MQCFST)**

The destination queue manager for messages published to this subscription (parameter identifier MQCACF\_DESTINATION\_Q\_MGR).

This parameter is relevant only for handles of subscriptions to topics. If *Destination* is a queue hosted on the local queue manager, this parameter contains the local queue manager name. If *Destination* is a queue hosted on a remote queue manager, this parameter contains the name of the remote queue manager.

**HandleState (MQCFIN)**

State of the handle (parameter identifier: MQIACF\_HANDLE\_STATE).

The value can be any of the following values:

**MQHSTATE\_ACTIVE**

An API call from this connection is currently in progress for this object. If the object is a queue, this condition can arise when an MQGET WAIT call is in progress.

If there is an MQGET SIGNAL outstanding, then this situation does not mean, by itself, that the handle is active.

**MQHSTATE\_INACTIVE**

No API call from this connection is currently in progress for this object. If the object is a queue, this condition can arise when no MQGET WAIT call is in progress.

**ObjectName (MQCFST)**

Object name (parameter identifier: MQCACF\_OBJECT\_NAME).

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

**ObjectType (MQCFIN)**

Object type (parameter identifier: MQIACF\_OBJECT\_TYPE).

If this parameter is a handle of a subscription to a topic, the SUBID parameter identifies the subscription and can be used with the Inquire Subscription command to find all the details about the subscription.

The value can be any of the following values:

**MQOT\_Q**

Queue.

**MQOT\_NAMELIST**

Namelist.

**MQOT\_PROCESS**

Process.

**MQOT\_Q\_MGR**

Queue manager.

**MQOT\_CHANNEL**

Channel.

**MQOT\_AUTH\_INFO**

Authentication information object.

**MQOT\_TOPIC**

Topic.

### OpenOptions (MQCFIN)

Open options currently in force for the object for connection (parameter identifier: MQIACF\_OPEN\_OPTIONS).

This parameter is not relevant for a subscription. Use the SUBID field of the DISPLAY SUB command to find all the details about the subscription.

#### z/OS OriginName (MQCFST)

Origin name (parameter identifier: MQCACF\_ORIGIN\_NAME).

Identifies the originator of the unit of recovery, except where *AppType* is MQAT\_RRS\_BATCH when it is omitted.

This parameter is valid only on z/OS.

The length of the string is MQ\_ORIGIN\_NAME\_LENGTH.

#### z/OS OriginUOWId (MQCFBS)

Origin UOW identifier (parameter identifier: MQBACF\_ORIGIN\_UOW\_ID).

The unit of recovery identifier assigned by the originator. It is an 8-byte value.

This parameter is valid only on z/OS.

The length of the string is MQ\_UOW\_ID\_LENGTH.

#### z/OS ProcessId (MQCFIN)

Process identifier (parameter identifier: MQIACF\_PROCESS\_ID).

### PSBName (MQCFST)

Program specification block name (parameter identifier: MQCACF\_PSB\_NAME).

The 8-character name of the program specification block (PSB) associated with the running IMS transaction.

This parameter is valid only on z/OS.

The length of the string is MQ\_PSB\_NAME\_LENGTH.

#### z/OS PSTId (MQCFST)

Program specification table identifier (parameter identifier: MQCACF\_PST\_ID).

The 4-character IMS program specification table (PST) region identifier for the connected IMS region.

This parameter is valid only on z/OS.

The length of the string is MQ\_PST\_ID\_LENGTH.

### QMgrUOWId (MQCFBS)

Unit of recovery identifier assigned by the queue manager (parameter identifier: MQBACF\_Q\_MGR\_UOW\_ID).

z/OS On z/OS platforms, this parameter is returned as an 8-byte RBA.

Multi On Multiplatforms, this parameter is an 8-byte transaction identifier.

The maximum length of the string is MQ\_UOW\_ID\_LENGTH.

#### z/OS QSGDisposition (MQCFIN)

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid only on z/OS. The value can be any of the following values:

#### MQQSGD\_COPY

The object is defined as MQQSGD\_COPY.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED.

**ReadAhead (MQCFIN)**

The read ahead connection status (parameter identifier: MQIA\_READ\_AHEAD).

The value can be any of the following values:

**MQREADA\_NO**

Read ahead for browsing messages, or of non-persistent messages is not enabled for the object that the connection has open.

**MQREADA\_YES**

Read ahead for browsing messages, or of non-persistent messages is enabled for the object that the connection has open and is being used efficiently.

**MQREADA\_BACKLOG**

Read ahead for browsing messages, or of non-persistent messages is enabled for this object. Read ahead is not being used efficiently because the client has been sent many messages which are not being consumed.

**MQREADA\_INHIBITED**

Read ahead was requested by the application but has been inhibited because of incompatible options specified on the first MQGET call.

**StartUOWLogExtent (MQCFST)**

Name of the first extent needed to recover the transaction (parameter identifier: MQCACF\_UOW\_LOG\_EXTENT\_NAME).

The 8-character name of the program specification block (PSB) associated with the running IMS transaction.

This parameter is not valid on z/OS.

The maximum length of the string is MQ\_LOG\_EXTENT\_NAME\_LENGTH.

**SubscriptionID (MQCFBS)**

The internal, all time unique identifier of the subscription (parameter identifier MQBACF\_SUB\_ID).

This parameter is relevant only for handles of subscriptions to topics.

Not all subscriptions can be seen using Inquire Connection; only those subscriptions that have current handles open to the subscriptions can be seen. Use the Inquire Subscription command to see all subscriptions.

**SubscriptionName (MQCFST)**

The unique subscription name of the application associated with the handle (parameter identifier MQCACF\_SUB\_NAME).

This parameter is relevant only for handles of subscriptions to topics. Not all subscriptions have a subscription name.

**ThreadId (MQCFIN)**

Thread identifier (parameter identifier: MQIACF\_THREAD\_ID).

**TopicString (MQCFST)**

Resolved topic string (parameter identifier: MQCA\_TOPIC\_STRING).

This parameter is relevant for handles with an ObjectType of MQOT\_TOPIC. For any other object type, this parameter is blank.

**z/OS TransactionId (MQCFST)**

Transaction identifier (parameter identifier: MQCACF\_TRANSACTION\_ID).

The 4-character CICS transaction identifier.

This parameter is valid only on z/OS.

The maximum length of the string is MQ\_TRANSACTION\_ID\_LENGTH.

**UOWIdentifier (MQCFBS)**

External unit of recovery identifier associated with the connection (parameter identifier: MQBACF\_EXTERNAL UOW\_ID).

This parameter is the recovery identifier for the unit of recovery. The value of *UOWType* determines its format.

The maximum length of the byte string is MQ\_UOW\_ID\_LENGTH.

**UOWLogStartDate (MQCFST)**

Logged unit of work start date, in the form yyyy-mm-dd (parameter identifier: MQCACF\_UOW\_LOG\_START\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

**UOWLogStartTime (MQCFST)**

Logged unit of work start time, in the form hh.mm.ss (parameter identifier: MQCACF\_UOW\_LOG\_START\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**UOWStartDate (MQCFST)**

Unit of work creation date (parameter identifier: MQCACF\_UOW\_START\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

**UOWStartTime (MQCFST)**

Unit of work creation time (parameter identifier: MQCACF\_UOW\_START\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**UOWState (MQCFIN)**

State of the unit of work (parameter identifier: MQIACF\_UOW\_STATE).

The value can be any of the following values:

**MQUOWST\_NONE**

There is no unit of work.

**MQUOWST\_ACTIVE**

The unit of work is active.

**MQUOWST\_PREPARED**

The unit of work is in the process of being committed.

**MQUOWST\_UNRESOLVED**

The unit of work is in the second phase of a two-phase commit operation. IBM MQ holds resources on behalf of the unit of work and external intervention is required to resolve it. It might be as simple as starting the recovery coordinator (such as CICS, IMS, or RRS) or it might involve a more complex operation such as using the RESOLVE INDOUBT command. This value can occur only on z/OS.

**UOWType (MQCFIN)**

Type of external unit of recovery identifier as perceived by the queue manager (parameter identifier: MQIACF\_UOW\_TYPE).

The value can be any of the following values:

**MQUOWT\_Q\_MGR**

**MQUOWT\_CICS**

**MQUOWT\_RRS**

**MQUOWT\_IMS**

**MQUOWT\_XA**

## **z/OS** URDisposition (MQCFIN)

The unit of recovery disposition associated with the connection.

This parameter is valid only on z/OS.

The value can be:

### **MQQSGD\_GROUP**

This connection has a GROUP unit of recovery disposition.

### **MQQSGD\_Q\_MGR**

This connection has a QMGR unit of recovery disposition.

## **UserId (MQCFST)**

User identifier (parameter identifier: MQCACF\_USER\_IDENTIFIER).

The maximum length of the string is MQ\_MAX\_USER\_ID\_LENGTH.

## **Multi** MQCMD\_INQUIRE\_ENTITY\_AUTH (Inquire Entity Authority) on Multiplatforms

The Inquire Entity Authority (MQCMD\_INQUIRE\_ENTITY\_AUTH) PCF command inquires about authorizations of an entity to a specified object.

## **Required parameters**

### **EntityName (MQCFST)**

Entity name (parameter identifier: MQCACF\_ENTITY\_NAME).

Depending on the value of *EntityType*, this parameter is either:

- A principal name. This name is the name of a user for whom to retrieve authorizations to the specified object. On IBM MQ for Windows, the name of the principal can optionally include a domain name, specified in this format: `user@domain`.
- A group name. This name is the name of the user group on which to make the inquiry. You can specify one name only and this name must be the name of an existing user group.

**Windows** For IBM MQ for Windows only, the group name can optionally include a domain name, specified in the following formats:

```
GroupName@domain  
domain\GroupName
```

The maximum length of the string is MQ\_ENTITY\_NAME\_LENGTH.

### **EntityType (MQCFIN)**

Entity type (parameter identifier: MQIACF\_ENTITY\_TYPE).

The value can be:

#### **MQZAET\_GROUP**

The value of the **EntityName** parameter refers to a group name.

#### **MQZAET\_PRINCIPAL**

The value of the **EntityName** parameter refers to a principal name.

### **ObjectType (MQCFIN)**

The type of object referred to by the profile (parameter identifier: MQIACF\_OBJECT\_TYPE).

The value can be any of the following values:

#### **MQOT\_AUTH\_INFO**

Authentication information.

#### **MQOT\_CHANNEL**

Channel object.

**MQOT\_CLNTCONN\_CHANNEL**

Client-connection channel object.

**MQOT\_COMM\_INFO**

Communication information object

**MQOT\_LISTENER**

Listener object.

**MQOT\_NAMELIST**

Namelist.

**MQOT\_PROCESS**

Process.

**MQOT\_Q**

Queue, or queues, that match the object name parameter.

**MQOT\_Q\_MGR**

Queue manager.

**MQOT\_REMOTE\_Q\_MGR\_NAME**

Remote queue manager.

**MQOT\_SERVICE**

Service object.

**MQOT\_TOPIC**

Topic object.

**Options (MQCFIN)**

Options to control the set of authority records that is returned (parameter identifier: MQIACF\_AUTH\_OPTIONS).

This parameter is required and you must set it to the value MQAUTHOPT\_CUMULATIVE. It returns a set of authorities representing the cumulative authority that an entity has to a specified object.

If a user ID is a member of more than one group, this command displays the combined authorizations of all groups.

**Optional parameters****ObjectName (MQCFST)**

Object name (parameter identifier: MQCACF\_OBJECT\_NAME).

The name of the queue manager, queue, process definition, or generic profile on which to make the inquiry.

You must include a parameter if the *ObjectType* is not MQOT\_Q\_MGR. If you do not include this parameter, it is assumed that you are making an inquiry on the queue manager.

You cannot specify a generic object name although you can specify the name of a generic profile.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

**ProfileAttrs (MQCFIL)**

Profile attributes (parameter identifier: MQIACF\_AUTH\_PROFILE\_ATTRS).

The attribute list might specify the following value on its own - default value if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

or a combination of the following:

**MQCACF\_ENTITY\_NAME**

Entity name.

**MQIACF\_AUTHORIZATION\_LIST**

Authorization list.

**MQIACF\_ENTITY\_TYPE**

Entity type.

**MQIACF\_OBJECT\_TYPE**

Object type.

**ServiceComponent (MQCFST)**

Service component (parameter identifier: MQCACF\_SERVICE\_COMPONENT).

If installable authorization services are supported, this parameter specifies the name of the authorization service to which the authorizations apply.

If you omit this parameter, the authorization inquiry is made to the first installable component for the service.

The maximum length of the string is MQ\_SERVICE\_COMPONENT\_LENGTH.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRC\_UNKNOWN\_ENTITY**

User ID not authorized, or unknown.

**MQRCCF\_OBJECT\_TYPE\_MISSING**

Object type missing.

### **MQCMD\_INQUIRE\_ENTITY\_AUTH (Inquire Entity Authority) Response on Multiplatforms**

Each response to the Inquire Entity Authority (MQCMD\_INQUIRE\_ENTITY\_AUTH) PCF command consists of the response header followed by the *QMgrName*, *Options*, and *ObjectName* structures and the requested combination of attribute parameter structures.

**Always returned:**

*ObjectName*, *Options*, *QMgrName*

**Returned if requested:**

*AuthorizationList*, *EntityName*, *EntityType*, *ObjectType*

**Response data****AuthorizationList (MQCFIL)**

Authorization list(parameter identifier: MQIACF\_AUTHORIZATION\_LIST).

This list can contain zero or more authorization values. Each returned authorization value means that any user ID in the specified group or principal has the authority to perform the operation defined by that value. The value can be any of the following values:

**MQAUTH\_NONE**

The entity has authority set to 'none'.

**MQAUTH\_ALT\_USER\_AUTHORITY**

Specify an alternate user ID on an MQI call.

**MQAUTH\_BROWSE**

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

**MQAUTH\_CHANGE**

Change the attributes of the specified object, using the appropriate command set.



**MQAUTH\_CLEAR**

Clear a queue.

**MQAUTH\_CONNECT**

Connect the application to the specified queue manager by issuing an MQCONN call.

**MQAUTH\_CREATE**

Create objects of the specified type using the appropriate command set.

**MQAUTH\_DELETE**

Delete the specified object using the appropriate command set.

**MQAUTH\_DISPLAY**

Display the attributes of the specified object using the appropriate command set.

**MQAUTH\_INPUT**

Retrieve a message from a queue by issuing an MQGET call.

**MQAUTH\_INQUIRE**

Make an inquiry on a specific queue by issuing an MQINQ call.

**MQAUTH\_OUTPUT**

Put a message on a specific queue by issuing an MQPUT call.

**MQAUTH\_PASS\_ALL\_CONTEXT**

Pass all context.

**MQAUTH\_PASS\_IDENTITY\_CONTEXT**

Pass the identity context.

**MQAUTH\_SET**

Set attributes on a queue from the MQI by issuing an MQSET call.

**MQAUTH\_SET\_ALL\_CONTEXT**

Set all context on a queue.

**MQAUTH\_SET\_IDENTITY\_CONTEXT**

Set the identity context on a queue.

**MQAUTH\_CONTROL**

For listeners and services, start and stop the specified channel, listener, or service.

For channels, start, stop, and ping the specified channel.

For topics, define, alter, or delete subscriptions.

**MQAUTH\_CONTROL\_EXTENDED**

Reset or resolve the specified channel.

**MQAUTH\_PUBLISH**

Publish to the specified topic.

**MQAUTH\_SUBSCRIBE**

Subscribe to the specified topic.

**MQAUTH\_RESUME**

Resume a subscription to the specified topic.

**MQAUTH\_SYSTEM**

Use queue manager for internal system operations.

**MQAUTH\_ALL**

Use all operations applicable to the object.

**MQAUTH\_ALL\_ADMIN**

Use all administration operations applicable to the object.

**MQAUTH\_ALL\_MQI**

Use all MQI calls applicable to the object.

Use the *Count* field in the MQCFIL structure to determine how many values are returned.

**EntityName (MQCFST)**

Entity name (parameter identifier: MQCACF\_ENTITY\_NAME).

This parameter can either be a principal name or a group name.

The maximum length of the string is MQ\_ENTITY\_NAME\_LENGTH.

#### **EntityType (MQCFIN)**

Entity type (parameter identifier: MQIACF\_ENTITY\_TYPE).

The value can be:

##### **MQZAET\_GROUP**

The value of the **EntityName** parameter refers to a group name.

##### **MQZAET\_PRINCIPAL**

The value of the **EntityName** parameter refers to a principal name.

##### **MQZAET\_UNKNOWN**

On Windows, an authority record still exists from a previous queue manager which did not originally contain entity type information.

#### **ObjectName (MQCFST)**

Object name (parameter identifier: MQCACF\_OBJECT\_NAME).

The name of the queue manager, queue, process definition, or generic profile on which the inquiry is made.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

#### **ObjectType (MQCFIN)**

Object type (parameter identifier: MQIACF\_OBJECT\_TYPE).

The value can be:

##### **MQOT\_AUTH\_INFO**

Authentication information.

##### **MQOT\_CHANNEL**

Channel object.

##### **MQOT\_CLNTCONN\_CHANNEL**

Client-connection channel object.

##### **MQOT\_COMM\_INFO**

Communication information object

##### **MQOT\_LISTENER**

Listener object.

##### **MQOT\_NAMELIST**

Namelist.

##### **MQOT\_PROCESS**

Process.

##### **MQOT\_Q**

Queue, or queues, that match the object name parameter.

##### **MQOT\_Q\_MGR**

Queue manager.

##### **MQOT\_REMOTE\_Q\_MGR\_NAME**

Remote queue manager.

##### **MQOT\_SERVICE**

Service object.

#### **QMgrName (MQCFST)**

Name of the queue manager on which the Inquire command is issued (parameter identifier: MQCA\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

## z/OS MQCMD\_INQUIRE\_QSG (Inquire Group) on z/OS

The Inquire Group (MQCMD\_INQUIRE\_QSG) PCF command inquires about the queue sharing group to which the queue manager is connected.

**Note:** This command is supported only on z/OS when the queue manager is a member of a queue sharing group.

### Optional parameters

#### ObsoleteDB2Msgs (MQCFIN)

Whether to look for obsolete Db2 messages (parameter identifier: MQIACF\_OBSOLETE\_MSGS).

The value can be any of the following values:

#### MQOM\_NO

Obsolete messages in Db2 are not looked for. MQOM\_NO is the default value used if the parameter is not specified.

#### MQOM\_YES

Obsolete messages in Db2 are looked for and messages containing information about any found are returned.

## z/OS MQCMD\_INQUIRE\_QSG (Inquire Group) Response on z/OS

The response to the Inquire Group (MQCMD\_INQUIRE\_QSG) PCF command consists of the response header followed by the *QMgrName* structure and a number of other parameter structures. One such message is generated for each queue manager in the queue sharing group.

If there are any obsolete Db2 messages, and that information is requested, one message, identified by a value of MQCMDI\_DB2\_OBSOLETE\_MSGS in the **CommandInformation** parameter, is returned for each such message.

#### Always returned for the queue manager:

*CommandLevel, DB2ConnectStatus, DB2Name, QmgrCPF, QmgrName, QmgrNumber, QmgrStatus, QSGName*

#### Always returned for obsolete Db2 messages:

*CommandInformation, CFMsgIdentifier*

### Response data relating to the queue manager

#### CommandLevel (MQCFIN)

Command level supported by the queue manager (parameter identifier: MQIA\_COMMAND\_LEVEL).

The value can be any of the following values:

#### MQCMDL\_LEVEL\_800

Level 800 of system control commands.

#### MQCMDL\_LEVEL\_802

Level 802 of system control commands.

#### MQCMDL\_LEVEL\_900

Level 900 of system control commands.

#### MQCMDL\_LEVEL\_901

Level 901 of system control commands.

#### MQCMDL\_LEVEL\_902

Level 902 of system control commands.

#### MQCMDL\_LEVEL\_903

Level 903 of system control commands.

#### MQCMDL\_LEVEL\_904

Level 904 of system control commands.

**MQCMDL\_LEVEL\_905**

Level 905 of system control commands.

**MQCMDL\_LEVEL\_910**

Level 910 of system control commands.

**MQCMDL\_LEVEL\_911**

Level 911 of system control commands.

**MQCMDL\_LEVEL\_912**

Level 912 of system control commands.

**MQCMDL\_LEVEL\_913**

Level 913 of system control commands.

**MQCMDL\_LEVEL\_914**

Level 914 of system control commands.

**MQCMDL\_LEVEL\_915**

Level 915 of system control commands.

**MQCMDL\_LEVEL\_920**

Level 920 of system control commands.

**MQCMDL\_LEVEL\_921**

Level 921 of system control commands.

**MQCMDL\_LEVEL\_922**

Level 922 of system control commands.

**MQCMDL\_LEVEL\_923**

Level 923 of system control commands.

**MQCMDL\_LEVEL\_924**

Level 924 of system control commands.

**MQCMDL\_LEVEL\_925**

Level 925 of system control commands.

**MQCMDL\_LEVEL\_930**

Level 930 of system control commands.

**MQCMDL\_LEVEL\_931**

Level 931 of system control commands.

**MQCMDL\_LEVEL\_932**

Level 932 of system control commands.

**DB2ConnectStatus (MQCFIN)**

The current status of the connection to Db2 (parameter identifier: MQIACF\_DB2\_CONN\_STATUS).

The current status of the queue manager. The value can be any of the following values:

**MQQSGS\_ACTIVE**

The queue manager is running and is connected to Db2.

**MQQSGS\_INACTIVE**

The queue manager is not running and is not connected to Db2.

**MQQSGS\_FAILED**

The queue manager is running but not connected because Db2 has terminated abnormally.

**MQQSGS\_PENDING**

The queue manager is running but not connected because Db2 has terminated normally.

**MQQSGS\_UNKNOWN**

The status cannot be determined.

**DB2Name (MQCFST)**

The name of the Db2 subsystem or group to which the queue manager is to connect (parameter identifier: MQCACF\_DB2\_NAME).

The maximum length is MQ\_DB2\_NAME\_LENGTH.

**QMGrCPF (MQCFST)**

The command prefix of the queue manager (parameter identifier: MQCACF\_Q\_MGR\_CPF).

The maximum length is MQ\_Q\_MGR\_CPF\_LENGTH.

**QMGrName (MQCFST)**

Name of the queue manager (parameter identifier: MQCA\_Q\_MGR\_NAME).

The maximum length is MQ\_Q\_MGR\_NAME\_LENGTH.

**QmgrNumber (MQCFIN)**

The number, generated internally, of the queue manager in the group.(parameter identifier: MQIACF\_Q\_MGR\_NUMBER).

**QMGrStatus (MQCFIN)**

Recovery (parameter identifier: MQIACF\_Q\_MGR\_STATUS).

The current status of the queue manager. The value can be any of the following values:

**MQQSGS\_ACTIVE**

The queue manager is running.

**MQQSGS\_INACTIVE**

The queue manager is not running, having terminated normally.

**MQQSGS\_FAILED**

The queue manager is not running, having terminated abnormally.

**MQQSGS\_CREATED**

The queue manager has been defined to the group, but has not yet been started.

**MQQSGS\_UNKNOWN**

The status cannot be determined.

**QSGName (MQCFST)**

The name of the queue sharing group (parameter identifier: MQCA\_QSG\_NAME).

The maximum length is MQ\_QSG\_NAME\_LENGTH.

**Response data relating to obsolete Db2 messages****CFMsgIdentifier (MQCFBS)**

CF list entry identifier (parameter identifier: MQBACF\_CF\_LEID).

The maximum length is MQ\_CF\_LEID\_LENGTH.

**CommandInformation (MQCFIN)**

Command information (parameter identifier: MQIACF\_COMMAND\_INFO). This indicates whether queue managers in the group contain obsolete messages. The value is MQCMDI\_DB2\_OBSOLETE\_MSGS.

**Multi**

**MQCMD\_INQUIRE\_LISTENER (Inquire Channel Listener) on Multiplatforms**

The Inquire Channel Listener (MQCMD\_INQUIRE\_LISTENER) PCF command inquires about the attributes of existing IBM MQ listeners.

**Required parameters****ListenerName (MQCFST)**

Listener name (parameter identifier: MQCACH\_LISTENER\_NAME).

This parameter is the name of the listener with attributes that are required. Generic listener names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all listeners having names that start with the selected character string. An asterisk on its own matches all possible names.

The listener name is always returned regardless of the attributes requested.

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

## Optional parameters

### IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ListenerAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

### ListenerAttrs (MQCFIL)

Listener attributes (parameter identifier: MQIACF\_LISTENER\_ATTRS).

The attribute list might specify the following value on its own- default value if the parameter is not specified:

#### MQIACF\_ALL

All attributes.

or a combination of the following:

#### MQCA\_ALTERATION\_DATE

Date on which the definition was last altered.

#### MQCA\_ALTERATION\_TIME

Time at which the definition was last altered.

#### MQCACH\_IP\_ADDRESS

IP address for the listener.

#### MQCACH\_LISTENER\_DESC

Description of listener definition.

#### MQCACH\_LISTENER\_NAME

Name of listener definition.

#### MQCACH\_LOCAL\_NAME

NetBIOS local name that the listener uses. MQCACH\_LOCAL\_NAME is valid only on Windows.

#### MQCACH\_TP\_NAME

The LU 6.2 transaction program name. MQCACH\_TP\_NAME is valid only on Windows.

#### MQIACH\_ADAPTER

Adapter number on which NetBIOS listens. MQIACH\_ADAPTER is valid only on Windows.

#### MQIACH\_BACKLOG

Number of concurrent connection requests that the listener supports.

#### MQIACH\_COMMAND\_COUNT

Number of commands that the listener can use. MQIACH\_COMMAND\_COUNT is valid only on Windows.

#### MQIACH\_LISTENER\_CONTROL

Specifies when the queue manager starts and stops the listener.

#### MQIACH\_NAME\_COUNT

Number of names that the listener can use. MQIACH\_NAME\_COUNT is valid only on Windows.

#### MQIACH\_PORT

Port number.

#### MQIACH\_SESSION\_COUNT

Number of sessions that the listener can use. MQIACH\_SESSION\_COUNT is valid only on Windows.

#### MQIACH\_SOCKET

SPX socket on which to listen. MQIACH\_SOCKET is valid only on Windows.

### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ListenerAttrs* except MQCACH\_LISTENER\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 1530 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

### **TransportType (MQCFIN)**

Transport protocol type (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

If you specify this parameter, information is returned relating only to those listeners defined with the specified transport protocol type. If you specify an attribute in the *ListenerAttrs* list which is valid only for listeners of a different transport protocol type, it is ignored and no error is raised. If you specify this parameter, it must occur immediately after the **ListenerName** parameter.

If you do not specify this parameter, or if you specify it with a value of MQXPT\_ALL, information about all listeners is returned. Valid attributes in the *ListenerAttrs* list which are not applicable to the listener are ignored, and no error messages are issued. The value can be any of the following values:

#### **MQXPT\_ALL**

All transport types.

#### **MQXPT\_LU62**

SNA LU 6.2. MQXPT\_LU62 is valid only on Windows.

#### **MQXPT\_NETBIOS**

NetBIOS. MQXPT\_NETBIOS is valid only on Windows.

#### **MQXPT\_SPX**

SPX. MQXPT\_SPX is valid only on Windows.

#### **MQXPT\_TCP**

Transmission Control Protocol/Internet Protocol (TCP/IP).

## **Multi *MQCMD\_INQUIRE\_LISTENER (Inquire Channel Listener) Response on Multiplatforms***

The response to the Inquire Channel Listener (MQCMD\_INQUIRE\_LISTENER) PCF command consists of the response header followed by the *ListenerName* structure and the requested combination of attribute parameter structures.

If a generic listener name was specified, one such message is generated for each listener found.

### **Always returned:**

*ListenerName*

### **Returned if requested:**

*Adapter, AlterationDate, AlterationTime, Backlog, Commands, IPAddress, ListenerDesc, LocalName, NetbiosNames, Port, Sessions, Socket, StartMode, TPname, TransportType*

## **Response data**

### **AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date, in the form yyyy-mm-dd, on which the information was last altered.

### **AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time, in the form hh.mm.ss, at which the information was last altered.

### **Adapter (MQCFIN)**

Adapter number (parameter identifier: MQIACH\_ADAPTER).

The adapter number on which NetBIOS listens. This parameter is valid only on Windows.

**Backlog (MQCFIN)**

Backlog (parameter identifier: MQIACH\_BACKLOG).

The number of concurrent connection requests that the listener supports.

**Commands (MQCFIN)**

Adapter number (parameter identifier: MQIACH\_COMMAND\_COUNT).

The number of commands that the listener can use. This parameter is valid only on Windows.

**IPAddress (MQCFST)**

IP address (parameter identifier: MQCACH\_IP\_ADDRESS).

IP address for the listener specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric host name form.

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH

**ListenerDesc (MQCFST)**

Description of listener definition (parameter identifier: MQCACH\_LISTENER\_DESC).

The maximum length of the string is MQ\_LISTENER\_DESC\_LENGTH.

**ListenerName (MQCFST)**

Name of listener definition (parameter identifier: MQCACH\_LISTENER\_NAME).

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

**LocalName (MQCFST)**

NetBIOS local name (parameter identifier: MQCACH\_LOCAL\_NAME).

The NetBIOS local name that the listener uses. This parameter is valid only on Windows.

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH

**NetbiosNames (MQCFIN)**

NetBIOS names (parameter identifier: MQIACH\_NAME\_COUNT).

The number of names that the listener supports. This parameter is valid only on Windows.

**Port (MQCFIN)**

Port number (parameter identifier: MQIACH\_PORT).

The port number for TCP/IP. This parameter is valid only if the value of *TransportType* is MQXPT\_TCP.

**Sessions (MQCFIN)**

NetBIOS sessions (parameter identifier: MQIACH\_SESSION\_COUNT).

The number of sessions that the listener can use. This parameter is valid only on Windows.

**Socket (MQCFIN)**

SPX socket number (parameter identifier: MQIACH\_SOCKET).

The SPX socket on which to listen. This parameter is valid only if the value of *TransportType* is MQXPT\_SPX.

**StartMode (MQCFIN)**

Service mode (parameter identifier: MQIACH\_LISTENER\_CONTROL).

Specifies how the listener is to be started and stopped. The value can be any of the following values:

**MQSVC\_CONTROL\_MANUAL**

The listener is not to be started automatically or stopped automatically. It is to be controlled by user command. MQSVC\_CONTROL\_MANUAL is the default value.

**MQSVC\_CONTROL\_Q\_MGR**

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.



### **MQSVC\_CONTROL\_Q\_MGR\_START**

The listener is to be started at the same time as the queue manager is started, but is not request to stop when the queue manager is stopped.

### **TPName (MQCFST)**

Transaction program name (parameter identifier: MQCACH\_TP\_NAME).

The LU 6.2 transaction program name. This parameter is valid only on Windows.

The maximum length of the string is MQ\_TP\_NAME\_LENGTH

### **TransportType (MQCFIN)**

Transmission protocol (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

The value can be:

#### **MQXPT\_TCP**

TCP.

#### **MQXPT\_LU62**

LU 6.2. MQXPT\_LU62 is valid only on Windows.

#### **MQXPT\_NETBIOS**

NetBIOS. MQXPT\_NETBIOS is valid only on Windows.

#### **MQXPT\_SPX**

SPX. MQXPT\_SPX is valid only on Windows.

## **Multi *MQCMD\_INQUIRE\_LISTENER\_STATUS (Inquire Channel Listener Status) on Multiplatforms***

The Inquire Channel Listener Status (MQCMD\_INQUIRE\_LISTENER\_STATUS) PCF command inquires about the status of one or more IBM MQ listener instances.

You must specify the name of a listener for which you want to receive status information. You can specify a listener by using either a specific listener name or a generic listener name. By using a generic listener name, you can display either:

- Status information for all listener definitions, by using a single asterisk (\*), or
- Status information for one or more listeners that match the specified name.

### **Required parameters**

#### **ListenerName (MQCFST)**

Listener name (parameter identifier: MQCACH\_LISTENER\_NAME).

Generic listener names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all listeners having names that start with the selected character string. An asterisk on its own matches all possible names.

The listener name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

### **Optional parameters**

#### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ListenerStatusAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the

**StringFilterCommand** parameter.

### **ListenerStatusAttrs (MQCFIL)**

Listener status attributes (parameter identifier: MQIACF\_LISTENER\_STATUS\_ATTRS).

The attribute list can specify the following value on its own - default value used if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes.

or a combination of the following:

#### **MQCACH\_IP\_ADDRESS**

IP address of the listener.

#### **MQCACH\_LISTENER\_DESC**

Description of listener definition.

#### **MQCACH\_LISTENER\_NAME**

Name of listener definition.

#### **MQCACH\_LISTENER\_START\_DATE**

The date on which the listener was started.

#### **MQCACH\_LISTENER\_START\_TIME**

The time at which the listener was started.

#### **MQCACH\_LOCAL\_NAME**

NetBIOS local name that the listener uses. MQCACH\_LOCAL\_NAME is valid only on Windows.

#### **MQCACH\_TP\_NAME**

LU6.2 transaction program name. MQCACH\_TP\_NAME is valid only on Windows.

#### **MQIACF\_PROCESS\_ID**

Operating system process identifier associated with the listener.

#### **MQIACH\_ADAPTER**

Adapter number on which NetBIOS listens. MQIACH\_ADAPTER is valid only on Windows.

#### **MQIACH\_BACKLOG**

Number of concurrent connection requests that the listener supports.

#### **MQIACH\_COMMAND\_COUNT**

Number of commands that the listener can use. MQIACH\_COMMAND\_COUNT is valid only on Windows.

#### **MQIACH\_LISTENER\_CONTROL**

How the listener is to be started and stopped.

#### **MQIACH\_LISTENER\_STATUS**

Status of the listener.

#### **MQIACH\_NAME\_COUNT**

Number of names that the listener can use. MQIACH\_NAME\_COUNT is valid only on Windows.

#### **MQIACH\_PORT**

Port number for TCP/IP.

#### **MQIACH\_SESSION\_COUNT**

Number of sessions that the listener can use. MQIACH\_SESSION\_COUNT is valid only on Windows.

#### **MQIACH\_SOCKET**

SPX socket. MQIACH\_SOCKET is valid only on Windows.

#### **MQIACH\_XMIT\_PROTOCOL\_TYPE**

Transport type.

### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ListenerStatusAttrs* except MQCACH\_LISTENER\_NAME. Use this parameter to restrict the

output from the command by specifying a filter condition. See [“MQCFST - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

## Error code

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRCCF\_LSTR\_STATUS\_NOT\_FOUND**

Listener status not found.

## **Multi** **MQCMD\_INQUIRE\_LISTENER\_STATUS (Inquire Channel Listener Status)**

### Response on Multiplatforms

The response to the Inquire Channel Listener Status (MQCMD\_INQUIRE\_LISTENER\_STATUS) PCF command consists of the response header followed by the *ListenerName* structure and the requested combination of attribute parameter structures.

If a generic listener name was specified, one such message is generated for each listener found.

### Always returned:

*ListenerName*

### Returned if requested:

*Adapter, Backlog, ChannelCount, Commands, IPAddress, ListenerDesc, LocalName, NetbiosNames, Port, ProcessId, Sessions, Socket, StartDate, StartMode, StartTime, Status, TPname, TransportType*

## Response data

### Adapter (MQCFIN)

Adapter number (parameter identifier: MQIACH\_ADAPTER).

The adapter number on which NetBIOS listens.

### Backlog (MQCFIN)

Backlog (parameter identifier: MQIACH\_BACKLOG).

The number of concurrent connection requests that the listener supports.

### Commands (MQCFIN)

Adapter number (parameter identifier: MQIACH\_COMMAND\_COUNT).

The number of commands that the listener can use.

### IPAddress (MQCFST)

IP address (parameter identifier: MQCACH\_IP\_ADDRESS).

IP address for the listener specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric host name form.

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH

### ListenerDesc (MQCFST)

Description of listener definition (parameter identifier: MQCACH\_LISTENER\_DESC).

The maximum length of the string is MQ\_LISTENER\_DESC\_LENGTH.

### ListenerName (MQCFST)

Name of listener definition (parameter identifier: MQCACH\_LISTENER\_NAME).

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

**LocalName (MQCFST)**

NetBIOS local name (parameter identifier: MQCACH\_LOCAL\_NAME).

The NetBIOS local name that the listener uses.

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH

**NetbiosNames (MQCFIN)**

NetBIOS names (parameter identifier: MQIACH\_NAME\_COUNT).

The number of names that the listener supports.

**Port (MQCFIN)**

Port number (parameter identifier: MQIACH\_PORT).

The port number for TCP/IP.

**ProcessId (MQCFIN)**

Process identifier (parameter identifier: MQIACF\_PROCESS\_ID).

The operating system process identifier associated with the listener.

**Sessions (MQCFIN)**

NetBIOS sessions (parameter identifier: MQIACH\_SESSION\_COUNT).

The number of sessions that the listener can use.

**Socket (MQCFIN)**

SPX socket number (parameter identifier: MQIACH\_SOCKET).

The SPX socket on which the listener is to listen.

**StartDate (MQCFST)**

Start date (parameter identifier: MQCACH\_LISTENER\_START\_DATE).

The date, in the form yyyy-mm-dd, on which the listener was started.

The maximum length of the string is MQ\_DATE\_LENGTH

**StartMode (MQCFIN)**

Service mode (parameter identifier: MQIACH\_LISTENER\_CONTROL).

Specifies how the listener is to be started and stopped. The value can be any of the following values:

**MQSVC\_CONTROL\_MANUAL**

The listener is not to be started automatically or stopped automatically. It is to be controlled by user command. MQSVC\_CONTROL\_MANUAL is the default value.

**MQSVC\_CONTROL\_Q\_MGR**

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

**MQSVC\_CONTROL\_Q\_MGR\_START**

The listener is to be started at the same time as the queue manager is started, but is not request to stop when the queue manager is stopped.

**StartTime (MQCFST)**

Start date (parameter identifier: MQCACH\_LISTENER\_START\_TIME).

The time, in the form hh.mm.ss, at which the listener was started.

The maximum length of the string is MQ\_TIME\_LENGTH

**Status (MQCFIN)**

Listener status (parameter identifier: MQIACH\_LISTENER\_STATUS).

The status of the listener. The value can be any of the following values:

**MQSVC\_STATUS\_STARTING**

The listener is in the process of initializing.

**MQSVC\_STATUS\_RUNNING**

The listener is running.

**MQSVC\_STATUS\_STOPPING**

The listener is stopping.

**TPName (MQCFST)**

Transaction program name (parameter identifier: MQCACH\_TP\_NAME).

The LU 6.2 transaction program name.

The maximum length of the string is MQ\_TP\_NAME\_LENGTH

**TransportType (MQCFIN)**

Transmission protocol (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

The value can be:

**MQXPT\_TCP**

TCP.

**MQXPT\_LU62**

LU 6.2. MQXPT\_LU62 is valid only on Windows.

**MQXPT\_NETBIOS**

NetBIOS. MQXPT\_NETBIOS is valid only on Windows.

**MQXPT\_SPX**

SPX. MQXPT\_SPX is valid only on Windows.

## **MQCMD\_INQUIRE\_LOG (Inquire Log) on z/OS**

The Inquire Log (MQCMD\_INQUIRE\_LOG) PCF command returns log system parameters and information.

**Optional parameters****CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## **MQCMD\_INQUIRE\_LOG (Inquire Log) Response on z/OS**

The response to the Inquire Log (MQCMD\_INQUIRE\_LOG) PCF command consists of the response header followed by the *ParameterType* structure and the combination of attribute parameter structures determined by the value of *ParameterType*.

**Always returned:**

*ParameterType*. Specifies the type of archive information being returned. The value can be any of the following values:

**MQSYSP\_TYPE\_INITIAL**

The initial settings of the log parameters.

**MQSYSP\_TYPE\_SET**

The settings of the log parameters if they have been altered since their initial setting.

**MQSYSP\_TYPE\_LOG\_COPY**

Information relating to the active log copy.

**MQSYSP\_TYPE\_LOG\_STATUS**

Information relating to the status of the logs.

**Returned if *ParameterType* is MQSYSP\_TYPE\_INITIAL (one message is returned):**

*DeallocateInterval, DualArchive, DualActive, DualBSDS, InputBufferSize, LogArchive, LogCompression, MaxArchiveLog, MaxConcurrentOffloads, MaxReadTapeUnits, OutputBufferCount, OutputBufferSize, ZHyperWrite*

**Returned if *ParameterType* is MQSYSP\_TYPE\_SET and any value is set (one message is returned):**

*DeallocateInterval, DualArchive, DualActive, DualBSDS, InputBufferSize, LogArchive, MaxArchiveLog, MaxConcurrentOffloads, MaxReadTapeUnits, OutputBufferCount, OutputBufferSize, ZHyperWrite*

**Returned if *ParameterType* is MQSYSP\_TYPE\_LOG\_COPY (one message is returned for each log copy):**

*DataSetName, LogCopyNumber, LogUsed, ZHyperWrite, Encrypted*

**Returned if *ParameterType* is MQSYSP\_TYPE\_LOG\_STATUS (one message is returned):**

*FullLogs, LogCompression, LogRBA, LogSuspend, OffloadStatus, QMgrStartDate, QMgrStartRBA, QMgrStartTime, TotalLogs*

**Response data - log parameter information****DeallocateInterval (MQCFIN)**

Deallocation interval (parameter identifier: MQIACF\_SYSP\_DEALLOC\_INTERVAL).

Specifies the length of time, in minutes, that an allocated archive read tape unit is allowed to remain unused before it is deallocated. The value can be in the range zero through 1440. If it is zero, the tape unit is deallocated immediately. If it is 1440, the tape unit is never deallocated.

**DualActive (MQCFIN)**

Specifies whether dual logging is being used (parameter identifier: MQIACF\_SYSP\_DUAL\_ACTIVE).

The value can be any of the following values:

**MQSYSP\_YES**

Dual logging is being used.

**MQSYSP\_NO**

Dual logging is not being used.

**DualArchive (MQCFIN)**

Specifies whether dual archive logging is being used (parameter identifier: MQIACF\_SYSP\_DUAL\_ARCHIVE).

The value can be any of the following values:

**MQSYSP\_YES**

Dual archive logging is being used.

**MQSYSP\_NO**

Dual archive logging is not being used.

**DualBSDS (MQCFIN)**

Specifies whether dual BSDS is being used (parameter identifier: MQIACF\_SYSP\_DUAL\_BSDS).

The value can be any of the following values:

**MQSYSP\_YES**

Dual BSDS is being used.

**MQSYSP\_NO**

Dual BSDS is not being used.

**InputBufferSize (MQCFIN)**

Specifies the size of input buffer storage for active and archive log data sets (parameter identifier: MQIACF\_SYSP\_IN\_BUFFER\_SIZE).

**LogArchive (MQCFIN)**

Specifies whether archiving is on or off (parameter identifier: MQIACF\_SYSP\_ARCHIVE).

The value can be any of the following values:

**MQSYSP\_YES**

Archiving is on.

**MQSYSP\_NO**

Archiving is off.

**LogCompression (MQCFIN)**

Specifies which log compression parameter is used (parameter identifier: MQIACF\_LOG\_COMPRESSION).

The value can be any of the following values:

**MQCOMPRESS\_NONE**

No log compression is performed.

**MQCOMPRESS\_RLE**

Run-length encoding compression is performed.

**MQCOMPRESS\_ANY**

Enable the queue manager to select the compression algorithm that gives the greatest degree of log record compression. Using this option currently results in RLE compression.

**MaxArchiveLog (MQCFIN)**

Specifies the maximum number of archive log volumes that can be recorded in the BSDS (parameter identifier: MQIACF\_SYSP\_MAX\_ARCHIVE).

**MaxConcurrentOffloads (MQCFIN)**

Specifies the maximum number of concurrent log offload tasks (parameter identifier: MQIACF\_SYSP\_MAX\_CONC\_OFFLOADS).

**MaxReadTapeUnits (MQCFIN)**

The maximum number of dedicated tape units that can be set to read archive log tape volumes (parameter identifier: MQIACF\_SYSP\_MAX\_READ\_TAPES).

**OutputBufferCount (MQCFIN)**

Specifies the number of output buffers to be filled before they are written to the active log data sets (parameter identifier: MQIACF\_SYSP\_OUT\_BUFFER\_COUNT).

**OutputBufferSize (MQCFIN)**

Specifies the size of output buffer storage for active and archive log data sets (parameter identifier: MQIACF\_SYSP\_OUT\_BUFFER\_SIZE).

**ZHyperWrite (MQCFIN)**

For *MQSYSP\_TYPE\_INITIAL* and *MQSYSP\_TYPE\_SET*, shows whether writes to the active logs are made with zHyperWrite being enabled, if the logs are on zHyperWrite capable volumes (parameter identifier: MQIACF\_SYSP\_ZHYPERWRITE).

The value can be one of the following values:

**MQSYSP\_YES**

Writes are made using zHyperWrite, for active log datasets that are on zHyperWrite capable volumes.

**MQSYSP\_NO**

Writes are not made using zHyperWrite.

For *MQSYSP\_TYPE\_LOG\_COPY*, shows whether the log copy is on a zHyperWrite capable volume (parameter identifier: MQIACF\_SYSP\_ZHYPERWRITE).

The value can be one of the following values:

**MQSYSP\_YES**

The log data set is on a zHyperWrite capable volume.

**MQSYSP\_NO**

The log data set is not on a zHyperWrite capable volume.

**Response data - to log status information****DataSetName (MQCFST)**

The data set name of the active log data set (parameter identifier: MQCACF\_DATA\_SET\_NAME).

If the copy is not currently active, this parameter is returned as blank.

The maximum length of the string is MQ\_DATA\_DATA\_SET\_NAME\_LENGTH.

**Encrypted (MQCFIN)**

For MQSYSP\_TYPE\_LOG\_COPY, shows whether the log copy is an encrypted data set (parameter identifier: MQIACF\_DS\_ENCRYPTED)

The value can be one of the following values:

**MQSYSP\_YES**

The log data set is encrypted.

**MQSYSP\_NO**

The log data set is not encrypted.

**FullLogs (MQCFIN)**

The total number of full active log data sets that have not yet been archived (parameter identifier: MQIACF\_SYSP\_FULL\_LOGS).

**LogCompression (MQCFIN)**

Specifies the current log compression option (parameter identifier: MQIACF\_LOG\_COMPRESSION).

The value can be any of the following values:

**MQCOMPRESS\_NONE**

Log compression is not enabled.

**MQCOMPRESS\_RLE**

Run-length encoding log compression is enabled.

**MQCOMPRESS\_ANY**

Any compression algorithm supported by the queue manager is enabled.

**LogCopyNumber (MQCFIN)**

Copy number (parameter identifier: MQIACF\_SYSP\_LOG\_COPY).

**LogRBA (MQCFST)**

The RBA of the most recently written log record (parameter identifier: MQCACF\_SYSP\_LOG\_RBA).

The maximum length of the string is MQ\_RBA\_LENGTH.

**LogSuspend (MQCFIN)**

Specifies whether logging is suspended (parameter identifier: MQIACF\_SYSP\_LOG\_SUSPEND).

The value can be any of the following values:

**MQSYSP\_YES**

Logging is suspended.

**MQSYSP\_NO**

Logging is not suspended.

**LogUsed (MQCFIN)**

The percentage of the active log data set that has been used (parameter identifier: MQIACF\_SYSP\_LOG\_USED).

**OffloadStatus (MQCFIN)**

Specifies the status of the offload task (parameter identifier: MQIACF\_SYSP\_OFFLOAD\_STATUS).



The value can be any of the following values:

**MQSYSP\_STATUS\_ALLOCATING\_ARCHIVE**

The offload task is busy, allocating the archive data set. MQSYSP\_STATUS\_ALLOCATING\_ARCHIVE could indicate that a tape mount request is pending.

**MQSYSP\_STATUS\_COPYING\_BSDS**

The offload task is busy, copying the BSDS data set.

**MQSYSP\_STATUS\_COPYING\_LOG**

The offload task is busy, copying the active log data set.

**MQSYSP\_STATUS\_BUSY**

The offload task is busy with other processing.

**MQSYSP\_STATUS\_AVAILABLE**

The offload task is waiting for work.

**QMgrStartDate (MQCFST)**

The date on which the queue manager was started, in the form yyyy-mm-dd (parameter identifier: MQCACF\_SYSP\_Q\_MGR\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

**QMgrStartRBA (MQCFST)**

The RBA from which logging began when the queue manager was started (parameter identifier: MQCACF\_SYSP\_Q\_MGR\_RBA).

The maximum length of the string is MQ\_RBA\_LENGTH.

**QMgrStartTime (MQCFST)**

The time that the queue manager was started, in the form hh.mm.ss (parameter identifier: MQCACF\_SYSP\_Q\_MGR\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**TotalLogs (MQCFIN)**

The total number of active log data sets (parameter identifier: MQIACF\_SYSP\_TOTAL\_LOGS).

**MQCMD\_INQUIRE\_NAMELIST (Inquire Namelist)**

The Inquire Namelist (MQCMD\_INQUIRE\_NAMELIST) PCF command inquires about the attributes of existing IBM MQ namelists.

**Required parameters:**

*NamelistName*

**Optional parameters:**

 *CommandScope, IntegerFilterCommand, NamelistAttrs,*  *QSGDisposition, StringFilterCommand*

**Required parameters**

**NamelistName (MQCFST)**

Namelist name (parameter identifier: MQCA\_NAMELIST\_NAME).

This parameter is the name of the namelist with attributes that are required. Generic namelist names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all namelists having names that start with the selected character string. An asterisk on its own matches all possible names.

The namelist name is always returned regardless of the attributes requested.

The maximum length of the string is MQ\_NAMELIST\_NAME\_LENGTH.

## Optional parameters

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *NameListAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 1523 for information about using this filter condition.

If you specify an integer filter for *NameListType* (MQIA\_NAMELIST\_TYPE), you cannot also specify the **NameListType** parameter.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

### **NameListAttrs (MQCFIL)**

NameList attributes (parameter identifier: MQIACF\_NAMELIST\_ATTRS).

The attribute list might specify the following value on its own - default value if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes.

or a combination of the following:

#### **MQCA\_NAMELIST\_NAME**

Name of namelist object.

#### **MQCA\_NAMELIST\_DESC**

Namelist description.

#### **MQCA\_NAMES**

Names in the namelist.

#### **MQCA\_ALTERATION\_DATE**

The date on which the information was last altered.

#### **MQCA\_ALTERATION\_TIME**

The time at which the information was last altered.

#### **MQIA\_NAME\_COUNT**

Number of names in the namelist.

#### **MQIA\_NAMELIST\_TYPE**

Namelist type (valid only on z/OS)

### **NamelistType (MQCFIN)**

Namelist attributes (parameter identifier: MQIA\_NAMELIST\_TYPE). This parameter applies to z/OS only.

Specifies the type of names in the namelist. The value can be any of the following values:

#### **MQNT\_NONE**

The names are of no particular type.

#### **MQNT\_Q**

A namelist that holds a list of queue names.

#### **MQNT\_CLUSTER**

A namelist that is associated with clustering, containing a list of the cluster names.

#### **MQNT\_AUTH\_INFO**

The namelist is associated with TLS, and contains a list of authentication information object names.

z/OS

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

#### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

#### **MQQSGD\_PRIVATE**

The object is defined as either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *NamelistAttrs* except MQCA\_NAMELIST\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 1530 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

## **MQCMD\_INQUIRE\_NAMELIST (Inquire Namelist) Response**

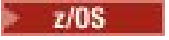
The response to the Inquire Namelist (MQCMD\_INQUIRE\_NAMELIST) PCF command consists of the response header followed by the *NamelistName* structure and the requested combination of attribute parameter structures.

If a generic namelist name was specified, one such message is generated for each namelist found.

### **Always returned:**

*NamelistName*,  *QSGDisposition*

### **Returned if requested:**

*AlterationDate*, *AlterationTime*, *NameCount*, *NamelistDesc*,  *NamelistType*,  
*Names*

## **Response data**

### **AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date when the information was last altered, in the form yyyy-mm-dd.

### **AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time when the information was last altered, in the form hh.mm.ss.

### **NameCount (MQCFIN)**

Number of names in the namelist (parameter identifier: MQIA\_NAME\_COUNT).

The number of names contained in the namelist.

### **NamelistDesc (MQCFST)**

Description of namelist definition (parameter identifier: MQCA\_NAMELIST\_DESC).

The maximum length of the string is MQ\_NAMELIST\_DESC\_LENGTH.

### **NamelistName (MQCFST)**

The name of the namelist definition (parameter identifier: MQCA\_NAMELIST\_NAME).

The maximum length of the string is MQ\_NAMELIST\_NAME\_LENGTH.

### **NamelistType (MQCFIN)**

Type of names in the namelist (parameter identifier: MQIA\_NAMELIST\_TYPE). This parameter applies to z/OS only.

Specifies the type of names in the namelist. The value can be any of the following values:

#### **MQNT\_NONE**

The names are of no particular type.

#### **MQNT\_Q**

A namelist that holds a list of queue names.

#### **MQNT\_CLUSTER**

A namelist that is associated with clustering, containing a list of the cluster names.

#### **MQNT\_AUTH\_INFO**

The namelist is associated with TLS, and contains a list of authentication information object names.

### **Names (MQCFSL)**

A list of the names contained in the namelist (parameter identifier: MQCA\_NAMES).

The number of names in the list is given by the *Count* field in the MQCFSL structure. The length of each name is given by the *StringLength* field in that structure. The maximum length of a name is MQ\_OBJECT\_NAME\_LENGTH.



## **QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter applies only to z/OS. The value can be any of the following values:

### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

## **MQCMD\_INQUIRE\_NAMELIST\_NAMES (Inquire Namelist Names)**

The Inquire Namelist Names (MQCMD\_INQUIRE\_NAMELIST\_NAMES) PCF command inquires for a list of namelist names that match the generic namelist name specified.

### **Required parameters**

#### **NamelistName (MQCFST)**

Name of namelist (parameter identifier: MQCA\_NAMELIST\_NAME).

Generic namelist names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

### **Optional parameters**



#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

#### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being processed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

#### **MQQSGD\_PRIVATE**

The object is defined with either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

### ***MQCMD\_INQUIRE\_NAMELIST\_NAMES (Inquire Namelist Names) Response***

The response to the Inquire Namelist Names (MQCMD\_INQUIRE\_NAMELIST\_NAMES) PCF command consists of the response header followed by a single parameter structure giving zero or more names that match the specified namelist name.

#### **z/OS**

Additionally, on z/OS only, the *QSGDispositions* structure (with the same number of entries as the *NamelistNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *NamelistNames* structure.

#### **Always returned:**

*NamelistNames*, **z/OS** *QSGDispositions*

#### **Returned if requested:**

None

### **Response data**

#### **NamelistNames (MQCFSL)**

List of namelist names (parameter identifier: MQCACF\_NAMELIST\_NAMES).

#### **z/OS**

#### **QSGDispositions (MQCFIL)**

List of queue sharing group dispositions (parameter identifier: MQIACF\_QSG\_DISPS). This parameter is valid only on z/OS. Possible values for fields in this structure are:

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

## **MQCMD\_INQUIRE\_PROCESS (Inquire Process)**

The Inquire Process (MQCMD\_INQUIRE\_PROCESS) PCF command inquires about the attributes of existing IBM MQ processes.

### **Required parameters**

#### **ProcessName (MQCFST)**

Process name (parameter identifier: MQCA\_PROCESS\_NAME).

Generic process names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all processes having names that start with the selected character string. An asterisk on its own matches all possible names.

The process name is always returned regardless of the attributes requested.

The maximum length of the string is MQ\_PROCESS\_NAME\_LENGTH.

### **Optional parameters**

#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

#### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ProcessAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 1523 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

#### **ProcessAttrs (MQCFIL)**

Process attributes (parameter identifier: MQIACF\_PROCESS\_ATTRS).

The attribute list might specify the following value on its own - default value used if the parameter is not specified:

##### **MQIACF\_ALL**

All attributes.

or a combination of the following:

##### **MQCA\_ALTERATION\_DATE**

The date at which the information was last altered.

##### **MQCA\_ALTERATION\_TIME**

The time at which the information was last altered.

**MQCA\_APPL\_ID**

Application identifier.

**MQCA\_ENV\_DATA**

Environment data.

**MQCA\_PROCESS\_DESC**

Description of process definition.

**MQCA\_PROCESS\_NAME**

Name of process definition.

**MQCA\_USER\_DATA**

User data.

**MQIA\_APPL\_TYPE**

Application type.

**z/OS** **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

**MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

**MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_PRIVATE**

The object is defined as either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ProcessAttrs* except MQCA\_PROCESS\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the

**IntegerFilterCommand** parameter.



## **MQCMD\_INQUIRE\_PROCESS (Inquire Process) Response**

The response to the Inquire Process (MQCMD\_INQUIRE\_PROCESS) PCF command consists of the response header followed by the *ProcessName* structure and the requested combination of attribute parameter structures.

If a generic process name was specified, one such message is generated for each process found.

### **Always returned:**

*ProcessName*,  *QSGDisposition*

### **Returned if requested:**

*AlterationDate*, *AlterationTime*, *ApplId*, *ApplType*, *EnvData*, *ProcessDesc*, *UserData*

## **Response data**

### **AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date when the information was last altered, in the form yyyy-mm-dd.

### **AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time when the information was last altered, in the form hh.mm.ss.

### **ApplId (MQCFST)**

Application identifier (parameter identifier: MQCA\_APPL\_ID).

The maximum length of the string is MQ\_PROCESS\_APPL\_ID\_LENGTH.

### **ApplType (MQCFIN)**

Application type (parameter identifier: MQIA\_APPL\_TYPE).

The value can be:

#### **MQAT\_AIX**

AIX application (same value as MQAT\_UNIX)

#### **MQAT\_CICS**

CICS transaction

#### **MQAT\_DOS**

DOS client application

#### **MQAT\_MVS**

z/OS application

#### **MQAT\_OS400**

IBM i application

#### **MQAT\_QMGR**

Queue manager

#### **MQAT\_UNIX**

UNIX application

#### **MQAT\_WINDOWS**

16-bit Windows application

#### **MQAT\_WINDOWS\_NT**

32-bit Windows application

#### **integer**

System-defined application type in the range zero through 65 535 or a user-defined application type in the range 65 536 through 999 999 999

### **EnvData (MQCFST)**

Environment data (parameter identifier: MQCA\_ENV\_DATA).

The maximum length of the string is MQ\_PROCESS\_ENV\_DATA\_LENGTH.

**ProcessDesc (MQCFST)**

Description of process definition (parameter identifier: MQCA\_PROCESS\_DESC).

The maximum length of the string is MQ\_PROCESS\_DESC\_LENGTH.

**ProcessName (MQCFST)**

The name of the process definition (parameter identifier: MQCA\_PROCESS\_NAME).

The maximum length of the string is MQ\_PROCESS\_NAME\_LENGTH.

 **QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be any of the following values:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**UserData (MQCFST)**

User data (parameter identifier: MQCA\_USER\_DATA).

The maximum length of the string is MQ\_PROCESS\_USER\_DATA\_LENGTH.

***MQCMD\_INQUIRE\_PROCESS\_NAMES (Inquire Process Names)***

The Inquire Process Names (MQCMD\_INQUIRE\_PROCESS\_NAMES) PCF command inquires for a list of process names that match the generic process name specified.

**Required parameters****ProcessName (MQCFST)**

Name of process-definition for queue (parameter identifier: MQCA\_PROCESS\_NAME).

Generic process names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

**Optional parameters****CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

#### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

#### **MQQSGD\_PRIVATE**

The object is defined with either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

### **MQCMD\_INQUIRE\_PROCESS\_NAMES (Inquire Process Names) Response**

The response to the Inquire Process Names (MQCMD\_INQUIRE\_PROCESS\_NAMES) PCF command consists of the response header followed by a single parameter structure giving zero or more names that match the specified process name.

Additionally, on z/OS only, a parameter structure *QSGDispositions* is returned. This parameter structure has the same number of entries as the *ProcessNames* structure.. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *ProcessNames* structure.

This response is not supported on Windows.

#### **Always returned:**

*ProcessNames, QSGDispositions*

#### **Returned if requested:**

None

### **Response data**

#### **ProcessNames (MQCFSL)**

List of process names (parameter identifier: MQCACF\_PROCESS\_NAMES).

#### **QSGDispositions (MQCFIL)**

List of queue sharing group dispositions (parameter identifier: MQIACF\_QSG\_DISPS). This parameter applies only to z/OS. Possible values for fields in this structure are:

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

## **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**Multi**

## **MQCMD\_INQUIRE\_PROT\_POLICY (inquire security policy) on**

### **Multiplatforms**

The Inquire Policy (MQCMD\_INQUIRE\_PROT\_POLICY) PCF command inquires about the policy, or policies, set on a queue.

### **Required parameters**

#### **policy-name (MQCFST)**

Policy name (parameter identifier: MQCA\_POLICY\_NAME).

This parameter is the name of the policy with attributes that are required. Generic policy names are not supported, however, an asterisk on its own can be used to return all policy objects.

The name of the policy, or policies (or part of the policy name or names) to inquire is the same as the name of the queue, or queues, that the policies control. The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

The policy name is always returned regardless of the attributes requested.

### **Optional parameters**

#### **PolicyAttrs (MQCFIL)**

Policy attributes (parameter identifier: MQIACF\_POLICY\_ATTRS).

The attribute list might specify the following value on its own- default value if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes.

or a combination of the following:

#### **MQCA\_POLICY\_NAME**

Name of the policy.

#### **MQIA\_SIGNATURE\_ALGORITHM**

The digital signature algorithm.

#### **MQIA\_ENCRYPTION\_ALGORITHM**

The encryption algorithm.

#### **MQCA\_SIGNER\_DN**

The distinguished name of an authorized signer, or signers.

#### **MQCA\_RECIPIENT\_DN**

The distinguished name of an intended recipient, or recipients.

#### **MQIA\_TOLERATE\_UNPROTECTED**

Whether the policy is enforced or unprotected messages tolerated.

#### **MQIA\_KEY\_REUSE\_COUNT**

The number of times that an encryption key can be re-used.

#### **MQIACF\_ACTION**

The action taken on the command with regards to signer and recipient parameters.

### **Expected behavior for inquiring a policy**

When inquiring a policy name, a policy object is always returned even if one does not exist. When a policy object does not exist, the policy object returned is a default policy object that specifies plain text protection, that is, no signing or encryption of message data.

To view policy objects that exist, the policy name should be set to '\*'. This returns all policy objects that exist.

### Related information

[Managing security policies in AMS](#)

## **Multi** MQCMD\_INQUIRE\_PROT\_POLICY (inquire security policy) Response on Multiplatforms

The response to the Inquire Policy (MQCMD\_INQUIRE\_PROT\_POLICY) PCF command consists of the response header followed by the *PolicyName* structure and the requested combination of attribute parameter structures.

If a generic security policy name was specified, one such message is generated for each policy found.

### Always returned:

*PolicyName*

The name of the policy, or policies (or part of the policy name or names) to inquire are the same as the name of the queue, or queues, that the policies control.

### Returned if requested:

*Action, EncAlg, Enforce and Tolerate, KeyReuse Recipient, Recipient, SignAlg, Signer*

## Response data

### Action (MQCFIL)

Action (parameter identifier: MQIACF\_ACTION).

The action taken on the command with regards to signer and recipient parameters.

### EncAlg (MQCFIL)

Encryption algorithm (parameter identifier: MQIA\_ENCRYPTION\_ALGORITHM).

The encryption algorithm specified.

### Enforce and Tolerate (MQCFST)

Indicates whether the security policy should be enforced or whether unprotected messages are tolerated (parameter identifier: MQIA\_TOLERATE\_UNPROTECTED).

### KeyReuse (MQCFIN)

Specifies the number of times that an encryption key can be re-used (parameter identifier MQIA\_KEY\_REUSE\_COUNT)

### Recipient (MQCFIL)

Specifies the distinguished name of the intended recipient (parameter identifier: MQCA\_RECIPIENT\_DN)

This parameter can be specified multiple times.

The maximum length of the string is MQ\_DISTINGUISHED\_NAME\_LENGTH.

### SignAlg (MQCFIL)

Specifies the digital signature algorithm (parameter identifier: MQIA\_SIGNATURE\_ALGORITHM).

### Signer (MQCFST)

Specifies the distinguished name of an authorized signer (parameter identifier: MQCA\_SIGNER\_DN)

This parameter can be specified multiple times.

The maximum length of the string is MQ\_DISTINGUISHED\_NAME\_LENGTH.

## ***MQCMD\_INQUIRE\_PUBSUB\_STATUS (Inquire Publish/Subscribe Status)***

The Inquire Pub/Sub Status (MQCMD\_INQUIRE\_PUBSUB\_STATUS) PCF command inquires about the status of publish/subscribe connections.

### **Optional parameters**

#### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

#### **blank (or omit the parameter altogether)**

The command is executed on the queue manager on which it was entered.

#### **a queue manager name**

The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

#### **an asterisk (\*)**

The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use CommandScope as a parameter to filter on.

#### **PubSubStatusAttrs (MQCFIL)**

Publish/subscribe status attributes (parameter identifier: MQIACF\_PUBSUB\_STATUS\_ATTRS).

The attribute list might specify the following value on its own - default value if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes.

or a combination of the following:

#### **MQIA\_SUB\_COUNT**

The total number of subscriptions against the local tree.

#### **MQIA\_TOPIC\_NODE\_COUNT**

The total number of topic nodes in the local tree.

#### **MQIACF\_PUBSUB\_STATUS**

Hierarchy status.

#### **MQIACF\_PS\_STATUS\_TYPE**

Hierarchy type.

#### **Type (MQCFIN)**

Type (parameter identifier: MQIACF\_PS\_STATUS\_TYPE).

The type can specify one of the following:

#### **MQPSST\_ALL**

Return status of both parent and child connections. MQPSST\_ALL is the default value if the parameter is not specified.

#### **MQPSST\_LOCAL**

Return local status information.

#### **MQPSST\_PARENT**

Return status of the parent connection.

## **MQPSST\_CHILD**

Return status of the child connections.

## **MQCMD\_INQUIRE\_PUBSUB\_STATUS (Inquire Publish/Subscribe Status) Response**

The response to the Inquire publish/subscribe Status (MQCMD\_INQUIRE\_PUBSUB\_STATUS) PCF command consists of the response header followed by the attribute structures.

A group of parameters is returned containing the following attributes: *Type*, *QueueManagerName*, *Status*, *SubCount*, and *TopicNodeCount*.

### **Always returned:**

*QueueManagerName*, *Status*, *Type*, *SubCount*, and *TopicNodeCount*.

### **Returned if requested:**

*None*

## **Response data**

### **QueueManagerName (MQCFST)**

Either the name of the local queue manager when TYPE is LOCAL, or the name of the hierarchically connected queue manager (parameter identifier: MQCA\_Q\_MGR\_NAME).

### **Type (MQCFIN)**

Type of status that is being returned (parameter identifier: MQIACF\_PS\_STATUS\_TYPE).

The value can be:

#### **MQPSST\_CHILD**

Publish/subscribe status for a child hierarchical connection.

#### **MQPSST\_LOCAL**

Publish/subscribe status for the local queue manager.

#### **MQPSST\_PARENT**

Publish/subscribe status for the parent hierarchical connection.

### **Status (MQCFIN)**

The status of the publish/subscribe engine or the hierarchical connection (parameter identifier: MQIACF\_PUBSUB\_STATUS).

When TYPE is LOCAL the following values can be returned:

#### **MQPS\_STATUS\_ACTIVE**

The publish/subscribe engine and the queued publish/subscribe interface are running. It is therefore possible to publish or subscribe using the application programming interface and the queues that are monitored by the queued publish/subscribe interface appropriately.

#### **MQPS\_STATUS\_COMPAT**

The publish/subscribe engine is running. It is therefore possible to publish or subscribe using the application programming interface. The queued publish/subscribe interface is not running. Therefore, any message that is put to the queues monitored by the queued publish/subscribe interface is not acted upon by IBM MQ.

#### **MQPS\_STATUS\_ERROR**

The publish/subscribe engine has failed. Check your error logs to determine the reason for the failure.

#### **MQPS\_STATUS\_INACTIVE**

The publish/subscribe engine and the queued publish/subscribe interface are not running. It is therefore not possible to publish or subscribe using the application programming interface. Any publish/subscribe messages that are put to the queues that are monitored by the queued publish/subscribe interface is not acted upon by IBM MQ.

If inactive and you want to start the publish/subscribe engine, on the Change Queue Manager command set PubSubMode to **MQPSM\_ENABLED**.

**MQPS\_STATUS\_STARTING**

The publish/subscribe engine is initializing and is not yet operational.

**MQPS\_STATUS\_STOPPING**

The publish/subscribe engine is stopping.

When TYPE is PARENT, the following values can be returned:

**MQPS\_STATUS\_ACTIVE**

The connection with the parent queue manager is active.

**MQPS\_STATUS\_ERROR**

This queue manager is unable to initialize a connection with the parent queue manager because of a configuration error.

A message is produced in the queue manager logs to indicate the specific error. If you receive error message AMQ5821 or on z/OS systems CSQT821E, possible causes include:

- Transmit queue is full
- Transmit queue put disabled

If you receive error message AMQ5814 or on z/OS systems CSQT814E, take the following actions:

- Check that the parent queue manager is correctly specified.
- Ensure that broker is able to resolve the queue manager name of the parent broker.

To resolve the queue manager name, at least one of the following resources must be configured:

- A transmission queue with the same name as the parent queue manager name.
- A queue manager alias definition with the same name as the parent queue manager name.
- A cluster with the parent queue manager a member of the same cluster as this queue manager.
- A cluster queue manager alias definition with the same name as the parent queue manager name.
- A default transmission queue.

After you have set up the configuration correctly, modify the parent queue manager name to blank. Then set with the parent queue manager name.

**MQPS\_STATUS\_REFUSED**

The connection has been refused by the parent queue manager.

This situation might be caused by the parent queue manager already having another child queue manager of the same name as this queue manager.

Alternatively, the parent queue manager has used the RESET QMGR TYPE(PUBSUB) CHILD command to remove this queue manager as one of its children.

**MQPS\_STATUS\_STARTING**

The queue manager is attempting to request that another queue manager is its parent.

If the parent status remains in starting status without progressing to active status, take the following actions:

- Check that the sender channel to parent queue manager is running
- Check that the receiver channel from parent queue manager is running

**MQPS\_STATUS\_STOPPING**

The queue manager is disconnecting from its parent.

If the parent status remains in stopping status, take the following actions:

- Check that the sender channel to parent queue manager is running
- Check that the receiver channel from parent queue manager is running

When TYPE is CHILD, the following values can be returned:



**MQPS\_STATUS\_ACTIVE**

The connection with the parent queue manager is active.

**MQPS\_STATUS\_ERROR**

This queue manager is unable to initialize a connection with the parent queue manager because of a configuration error.

A message is produced in the queue manager logs to indicate the specific error. If you receive error message AMQ5821 or on z/OS systems CSQT821E, possible causes include:

- Transmit queue is full
- Transmit queue put disabled

If you receive error message AMQ5814 or on z/OS systems CSQT814E, take the following actions:

- Check that the child queue manager is correctly specified.
- Ensure that broker is able to resolve the queue manager name of the child broker.

To resolve the queue manager name, at least one of the following resources must be configured:

- A transmission queue with the same name as the child queue manager name.
- A queue manager alias definition with the same name as the child queue manager name.
- A cluster with the child queue manager a member of the same cluster as this queue manager.
- A cluster queue manager alias definition with the same name as the child queue manager name.
- A default transmission queue.

After you have set up the configuration correctly, modify the child queue manager name to blank. Then set with the child queue manager name.

**MQPS\_STATUS\_STARTING**

The queue manager is attempting to request that another queue manager is its parent.

If the child status remains in starting status without progressing to active status, take the following actions:

- Check that the sender channel to child queue manager is running
- Check that the receiver channel from child queue manager is running

**MQPS\_STATUS\_STOPPING**

The queue manager is disconnecting from its parent.

If the child status remains in stopping status, take the following actions:

- Check that the sender channel to child queue manager is running
- Check that the receiver channel from child queue manager is running

**SubCount (MQCFIN)**

When *Type* is MQPSST\_LOCAL, the total number of subscriptions against the local tree is returned.

When *Type* is MQPSST\_CHILD or MQPSST\_PARENT, queue manager relations are not inquired and the value MQPSCT\_NONE is returned. (parameter identifier: MQIA\_SUB\_COUNT).

**TopicNodeCount (MQCFIN)**

When *Type* is MQPSST\_LOCAL, the total number of topic nodes in the local tree is returned. When

*Type* is MQPSST\_CHILD or MQPSST\_PARENT, queue manager relations are not inquired and the value MQPSCT\_NONE is returned. (parameter identifier: MQIA\_TOPIC\_NODE\_COUNT).

**MQCMD\_INQUIRE\_Q (Inquire Queue)**

Use the Inquire Queue (MQCMD\_INQUIRE\_Q) PCF command to query the attributes of IBM MQ queues.

**Required parameters****QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

Generic queue names are supported. A generic name is a character string followed by an asterisk \* ; for example ABC\*. It selects all queues having names that start with the selected character string. An asterisk on its own matches all possible names.

The queue name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

## Optional parameters

V 9.3.1

Multi

### CapExpiry (MQCFIN)

Capped expiry processing (parameter identifier MQIA\_CAP\_EXPIRY) which can be an integer value or can take the value of MQCEX\_NOLIMIT.

Specifies a lifetime limit for messages put using the object, expressed in 10ths of a second. A value of -1, displayed as NOLIMIT, has no effect on processing.

CapExpiry provides, or limits, the value in the MQMD [Expiry](#) field of any message put.

An application provided MQMD **Expiry** that is less than any resolved CapExpiry value is passed. This value is not replaced by that resolved CapExpiry value.

This process allows an IBM MQ administrator to limit the life of messages put by an application that overlooked (or was unable to provide, in the case of MQTT) message expiration criteria.

However, this option does not allow an administrator to override application behavior where the required lifetime of messages was under-estimated.

The new capped value for expiry is used during the put processing as if it had been provided by the application in the MQMD structure.

The *capped* value is evaluated for each put being performed, and so is sensitive to the resolution of the put operation. For example, in a cluster, where the put operation is performed with BIND NOT FIXED, messages might pick up different expiry values depending on the CapExpiry value set for the transmission queue used by the channel.

z/OS

### CFStructure (MQCFST)

CF structure (parameter identifier: MQCA\_CF\_STRUC\_NAME). Specifies the name of the CF structure. This parameter is valid only on z/OS.

This parameter specifies that eligible queues are limited to those having the specified *CFStructure* value. If this parameter is not specified, then all queues are eligible.

Generic CF structure names are supported. A generic name is a character string followed by an asterisk \* ; for example ABC\*. It selects all CF structures having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

### ClusterInfo (MQCFIN)

Cluster information (parameter identifier: MQIACF\_CLUSTER\_INFO).

This parameter requests that cluster information about these queues and other queues in the repository that match the selection criteria is displayed. The cluster information is displayed in addition to information about attributes of queues defined on this queue manager.

In this case, there might be multiple queues with the same name displayed. The cluster information is shown with a queue type of MQQT\_CLUSTER.

You can set this parameter to any integer value, the value used does not affect the response to the command.

The cluster information is obtained locally from the queue manager.

### ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

This parameter specifies that eligible queues are limited to those having the specified *ClusterName* value. If this parameter is not specified, then all queues are eligible.

Generic cluster names are supported. A generic name is a character string followed by an asterisk \* ; for example ABC\*. It selects all clusters having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

### **ClusterNameList (MQCFST)**

Cluster namelist (parameter identifier: MQCA\_CLUSTER\_NAMELIST).

This parameter specifies that eligible queues are limited to those having the specified *ClusterNameList* value. If this parameter is not specified, then all queues are eligible.

Generic cluster namelists are supported. A generic name is a character string followed by an asterisk \* ; for example ABC\*. It selects all cluster namelists having names that start with the selected character string. An asterisk on its own matches all possible names.

### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following values:

- Blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- A queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment. The command server must be enabled.
- An asterisk " \* ". The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *QAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter for *Qtype* or *PageSetID*, you cannot also specify the *Qtype* or *PageSetID* parameter.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

### **z/OS PageSetID (MQCFIN)**

Page set identifier (parameter identifier: MQIA\_PAGESET\_ID). This parameter applies to z/OS only.

This parameter specifies that eligible queues are limited to those having the specified *PageSetID* value. If this parameter is not specified, then all queues are eligible.

### **QAttrs (MQCFIL)**

Queue attributes (parameter identifier: MQIACF\_Q\_ATTRS).

The attribute list might specify the following value on its own. If the parameter is not specified, this value is the default:

#### **MQIACF\_ALL**

All attributes.

You can also specify a combination of the parameters in the following table:

<i>Table 210. Inquire Queue command, queue attributes</i>					
	<b>Local queue</b>	<b>Model queue</b>	<b>Alias queue</b>	<b>Remote queue</b>	<b>Cluster queue</b>
MQCA_ALTERATION_DATE The date on which the information was last altered	✓	✓	✓	✓	✓
MQCA_ALTERATION_TIME The time at which the information was last altered	✓	✓	✓	✓	✓
MQCA_BACKOUT_REQ_Q_NAME Excessive backout requeue name	✓	✓			
MQCA_BASE_NAME Name of queue that alias resolves to			✓		
MQCA_CF_STRUC_NAME Coupling facility structure name. This attribute is valid on z/OS only	✓	✓			
MQCA_CLUS_CHL_NAME The generic name of the cluster-sender channels that use this queue as a transmission queue.	✓	✓			
MQCA_CLUSTER_DATE Date when the definition became available to the local queue manager					✓
MQCA_CLUSTER_NAME Cluster name	✓		✓	✓	✓
MQCA_CLUSTER_NAMELIST Cluster namelist	✓		✓	✓	
MQCA_CLUSTER_Q_MGR_NAME Queue manager name that hosts the queue					✓
MQCA_CLUSTER_TIME Time when the definition became available to the local queue manager					✓
MQCA_CREATION_DATE Queue creation date	✓	✓			
MQCA_CREATION_TIME Queue creation time	✓	✓			

Table 210. Inquire Queue command, queue attributes (continued)




	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQCA_CUSTOM The custom attribute for new features	✓	✓	✓	✓	✓
MQCA_INITIATION_Q_NAME Initiation queue name	✓	✓			
MQCA_PROCESS_NAME Name of process definition	✓	✓			
MQCA_Q_DESC Queue description	✓	✓	✓	✓	✓
MQCA_Q_MGR_IDENTIFIER Internally generated queue manager name					✓
MQCA_Q_NAME Queue name	✓	✓	✓	✓	✓
MQCA_REMOTE_Q_MGR_NAME Name of remote queue manager				✓	
MQCA_REMOTE_Q_NAME Name of remote queue as known locally on the remote queue manager				✓	
  MQCA_STORAGE_CLASS Storage class. MQCA_STORAGE_CLASS is valid on z/OS only	✓	✓			
  MQCA_STREAM_QUEUE_NAME Name of streaming queue	✓	✓			
MQCA_TPIPE_NAME The <b>TPIPE</b> name used for communication with OTMA using the IBM MQ IMS bridge	✓				
MQCA_TRIGGER_DATA Trigger data	✓	✓			
MQCA_XMIT_Q_NAME Transmission queue name				✓	

Table 210. Inquire Queue command, queue attributes (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQIA_ACCOUNTING_Q Accounting data collection	✓	✓			
MQIA_BACKOUT_THRESHOLD Backout threshold	✓	✓			
MQIA_BASE_TYPE Type of object	✓	✓	✓	✓	✓
MQIA_CLUSTER_Q_TYPE Cluster queue type					✓
MQIA_CLWL_Q_PRIORITY Cluster workload queue priority	✓		✓	✓	✓
MQIA_CLWL_Q_RANK Cluster workload queue rank	✓		✓	✓	✓
MQIA_CLWL_USEQ Cluster workload use remote setting	✓				
MQIA_CURRENT_Q_DEPTH Number of messages on queue	✓				
MQIA_DEF_BIND Default binding	✓		✓	✓	✓
MQIA_DEF_INPUT_OPEN_OPTION Default open-for-input option	✓	✓			
MQIA_DEF_PERSISTENCE Default message persistence	✓	✓	✓	✓	✓
MQIA_DEF_PRIORITY Default message priority	✓	✓	✓	✓	✓
MQIA_DEF_PUT_RESPONSE_TYPE Default put response type	✓	✓	✓	✓	✓
MQIA_DEF_READ_AHEAD Default put response type	✓	✓	✓	✓	✓
MQIA_DEFINITION_TYPE Queue definition type	✓	✓			

Table 210. Inquire Queue command, queue attributes (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQIA_DIST_LISTS Distribution list support. MQIA_DIST_LISTS is not valid on z/OS	✓	✓			
MQIA_HARDEN_GET_BACKOUT Whether to harden backout count	✓	✓			
MQIA_INDEX_TYPE Index type. This attribute is valid on z/OS only.	✓	✓			
MQIA_INHIBIT_GET Whether get operations are allowed	✓	✓	✓		
MQIA_INHIBIT_PUT Whether put operations are allowed	✓	✓	✓	✓	✓
MQIA_MAX_MSG_LENGTH Maximum message length	✓	✓			
MQIA_MAX_Q_DEPTH Maximum number of messages allowed on queue	✓	✓			
MQIA_MEDIA_IMAGE_RECOVER_Q Whether a queue object is recoverable from a media image, if linear logging is being used.	✓	✓			
MQIA_MONITORING_Q Online monitoring data collection	✓	✓			
MQIA_MSG_DELIVERY_SEQUENCE Whether message priority is relevant	✓	✓			
MQIA_NPM_CLASS Level of reliability assigned to non-persistent messages that are put to the queue	✓	✓			
MQIA_OPEN_INPUT_COUNT Number of MQOPEN calls that have the queue open for input	✓				
MQIA_OPEN_OUTPUT_COUNT Number of MQOPEN calls that have the queue open for output	✓				

Table 210. Inquire Queue command, queue attributes (continued)


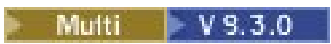
	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
 MQIA_PAGESET_ID Page set identifier	✓				
MQIA_PROPERTY_CONTROL Property control attribute	✓	✓	✓		
MQIA_Q_DEPTH_HIGH_EVENT Control attribute for queue depth high events. You cannot use MQIA_Q_DEPTH_HIGH_EVENT as a filter attribute.	✓	✓			
MQIA_Q_DEPTH_HIGH_LIMIT High limit for queue depth	✓	✓			
MQIA_Q_DEPTH_LOW_EVENT Control attribute for queue depth low events. You cannot use MQIA_Q_DEPTH_LOW_EVENT as a filter attribute.	✓	✓			
MQIA_Q_DEPTH_LOW_LIMIT Low limit for queue depth	✓	✓			
MQIA_Q_DEPTH_MAX_EVENT Control attribute for queue depth max events	✓	✓			
MQIA_Q_SERVICE_INTERVAL Limit for queue service interval	✓	✓			
MQIA_Q_SERVICE_INTERVAL_ EVENT Control attribute for queue service interval events	✓	✓			
MQIA_Q_TYPE Queue type	✓	✓	✓	✓	✓
MQIA_RETENTION_INTERVAL Queue retention interval	✓	✓			
MQIA_SCOPE Queue definition scope. MQIA_SCOPE is not valid on z/OS or IBM i	✓		✓	✓	



Table 210. Inquire Queue command, queue attributes (continued)					
	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQIA_SHAREABILITY Whether queue can be shared	✓	✓			
MQIA_STATISTICS_Q Statistics data collection. MQIA_STATISTICS_Q is valid only on Multiplatforms.	✓	✓			
 MQIA_STREAM_QUEUE_QOS Quality of service on streaming queue	✓	✓			
MQIA_TRIGGER_CONTROL Trigger control	✓	✓			
MQIA_TRIGGER_DEPTH Trigger depth	✓	✓			
MQIA_TRIGGER_MSG_PRIORITY Threshold message priority for triggers	✓	✓			
MQIA_TRIGGER_MTYPE Trigger type	✓	✓			
MQIA_USAGE Usage	✓	✓			

 **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP ). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned. The meaning of "the disposition of an object" is where the object is defined and how it behaves. The value can be any of the following values:

**MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. In a shared queue manager environment, if the command is run on the queue manager where it was issued, MQQSGD\_LIVE also returns information for objects defined with MQQSGD\_SHARED. MQQSGD\_LIVE is the default value if the parameter is not specified.

**MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

In a shared queue manager environment, if the command is run on the queue manager where it was issued, MQQSGD\_ALL also displays information for objects defined with MQQSGD\_GROUP or MQQSGD\_SHARED.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names, with different dispositions.

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_PRIVATE**

The object is defined with either MQQSGD\_Q\_MGR or MQQSGD\_COPY.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED. MQQSGD\_SHARED is permitted only in a shared queue environment.

You cannot use *QSGDisposition* as a parameter to filter on.

**QType (MQCFIN)**

Queue type (parameter identifier: MQIA\_Q\_TYPE).

If this parameter is present, eligible queues are limited to the specified type. Any attribute selector specified in the *QAttr*s list which is valid only for queues of a different type or types is ignored; no error is raised.

If this parameter is not present, or if MQQT\_ALL is specified, queues of all types are eligible. Each attribute specified must be a valid queue attribute selector. The attribute can apply to some of the queues returned. It does not have to apply to all the queues. Queue attribute selectors that are valid but not applicable to the queue are ignored, no error messages occur and no attribute is returned. The following lists contains the value of all valid queue attribute selectors:

**MQQT\_ALL**

All queue types.

**MQQT\_LOCAL**

Local queue.

**MQQT\_ALIAS**

Alias queue definition.

**MQQT\_REMOTE**


Local definition of a remote queue.

**MQQT\_CLUSTER**

Cluster queue.

**MQQT\_MODEL**

Model queue definition.

**Note:**  On [Multiplatforms](#), if this parameter is present, it must occur immediately after the **QName** parameter.

** StorageClass (MQCFST)**

Storage class (parameter identifier: MQCA\_STORAGE\_CLASS). Specifies the name of the storage class. This parameter is valid only on z/OS.

This parameter specifies that eligible queues are limited to those having the specified *StorageClass* value. If this parameter is not specified, then all queues are eligible.

Generic names are supported. A generic name is a character string followed by an asterisk \* ; for example ABC\*. It selects all storage classes having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.

### **StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *QAttrs* except *MQCA\_Q\_NAME*. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 1530 for information about using this filter condition.

If you specify a string filter for *ClusterName*, *ClusterNameList*, *StorageClass*, or *CFStructure*, you cannot also specify that as a parameter.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

### **Error codes**

This command might return the following error code in the response format header, in addition to the values shown in “Error codes applicable to all commands” on page 990.

#### **Reason (MQLONG)**

The value can be any of the following values:

##### **MQRCCF\_Q\_TYPE\_ERROR**

Queue type not valid.

### **MQCMD\_INQUIRE\_Q (Inquire Queue) Response**

The response to the Inquire Queue (MQCMD\_INQUIRE\_Q) PCF command consists of the response header followed by the *QName* structure. On z/OS only, response includes the *QSGDisposition* structure, and the requested combination of attribute parameter structures.

If a generic queue name was specified, or cluster queues requested, by setting either *MQQT\_CLUSTER* or *MQIACF\_CLUSTER\_INFO*, one message is generated for each queue found.

#### **Always returned:**

*QName*, *QSGDisposition*, *QType*

#### **Returned if requested:**

*AlterationDate*, *AlterationTime*, *BackoutRequeueName*, *BackoutThreshold*, *BaseQName*, *CapExpiry*, *CFStructure*, *ClusterChannelName*, *ClusterDate*, *ClusterName*, *ClusterNameList*, *ClusterQType*, *ClusterTime*, *CLWLQueuePriority*, *CLWLQueueRank*, *CLWLUseQ*, *CreationDate*, *CreationTime*, *CurrentQDepth*, *Custom*, *DefaultPutResponse*, *DefBind*, *DefinitionType*, *DefInputOpenOption*, *DefPersistence*, *DefPriority*, *DefReadAhead*, *DistLists*, *HardenGetBackout*, *Imgrcovq*, *IndexType*, *InhibitGet*, *InhibitPut*, *InitiationQName*, *MaxMsgLength*, *MaxQDepth*, *MsgDeliverySequence*, *NonPersistentMessageClass*, *OpenInputCount*, *OpenOutputCount*, *PageSetID*, *ProcessName*, *PropertyControl*, *QDepthHighEvent*, *QDepthHighLimit*, *QDepthLowEvent*, *QDepthLowLimit*, *QDepthMaxEvent*, *QDesc*, *QMgrIdentifier*, *QMgrName*, *QServiceInterval*, *QServiceIntervalEvent*, *QueueAccounting*, *QueueMonitoring*, *QueueStatistics*, *RemoteQMgrName*, *RemoteQName*, *RetentionInterval*, *Scope*, *Shareability*, *StorageClass*, *StreamQ*, *StreamQService*, *TpipeNames*, *TriggerControl*, *TriggerData*, *TriggerDepth*, *TriggerMsgPriority*, *TriggerType*, *Usage*, *XmitQName*

### **Response data**

#### **AlterationDate (MQCFST)**

Alteration date (parameter identifier: *MQCA\_ALTERATION\_DATE*).

The date when the information was last altered, in the form yyyy-mm-dd.

#### **AlterationTime (MQCFST)**

Alteration time (parameter identifier: *MQCA\_ALTERATION\_TIME*).

The time when the information was last altered, in the form hh.mm.ss.

**BackoutRequeueName (MQCFST)**

Excessive backout requeue name (parameter identifier: MQCA\_BACKOUT\_REQ\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**BackoutThreshold (MQCFIN)**

Backout threshold (parameter identifier: MQIA\_BACKOUT\_THRESHOLD).

**BaseQName (MQCFST)**

Queue name to which the alias resolves (parameter identifier: MQCA\_BASE\_Q\_NAME).

The name of a queue that is defined to the local queue manager.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**V 9.3.1****Multi****CapExpiry (MQCFIN)**

Capped message expiry processing (parameter identifier MQIA\_CAP\_EXPIRY).

Specifies a lifetime limit for messages put using the object, expressed in 10ths of a second.

**CFStructure (MQCFST)**

Coupling facility structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME). This parameter applies to z/OS only.

Specifies the name of the coupling facility structure where you want to store messages when you use shared queues.

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

**ClusterChannelName (MQCFST)**

Cluster-sender channel name (parameter identifier: MQCA\_CLUS\_CHL\_NAME).

ClusterChannelName is the generic name of the cluster-sender channels that use this queue as a transmission queue.

The maximum length of the channel name is: MQ\_CHANNEL\_NAME\_LENGTH.

**ClusterDate (MQCFST)**

Cluster date (parameter identifier: MQCA\_CLUSTER\_DATE).

The date on which the information became available to the local queue manager, in the form yyyy-mm-dd.

**ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

**ClusterNamelist (MQCFST)**

Cluster namelist (parameter identifier: MQCA\_CLUSTER\_NAMELIST).

**ClusterQType (MQCFIN)**

Cluster queue type (parameter identifier: MQIA\_CLUSTER\_Q\_TYPE).

The value can be:

**MQCQT\_LOCAL\_Q**

The cluster queue represents a local queue.

**MQCQT\_ALIAS\_Q**

The cluster queue represents an alias queue.

**MQCQT\_REMOTE\_Q**

The cluster queue represents a remote queue.

**MQCQT\_Q\_MGR\_ALIAS**

The cluster queue represents a queue manager alias.

**ClusterTime (MQCFST)**

Cluster time (parameter identifier: MQCA\_CLUSTER\_TIME).

The time at which the information became available to the local queue manager, in the form hh.mm.ss.

**CLWLQueuePriority (MQCFIN)**

Cluster workload queue priority (parameter identifier: MQIA\_CLWL\_Q\_PRIORITY).

Priority of the queue in cluster workload management. The value is in the range zero through 9, where zero is the lowest priority and 9 is the highest.

**CLWLQueueRank (MQCFIN)**

Cluster workload queue rank (parameter identifier: MQIA\_CLWL\_Q\_RANK).

Rank of the queue in cluster workload management. The value is in the range zero through 9, where zero is the lowest rank and 9 is the highest.

**CLWLUseQ (MQCFIN)**

Cluster workload queue rank (parameter identifier: MQIA\_CLWL\_USEQ).

The value can be:

**MQCLWL\_USEQ\_AS\_Q\_MGR**

Use the value of the **CLWLUseQ** parameter on the queue manager's definition.

**MQCLWL\_USEQ\_ANY**

Use remote and local queues.

**MQCLWL\_USEQ\_LOCAL**

Do not use remote queues.

**CreationDate (MQCFST)**

Queue creation date, in the form yyyy-mm-dd (parameter identifier: MQCA\_CREATION\_DATE).

The maximum length of the string is MQ\_CREATION\_DATE\_LENGTH.

**CreationTime (MQCFST)**

Creation time, in the form hh.mm.ss (parameter identifier: MQCA\_CREATION\_TIME).

The maximum length of the string is MQ\_CREATION\_TIME\_LENGTH.

**CurrentQDepth (MQCFIN)**

Current queue depth (parameter identifier: MQIA\_CURRENT\_Q\_DEPTH).

**Custom (MQCFST)**

Custom attribute for new features (parameter identifier: MQCA\_CUSTOM).

This attribute is reserved for the configuration of new features before separate attributes are named. It can contain the values of zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE).

This description is updated when features using this attribute are introduced.

**DefaultPutResponse (MQCFIN)**

Default put response type definition (parameter identifier: MQIA\_DEF\_PUT\_RESPONSE\_TYPE).

The parameter specifies the type of response to be used for put operations to the queue when an application specifies MQPMO\_RESPONSE\_AS\_Q\_DEF. The value can be any of the following values:

**MQPRT\_SYNC\_RESPONSE**

The put operation is issued synchronously, returning a response.

**MQPRT\_ASYNC\_RESPONSE**

The put operation is issued asynchronously, returning a subset of MQMD fields.

**DefBind (MQCFIN)**

Default binding (parameter identifier: MQIA\_DEF\_BIND).

The value can be:

**MQBND\_BIND\_ON\_OPEN**

Binding fixed by MQOPEN call.

**MQBND\_BIND\_NOT\_FIXED**

Binding not fixed.

**MQBND\_BIND\_ON\_GROUP**

Allows an application to request that a group of messages are all allocated to the same destination instance.

**DefinitionType (MQCFIN)**

Queue definition type (parameter identifier: MQIA\_DEFINITION\_TYPE).

The value can be:

**MQQDT\_PREDEFINED**

Predefined permanent queue.

**MQQDT\_PERMANENT\_DYNAMIC**

Dynamically defined permanent queue.

**MQQDT\_SHARED\_DYNAMIC**

Dynamically defined shared queue. This option is available on z/OS only.

**MQQDT\_TEMPORARY\_DYNAMIC**

Dynamically defined temporary queue.

**DefInputOpenOption (MQCFIN)**

Default input open option for defining whether queues can be shared (parameter identifier: MQIA\_DEF\_INPUT\_OPEN\_OPTION).

The value can be:

**MQOO\_INPUT\_EXCLUSIVE**

Open queue to get messages with exclusive access.

**MQOO\_INPUT\_SHARED**

Open queue to get messages with shared access.

**DefPersistence (MQCFIN)**

Default persistence (parameter identifier: MQIA\_DEF\_PERSISTENCE).

The value can be:

**MQPER\_PERSISTENT**

Message is persistent.

**MQPER\_NOT\_PERSISTENT**

Message is not persistent.

**DefPriority (MQCFIN)**

Default priority (parameter identifier: MQIA\_DEF\_PRIORITY).

**DefReadAhead (MQCFIN)**

Default read ahead (parameter identifier: MQIA\_DEF\_READ\_AHEAD).

Specifies the default read ahead behavior for non-persistent messages delivered to the client.

The value can be any of the following values:

**MQREADA\_NO**

Non-persistent messages are not sent ahead to the client before an application requests them. A maximum of one non-persistent message can be lost if the client ends abnormally.

**MQREADA\_YES**

Non-persistent messages are sent ahead to the client before an application requests them. Non-persistent messages can be lost if the client ends abnormally or if the client does not consume all the messages it is sent.

**MQREADA\_DISABLED**

Read ahead of non-persistent messages is not enabled for this queue. Messages are not sent ahead to the client regardless of whether read ahead is requested by the client application.

**Multi DistLists (MQCFIN)**

Distribution list support (parameter identifier: MQIA\_DIST\_LISTS).

The value can be:

**MQDL\_SUPPORTED**

Distribution lists supported.

**MQDL\_NOT\_SUPPORTED**

Distribution lists not supported.

This parameter is supported only on [Multiplatforms](#).

**HardenGetBackout (MQCFIN)**

Harden backout, or not: (parameter identifier: MQIA\_HARDEN\_GET\_BACKOUT).

The value can be:

**MQQA\_BACKOUT\_HARDENED**

Backout count remembered.

**MQQA\_BACKOUT\_NOT\_HARDENED**

Backout count may not be remembered.

**ImageRecoverQueue (MQCFST)**

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used (parameter identifier: MQIA\_MEDIA\_IMAGE\_RECOVER\_Q).

This parameter is not valid on z/OS. Possible values are:

**MQIMGRCOV\_YES**

These queue objects are recoverable.

**MQIMGRCOV\_NO**

Automatic media images, if enabled, are not written for these objects.

**MQIMGRCOV\_AS\_Q\_MGR**

If the **ImageRecoverQueue** attribute for the queue manager specifies MQIMGRCOV\_YES, these queue objects are recoverable.

If the **ImageRecoverQueue** attribute for the queue manager specifies MQIMGRCOV\_NO, the [“rcdmqimg \(record media image\)”](#) on page 139 and [“rcrmqobj \(re-create object\)”](#) on page 141 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

**IndexType (MQCFIN)**

Index type (parameter identifier: MQIA\_INDEX\_TYPE). This parameter applies to z/OS only.

Specifies the type of index maintained by the queue manager to expedite MQGET operations on the queue. The value can be any of the following values:

**MQIT\_NONE**

No index.

**MQIT\_MSG\_ID**

The queue is indexed using message identifiers.

**MQIT\_CORREL\_ID**

The queue is indexed using correlation identifiers.

**MQIT\_MSG\_TOKEN**

The queue is indexed using message tokens.

**MQIT\_GROUP\_ID**

The queue is indexed using group identifiers.

**InhibitGet (MQCFIN)**

Get operations are allowed or inhibited: (parameter identifier: MQIA\_INHIBIT\_GET).

The value can be:

**MQQA\_GET\_ALLOWED**

Get operations are allowed.

**MQQA\_GET\_INHIBITED**

Get operations are inhibited.

**InhibitPut (MQCFIN)**

Put operations are allowed or inhibited: (parameter identifier: MQIA\_INHIBIT\_PUT).

The value can be:

**MQQA\_PUT\_ALLOWED**

Put operations are allowed.

**MQQA\_PUT\_INHIBITED**

Put operations are inhibited.

**InitiationQName (MQCFST)**

Initiation queue name (parameter identifier: MQCA\_INITIATION\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**MaxMsgLength (MQCFIN)**

Maximum message length (parameter identifier: MQIA\_MAX\_MSG\_LENGTH).

**MaxQDepth (MQCFIN)**

Maximum queue depth (parameter identifier: MQIA\_MAX\_Q\_DEPTH).

**MsgDeliverySequence (MQCFIN)**

Messages ordered by priority or sequence: (parameter identifier: MQIA\_MSG\_DELIVERY\_SEQUENCE).

The value can be:

**MQMDS\_PRIORITY**

Messages are returned in priority order.

**MQMDS\_FIFO**

Messages are returned in FIFO order (first in, first out).

**NonPersistentMessageClass (MQCFIN)**

The level of reliability assigned to non-persistent messages that are put to the queue (parameter identifier: MQIA\_NPM\_CLASS).

Specifies the circumstances under which non-persistent messages put to the queue may be lost. The value can be any of the following values:

**MQNPM\_CLASS\_NORMAL**

Non-persistent messages are limited to the lifetime of the queue manager session. They are discarded in the event of a queue manager restart. MQNPM\_CLASS\_NORMAL is the default value.

**MQNPM\_CLASS\_HIGH**

The queue manager attempts to retain non-persistent messages for the lifetime of the queue. Non-persistent messages may still be lost in the event of a failure.

**OpenInputCount (MQCFIN)**

Number of MQOPEN calls that have the queue open for input (parameter identifier: MQIA\_OPEN\_INPUT\_COUNT).

**OpenOutputCount (MQCFIN)**

Number of MQOPEN calls that have the queue open for output (parameter identifier: MQIA\_OPEN\_OUTPUT\_COUNT).

**PageSetID (MQCFIN)**

Page set identifier (parameter identifier: MQIA\_PAGESET\_ID).

Specifies the identifier of the page set on which the queue resides.

This parameter applies to z/OS only when the queue is actively associated with a page set.

**ProcessName (MQCFST)**

Name of process definition for queue (parameter identifier: MQCA\_PROCESS\_NAME).

The maximum length of the string is MQ\_PROCESS\_NAME\_LENGTH.



**PropertyControl (MQCFIN)**

Property control attribute (parameter identifier MQIA\_PROPERTY\_CONTROL).

Specifies how message properties are handled for messages that are retrieved from queues using the MQGET call with the MQGMO\_PROPERTIES\_AS\_Q\_DEF option. The value can be any of the following values:

**MQPROP\_COMPATIBILITY**

If the message contains a property with a prefix of **mcd.**, **jms.**, **usr.** or **mqext.**, all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except properties contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

MQPROP\_COMPATIBILITY is the default value. It allows applications which expect JMS-related properties to be in an MQRFH2 header in the message data to continue to work unmodified.

**MQPROP\_NONE**

All properties of the message are removed from the message before the message is sent to the remote queue manager. Properties in the message descriptor (or extension) are not removed.

**MQPROP\_ALL**

All properties of the message are included with the message when it is sent to the remote queue manager. The properties are placed in one or more MQRFH2 headers in the message data. Properties in the message descriptor (or extension) are not placed in MQRFH2 headers.

**MQPROP\_FORCE\_MQRFH2**

Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

A valid message handle supplied in the `MsgHandle` field of the MQGMO structure on the MQGET call is ignored. Properties of the message are not accessible via the message handle.

This parameter is applicable to local, alias, and model queues.

**QDepthHighEvent (MQCFIN)**

Controls whether Queue Depth High events are generated (parameter identifier: MQIA\_Q\_DEPTH\_HIGH\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**QDepthHighLimit (MQCFIN)**

High limit for queue depth (parameter identifier: MQIA\_Q\_DEPTH\_HIGH\_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth High event.

**QDepthLowEvent (MQCFIN)**

Controls whether Queue Depth Low events are generated (parameter identifier: MQIA\_Q\_DEPTH\_LOW\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**QDepthLowLimit (MQCFIN)**

Low limit for queue depth (parameter identifier: MQIA\_Q\_DEPTH\_LOW\_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

**QDepthMaxEvent (MQCFIN)**

Controls whether Queue Full events are generated (parameter identifier: MQIA\_Q\_DEPTH\_MAX\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**QDesc (MQCFST)**

Queue description (parameter identifier: MQCA\_Q\_DESC).

The maximum length of the string is MQ\_Q\_DESC\_LENGTH.

**QMgrIdentifier (MQCFST)**

Queue manager identifier (parameter identifier: MQCA\_Q\_MGR\_IDENTIFIER).

The unique identifier of the queue manager.

**QMgrName (MQCFST)**

Name of local queue manager (parameter identifier: MQCA\_CLUSTER\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**QServiceInterval (MQCFIN)**

Target for queue service interval (parameter identifier: MQIA\_Q\_SERVICE\_INTERVAL).

The service interval used for comparison to generate Queue Service Interval High and Queue Service Interval OK events.

**QServiceIntervalEvent (MQCFIN)**

Controls whether Service Interval High or Service Interval OK events are generated (parameter identifier: MQIA\_Q\_SERVICE\_INTERVAL\_EVENT).

The value can be:

**MQQSIE\_HIGH**

Queue Service Interval High events enabled.

**MQQSIE\_OK**

Queue Service Interval OK events enabled.

**MQQSIE\_NONE**

No queue service interval events enabled.

**QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). *QSGDisposition* is valid only on z/OS. The value can be any of the following values:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED.

### **QType (MQCFIN)**

Queue type (parameter identifier: MQIA\_Q\_TYPE).

The value can be:

#### **MQQT\_ALIAS**

Alias queue definition.

#### **MQQT\_CLUSTER**

Cluster queue definition.

#### **MQQT\_LOCAL**

Local queue.

#### **MQQT\_REMOTE**

Local definition of a remote queue.

#### **MQQT\_MODEL**

Model queue definition.

### **QueueAccounting (MQCFIN)**

Controls the collection of accounting (thread-level and queue-level accounting) data (parameter identifier: MQIA\_ACCOUNTING\_Q).

The value can be:

#### **MQMON\_Q\_MGR**

The collection of accounting data for the queue is performed based upon the setting of the **QueueAccounting** parameter on the queue manager.

#### **MQMON\_OFF**

Do not collect accounting data for the queue.

#### **MQMON\_ON**

Collect accounting data for the queue.

### **QueueMonitoring (MQCFIN)**

Online monitoring data collection (parameter identifier: MQIA\_MONITORING\_Q).

The value can be:

#### **MQMON\_OFF**

Online monitoring data collection is turned off for this queue.

#### **MQMON\_Q\_MGR**

The value of the queue manager's **QueueMonitoring** parameter is inherited by the queue.

#### **MQMON\_LOW**

Online monitoring data collection is turned on, with a low rate of data collection, for this queue unless *QueueMonitoring* for the queue manager is MQMON\_NONE.

#### **MQMON\_MEDIUM**

Online monitoring data collection is turned on, with a moderate rate of data collection, for this queue unless *QueueMonitoring* for the queue manager is MQMON\_NONE.

#### **MQMON\_HIGH**

Online monitoring data collection is turned on, with a high rate of data collection, for this queue unless *QueueMonitoring* for the queue manager is MQMON\_NONE.

### **Multi QueueStatistics (MQCFIN)**

Controls the collection of statistics data (parameter identifier: MQIA\_STATISTICS\_Q).

The value can be:

#### **MQMON\_Q\_MGR**

The collection of statistics data for the queue is performed based upon the setting of the **QueueStatistics** parameter on the queue manager.

#### **MQMON\_OFF**

Do not collect statistics data for the queue.

**MQMON\_ON**

Collect statistics data for the queue unless *QueueStatistics* for the queue manager is MQMON\_NONE.

This parameter is supported only on [Multiplatforms](#).

**RemoteQMgrName (MQCFST)**

Name of remote queue manager (parameter identifier: MQCA\_REMOTE\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**RemoteQName (MQCFST)**

Name of remote queue as known locally on the remote queue manager (parameter identifier: MQCA\_REMOTE\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**RetentionInterval (MQCFIN)**

Retention interval (parameter identifier: MQIA\_RETENTION\_INTERVAL).

**Scope (MQCFIN)**

Scope of the queue definition (parameter identifier: MQIA\_SCOPE).

The value can be:

**MQSCO\_Q\_MGR**

Queue manager scope.

**MQSCO\_CELL**

Cell scope.

This parameter is not valid on IBM i or z/OS.

**Shareability (MQCFIN)**

The queue can be shared, or not: (parameter identifier: MQIA\_SHAREABILITY).

The value can be:

**MQQA\_SHAREABLE**

Queue is shareable.

**MQQA\_NOT\_SHAREABLE**

Queue is not shareable.

**StorageClass (MQCFST)**

Storage class (parameter identifier: MQCA\_STORAGE\_CLASS). This parameter applies to z/OS only.

Specifies the name of the storage class.

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.

▶ Multi ▶ V 9.3.0 **StreamQ (MQCFST)**

Name of the streaming queue (parameter identifier: MQCA\_STREAM\_QUEUE\_NAME)

▶ Multi ▶ V 9.3.0 **StreamQService (MQCFIN)**

Quality of service used when delivering messages to **Streamq** (parameter identifier: MQIA\_STREAM\_QUEUE\_QOS)

The value can be:

**MQST\_BEST\_EFFORT**

If the original message can be delivered, but the streamed message cannot, the original message is still delivered to its queue.

This is the default value.

**MQST\_MUST\_DUP**

The queue manager ensures that both the original message and the streamed message are successfully delivered to their queues.

If, for some reason, the streamed message cannot be delivered to its queue, the original message is not delivered to its queue either.

### **TpipeNames (MQCFSL)**

TPIPE names (parameter identifier: MQCA\_TPIPE\_NAME). This parameter applies to local queues on z/OS only.

Specifies the TPIPE names used for communication with OTMA via the IBM MQ IMS bridge, if the bridge is active.

The maximum length of the string is MQ\_TPIPE\_NAME\_LENGTH.

### **TriggerControl (MQCFIN)**

Trigger control (parameter identifier: MQIA\_TRIGGER\_CONTROL).

The value can be:

#### **MQTC\_OFF**

Trigger messages not required.

#### **MQTC\_ON**

Trigger messages required.

### **TriggerData (MQCFST)**

Trigger data (parameter identifier: MQCA\_TRIGGER\_DATA).

The maximum length of the string is MQ\_TRIGGER\_DATA\_LENGTH.

### **TriggerDepth (MQCFIN)**

Trigger depth (parameter identifier: MQIA\_TRIGGER\_DEPTH).

### **TriggerMsgPriority (MQCFIN)**

Threshold message priority for triggers (parameter identifier: MQIA\_TRIGGER\_MSG\_PRIORITY).

### **TriggerType (MQCFIN)**

Trigger type (parameter identifier: MQIA\_TRIGGER\_TYPE).

The value can be:

#### **MQTT\_NONE**

No trigger messages.

#### **MQTT\_FIRST**

Trigger message when queue depth goes from 0 to 1.

#### **MQTT EVERY**

Trigger message for every message.

#### **MQTT\_DEPTH**

Trigger message when depth threshold exceeded.

### **Usage (MQCFIN)**

Usage (parameter identifier: MQIA\_USAGE).

The value can be:

#### **MQUS\_NORMAL**

Normal usage.

#### **MQUS\_TRANSMISSION**

Transmission queue.

### **XmitQName (MQCFST)**

Transmission queue name (parameter identifier: MQCA\_XMIT\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

## **MQCMD\_INQUIRE\_Q\_MGR (Inquire Queue Manager)**

The Inquire Queue Manager (MQCMD\_INQUIRE\_Q\_MGR) PCF command inquires about the attributes of a queue manager.

### **Optional parameters**

#### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following values:

- Blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- A queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment. The command server must be enabled.
- An asterisk " \* ". The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

#### **QMGrAttrs (MQCFIL)**

Queue manager attributes (parameter identifier: MQIACF\_Q\_MGR\_ATTRS).

The attribute list might specify the following value on its own - default value used if the parameter is not specified:

##### **MQIACF\_ALL**

All attributes.

Or a combination of the following values:

##### **MQCA\_ALTERATION\_DATE**

Date at which the definition was last altered.

##### **MQCA\_ALTERATION\_TIME**

Time at which the definition was last altered.

##### **MQCA\_CERT\_LABEL**

Queue manager certificate label.

##### **MQCA\_CHANNEL\_AUTO\_DEF\_EXIT**

Automatic channel definition exit name. **MQCA\_CHANNEL\_AUTO\_DEF\_EXIT** is not valid on z/OS.

##### **MQCA\_CLUSTER\_WORKLOAD\_DATA**

Data passed to the cluster workload exit.

##### **MQCA\_CLUSTER\_WORKLOAD\_EXIT**

Name of the cluster workload exit.

##### **MQCA\_COMMAND\_INPUT\_Q\_NAME**

System command input queue name.

##### **MQCA\_CONN\_AUTH**

Name of the authentication information object that is used to provide the location of user ID and password authentication.

##### **MQCA\_CREATION\_DATE**

Queue manager creation date.

##### **MQCA\_CREATION\_TIME**

Queue manager creation time.

**MQCA\_CUSTOM**

The custom attribute for new features.

**MQCA\_DEAD\_LETTER\_Q\_NAME**

Name of dead-letter queue.

**MQCA\_DEF\_XMIT\_Q\_NAME**

Default transmission queue name.

**z/OS** **MQCA\_DNS\_GROUP**

The name of the group that the TCP listener handling inbound transmissions for the queue sharing group must join when using Workload Manager for Dynamic Domain Name Services support (DDNS). **MQCA\_DNS\_GROUP** is valid on z/OS only.

**z/OS** **MQCA\_IGQ\_USER\_ID**

Intra-group queuing user identifier. This parameter is valid on z/OS only.

**Multi** **V 9.3.0** **V 9.3.0** **MQCA\_INITIAL\_KEY**

The initial key for the password protection system.

**z/OS** **MQCA\_LU\_GROUP\_NAME**

Generic LU name for the LU 6.2 listener. **MQCA\_LU\_GROUP\_NAME** is valid on z/OS only.

**z/OS** **MQCA\_LU\_NAME**

LU name to use for outbound LU 6.2 transmissions. **MQCA\_LU\_NAME** is valid on z/OS only.

**z/OS** **MQCA\_LU62\_ARM\_SUFFIX**

APPCPM suffix. **MQCA\_LU62\_ARM\_SUFFIX** is valid on z/OS only.

**MQCA\_PARENT**

The name of the hierarchically connected queue manager that is nominated as the parent of this queue manager.

**MQCA\_Q\_MGR\_DESC**

Queue manager description.

**MQCA\_Q\_MGR\_IDENTIFIER**

Internally generated unique queue manager name.

**MQCA\_Q\_MGR\_NAME**

Name of local queue manager.

**z/OS** **MQCA\_QSG\_CERT\_LABEL**

Queue sharing group certificate label. This parameter attribute is valid on z/OS only.

**z/OS** **MQCA\_QSG\_NAME**

Queue sharing group name. This parameter attribute is valid on z/OS only.

**MQCA\_REPOSITORY\_NAME**

Cluster name for the queue manager repository.

**MQCA\_REPOSITORY\_NAMELIST**

Name of the list of clusters for which the queue manager is providing a repository manager service.

**MQCA\_SSL\_CRL\_NAMELIST**

TLS certificate revocation location namelist.

**ALW** **MQCA\_SSL\_CRYPTO\_HARDWARE**

Parameters to configure the TLS cryptographic hardware. This parameter is supported only on AIX, Linux, and Windows.

**V 9.3.0** **ALW** **V 9.3.0** **MQCA\_SSL\_KEY\_REPO\_PASSWORD**

The password to access the TLS key repository.

**MQCA\_SSL\_KEY\_REPOSITORY**

Location and name of the TLS key repository.

**z/OS** **MQCA\_TCP\_NAME**

Name of the TCP/IP system that you are using. **MQCA\_TCP\_NAME** is valid on z/OS only.

**MQCA\_VERSION**

The version of the IBM MQ installation, the queue manager is associated with. The version has the format *VVRRMMFF*:

*VV*: Version

*RR*: Release

*MM*: Maintenance level

*FF*: Fix level

**ALW** **MQIA\_ACCOUNTING\_CONN\_OVERRIDE**

Specifies whether the settings of the **MQIAccounting** and **QueueAccounting** queue manager parameters can be overridden. **MQIA\_ACCOUNTING\_CONN\_OVERRIDE** is valid only on AIX, Linux, and Windows.

**ALW** **MQIA\_ACCOUNTING\_INTERVAL**

Intermediate accounting data collection interval. **MQIA\_ACCOUNTING\_INTERVAL** is valid only on AIX, Linux, and Windows.

**ALW** **MQIA\_ACCOUNTING\_MQI**

Specifies whether accounting information is to be collected for MQI data. **MQIA\_ACCOUNTING\_MQI** is valid only on AIX, Linux, and Windows.

**MQIA\_ACCOUNTING\_Q**

Accounting data collection for queues.

**z/OS** **MQIA\_ACTIVE\_CHANNELS**

Maximum number of channels that can be active at any time. **MQIA\_ACTIVE\_CHANNELS** is valid on z/OS only.

**MQIA\_ACTIVITY\_CONN\_OVERRIDE**

Specifies whether the value of application activity trace can be overridden.

**MQIA\_ACTIVITY\_RECORDING**

Specifies whether activity reports can be generated.

**MQIA\_ACTIVITY\_TRACE**

Specifies whether application activity trace reports can be generated.

**z/OS** **MQIA\_ADOPTNEWMCA\_CHECK**

Elements checked to determine whether an MCA must be adopted when a new inbound channel is detected with the same name as an MCA that is already active. **MQIA\_ADOPTNEWMCA\_CHECK** is valid on z/OS only.

**z/OS** **MQIA\_ADOPTNEWMCA\_TYPE**

Specifies whether an orphaned instance of an MCA must be restarted automatically when a new inbound channel request matching the **AdoptNewMCACheck** parameter is detected. **MQIA\_ADOPTNEWMCA\_TYPE** is valid on z/OS only.

**MQ Adv.** **MQIA\_ADVANCED\_CAPABILITY**

Specifies whether IBM MQ Advanced extended capabilities are available for a queue manager.

**ALW** **MQIA\_AMQP\_CAPABILITY**

Specifies whether AMQP capabilities are available for a queue manager.

**MQIA\_AUTHORITY\_EVENT**

Control attribute for authority events.

**z/OS** **MQIA\_BRIDGE\_EVENT**

Control attribute for IMS bridge events. **MQIA\_BRIDGE\_EVENT** is valid only on z/OS.



**ALW** **MQIA\_CERT\_VAL\_POLICY**

Specifies which TLS certificate validation policy is used to validate digital certificates received from remote partner systems. This attribute controls how strictly the certificate chain validation conforms to industry security standards. **MQIA\_CERT\_VAL\_POLICY** is valid only on AIX, Linux, and Windows. For more information, see [Certificate validation policies in IBM MQ](#).

**z/OS** **MQIA\_CHANNEL\_AUTO\_DEF**

Control attribute for automatic channel definition. **MQIA\_CHANNEL\_AUTO\_DEF** is not valid on z/OS.

**z/OS** **MQIA\_CHANNEL\_AUTO\_DEF\_EVENT**

Control attribute for automatic channel definition events. **MQIA\_CHANNEL\_AUTO\_DEF\_EVENT** is not valid on z/OS.

**MQIA\_CHANNEL\_EVENT**

Control attribute for channel events.

**z/OS** **MQIA\_CHINIT\_ADAPTERS**

Number of adapter subtasks to use for processing IBM MQ calls. **MQIA\_CHINIT\_ADAPTERS** is valid on z/OS only.

**MQIA\_CHINIT\_CONTROL**

Start channel initiator automatically when queue manager starts.

**z/OS** **MQIA\_CHINIT\_DISPATCHERS**

Number of dispatchers to use for the channel initiator. **MQIA\_CHINIT\_DISPATCHERS** is valid on z/OS only.

**z/OS** **MQIA\_CHINIT\_SERVICE\_PARM**

Reserved for use by IBM. **MQIA\_CHINIT\_SERVICE\_PARM** is valid only on z/OS.

**z/OS** **MQIA\_CHINIT\_TRACE\_AUTO\_START**

Specifies whether the channel initiator trace must start automatically. **MQIA\_CHINIT\_TRACE\_AUTO\_START** is valid on z/OS only.

**z/OS** **MQIA\_CHINIT\_TRACE\_TABLE\_SIZE**

Size, in megabytes, of the trace data space of the channel initiator. **MQIA\_CHINIT\_TRACE\_TABLE\_SIZE** is valid on z/OS only.

**MQIA\_CHLAUTH\_RECORDS**

Control attribute for checking of channel authentication records.

**MQIA\_CLUSTER\_WORKLOAD\_LENGTH**

Maximum length of the message passed to the cluster workload exit.

**MQIA\_CLWL\_MRU\_CHANNELS**

Cluster workload most recently used channels.

**MQIA\_CLWL\_USEQ**

Cluster workload remote queue use.

**MQIA\_CMD\_SERVER\_CONTROL**

Start command server automatically when queue manager starts.

**MQIA\_CODED\_CHAR\_SET\_ID**

Coded character set identifier.

**MQIA\_COMMAND\_EVENT**

Control attribute for command events.

**MQIA\_COMMAND\_LEVEL**

Command level supported by queue manager.

**MQIA\_CONFIGURATION\_EVENT**

Control attribute for configuration events.

**MQIA\_CPI\_LEVEL**

Reserved for use by IBM.

**MQIA\_DEF\_CLUSTER\_XMIT\_Q\_TYPE**

Default transmission queue type to be used for cluster-sender channels.

**Multi MQIA\_DIST\_LISTS**

Distribution list support. This parameter is not valid on z/OS.

**z/OS MQIA\_DNS\_WLM**

Specifies whether the TCP listener that handles inbound transmissions for the queue sharing group must register with Workload Manager (WLM) for DDNS. **MQIA\_DNS\_WLM** is valid on z/OS only.

**z/OS MQIA\_EXPIRY\_INTERVAL**

Expiry interval. This parameter is valid on z/OS only.

**z/OS MQIA\_GROUP\_UR**

Control attribute for whether transactional applications can connect with a GROUP unit of recovery disposition. This parameter is valid only on z/OS.

**z/OS MQIA\_IGQ\_PUT\_AUTHORITY**

Intra-group queuing put authority. This parameter is valid on z/OS only.

**MQIA\_INHIBIT\_EVENT**

Control attribute for inhibit events.

**z/OS MQIA\_INTRA\_GROUP\_queuing**

Intra-group queuing support. This parameter is valid on z/OS only.

**MQIA\_IP\_ADDRESS\_VERSION**

IP address version selector.

**z/OS MQIA\_LISTENER\_TIMER**

Listener restart interval. **MQIA\_LISTENER\_TIMER** is valid on z/OS only.

**MQIA\_LOCAL\_EVENT**

Control attribute for local events.

**MQIA\_LOGGER\_EVENT**

Control attribute for recovery log events.

**z/OS MQIA\_LU62\_CHANNELS**

Maximum number of LU 6.2 channels. **MQIA\_LU62\_CHANNELS** is valid on z/OS only.

**MQIA\_MSG\_MARK\_BROWSE\_INTERVAL**

Interval for which messages that were browsed, remain marked.

**z/OS MQIA\_MAX\_CHANNELS**

Maximum number of channels that can be current. **MQIA\_MAX\_CHANNELS** is valid on z/OS only.

**MQIA\_MAX\_HANDLES**

Maximum number of handles.

**MQIA\_MAX\_MSG\_LENGTH**

Maximum message length.

**MQIA\_MAX\_PRIORITY**

Maximum priority.

**MQIA\_MAX\_PROPERTIES\_LENGTH**

Maximum properties length.

**MQIA\_MAX\_UNCOMMITTED\_MSGS**

Maximum number of uncommitted messages within a unit of work.

**ALW MQIA\_MEDIA\_IMAGE\_INTERVAL**

The target frequency with which the queue manager automatically writes media images.

**ALW MQIA\_MEDIA\_IMAGE\_LOG\_LENGTH**

The target size of the recovery log.

**ALW MQIA\_MEDIA\_IMAGE\_RECOVER\_OBJ**

Specifies the recoverable objects from a media image, if linear logging is being used.

**ALW MQIA\_MEDIA\_IMAGE\_RECOVER\_Q**

Specifies whether local and permanent dynamic queues defined with **ImageRecoverQueue** set to **MQIMGRCOV\_AS\_Q\_MGR** are recoverable from a media image.

**ALW MQIA\_MEDIA\_IMAGE\_SCHEDULING**

Whether the queue manager automatically writes media images.

**MQIA\_MONITORING\_AUTO\_CLUSSDR**

Default value of the **ChannelMonitoring** attribute of automatically defined cluster-sender channels.

**MQIA\_MONITORING\_CHANNEL**

Specifies whether channel monitoring is enabled.

**MQIA\_MONITORING\_Q**

Specifies whether queue monitoring is enabled.

**z/OS MQIA\_OUTBOUND\_PORT\_MAX**

Maximum value in the range for the binding of outgoing channels. **MQIA\_OUTBOUND\_PORT\_MAX** is valid on z/OS only.

**z/OS MQIA\_OUTBOUND\_PORT\_MIN**

Minimum value in the range for the binding of outgoing channels. **MQIA\_OUTBOUND\_PORT\_MIN** is valid on z/OS only.

**MQIA\_PERFORMANCE\_EVENT**

Control attribute for performance events.

**MQIA\_PLATFORM**

Platform on which the queue manager resides.

**z/OS MQIA\_PROT\_POLICY\_CAPABILITY**

Specifies whether Advanced Message Security is installed for the version of IBM MQ that the queue manager is running.

**MQIA\_PUBSUB\_CLUSTER**

Controls whether this queue manager participates in the publish/subscribe clustering.

**MQIA\_PUBSUB\_MAXMSG\_RETRY\_COUNT**

The number of retries when processing (under sync point) a failed command message

**MQIA\_PUBSUB\_MODE**

Inquires if the publish/subscribe engine and the queued publish/subscribe interface are running, which allow applications to publish/subscribe by using the application programming interface and the queues that are being monitored by the queued publish/subscribe interface.

**MQIA\_PUBSUB\_NP\_MSG**

Specifies whether to discard (or keep) an undelivered input message.

**MQIA\_PUBSUB\_NP\_RESP**

The behavior of undelivered response messages.

**MQIA\_PUBSUB\_SYNC\_PT**

Specifies whether only persistent (or all) messages must be processed under sync point.

**z/OS MQIA\_QMGR\_CFCONLOS**

Specifies action to be taken when the queue manager loses connectivity to the administration structure, or any CF structure with CFCONLOS set to **ASQMGR**. **MQIA\_QMGR\_CFCONLOS** is valid on z/OS only.

**z/OS MQIA\_RECEIVE\_TIMEOUT**

How long a TCP/IP channel waits to receive data from its partner. **MQIA\_RECEIVE\_TIMEOUT** is valid on z/OS only.

**z/OS MQIA\_RECEIVE\_TIMEOUT\_MIN**

Minimum length of time that a TCP/IP channel waits to receive data from its partner . **MQIA\_RECEIVE\_TIMEOUT\_MIN** is valid on z/OS only.

**z/OS MQIA\_RECEIVE\_TIMEOUT\_TYPE**

Qualifier to apply to the **ReceiveTimeout** parameter. **MQIA\_RECEIVE\_TIMEOUT\_TYPE** is valid on z/OS only.

**MQIA\_REMOTE\_EVENT**

Control attribute for remote events.

**z/OS MQIA\_SECURITY\_CASE**

Specifies whether the queue manager supports security profile names either in mixed case, or in uppercase only. **MQIA\_SECURITY\_CASE** is valid only on z/OS.

**z/OS MQIA\_SHARED\_Q\_Q\_MGR\_NAME**

When a queue manager makes an MQOPEN call for a shared queue and the queue manager that is specified in the **ObjectQmgrName** parameter of the MQOPEN call is in the same queue sharing group as the processing queue manager, the SQQMNAME attribute specifies whether the **ObjectQmgrName** is used or whether the processing queue manager opens the shared queue directly. **MQIA\_SHARED\_Q\_Q\_MGR\_NAME** is valid only on z/OS.

**MQIA\_SSL\_EVENT**

Control attribute for TLS events.

**MQIA\_SSL\_FIPS\_REQUIRED**

Specifies whether only FIPS-certified algorithms are to be used if cryptography is executed in IBM MQ rather than in the cryptographic hardware itself.

**MQIA\_SSL\_RESET\_COUNT**

TLS key reset count.

**z/OS MQIA\_SSL\_TASKS**

TLS tasks. This parameter is valid on z/OS only.

**MQIA\_START\_STOP\_EVENT**

Control attribute for start stop events.

**MQIA\_STATISTICS\_AUTO\_CLUSSDR**

Specifies whether statistics data is to be collected for auto-defined cluster-sender channels and, if so, the rate of data collection.

**MQIA\_STATISTICS\_CHANNEL**

Specifies whether statistics monitoring data is to be collected for channels and, if so, the rate of data collection.

**ALW MQIA\_STATISTICS\_INTERVAL**

Statistics data collection interval. **MQIA\_STATISTICS\_INTERVAL** is valid only on AIX, Linux, and Windows.

**ALW MQIA\_STATISTICS\_MQI**

Specifies whether statistics monitoring data is to be collected for the queue manager. **MQIA\_STATISTICS\_MQI** is valid only on AIX, Linux, and Windows.

**ALW**

**MQIA\_STATISTICS\_Q**

Specifies whether statistics monitoring data is to be collected for queues. **MQIA\_STATISTICS\_Q** is valid only on AIX, Linux, and Windows.

**MQIA\_SUITE\_B\_STRENGTH**

Specifies whether Suite B-compliant cryptography is used and the level of strength employed. For more information about Suite B configuration and its effect on TLS channels, see [NSA Suite B Cryptography in IBM MQ](#).

**MQIA\_SYNCPOINT**

Sync point availability.

**MQIA\_TCP\_CHANNELS**

Maximum number of channels that can be current, or clients that can be connected, that use the TCP/IP transmission protocol This is valid only on z/OS.

**z/OS**

**MQIA\_TCP\_KEEP\_ALIVE**

Specifies whether the TCP KEEPALIVE facility is to be used to check whether the other end of a connection is still available. **MQIA\_TCP\_KEEP\_ALIVE** is valid only on z/OS.

**z/OS**

**MQIA\_TCP\_STACK\_TYPE**

Specifies whether the channel initiator can use only the TCP/IP address space specified in the **TCPName** parameter, or can optionally bind to any selected TCP/IP address.

**MQIA\_TCP\_STACK\_TYPE** is valid only on z/OS.

**MQIA\_TRACE\_ROUTE\_RECORDING**

Specifies whether trace-route information can be recorded and reply messages generated.

**MQIA\_TREE\_LIFE\_TIME**

The lifetime of non-administrative topics.

**MQIA\_TRIGGER\_INTERVAL**

Trigger interval.

**MQIA\_XR\_CAPABILITY**

Specifies whether telemetry commands are supported.

**MQIACF\_Q\_MGR\_CLUSTER**

All clustering attributes. These attributes are:

- **MQCA\_CLUSTER\_WORKLOAD\_DATA**
- **MQCA\_CLUSTER\_WORKLOAD\_EXIT**
- **MQCA\_CHANNEL\_AUTO\_DEF\_EXIT**
- **MQCA\_REPOSITORY\_NAME**
- **MQCA\_REPOSITORY\_NAMELIST**
- **MQIA\_CLUSTER\_WORKLOAD\_LENGTH**
- **MQIA\_CLWL\_MRU\_CHANNELS**
- **MQIA\_CLWL\_USEQ**
- **MQIA\_MONITORING\_AUTO\_CLUSSDR**
- **MQCA\_Q\_MGR\_IDENTIFIER**

**MQIACF\_Q\_MGR\_DQM**

All distributed queuing attributes. These attributes are:

- **MQCA\_CERT\_LABEL**
- **MQCA\_CHANNEL\_AUTO\_DEF\_EXIT**
- **MQCA\_DEAD\_LETTER\_Q\_NAME**
- **MQCA\_DEF\_XMIT\_Q\_NAME**
- **MQCA\_DNS\_GROUP**
- **MQCA\_IGQ\_USER\_ID**

- MQCA\_LU\_GROUP\_NAME
- MQCA\_LU\_NAME
- MQCA\_LU62\_ARM\_SUFFIX
- MQCA\_Q\_MGR\_IDENTIFIER
- MQCA\_QSG\_CERT\_LABEL
- MQCA\_SSL\_CRL\_NAMELIST
- MQCA\_SSL\_CRYPTO\_HARDWARE
- **V 9.3.0** **V 9.3.0** MQCA\_SSL\_KEY\_REPO\_PASSWORD
- MQCA\_SSL\_KEY\_REPOSITORY
- MQCA\_TCP\_NAME
- MQIA\_ACTIVE\_CHANNELS
- MQIA\_ADOPTNEWMCA\_CHECK
- MQIA\_ADOPTNEWMCA\_TYPE
- MQIA\_CERT\_VAL\_POLICY
- MQIA\_CHANNEL\_AUTO\_DEF
- MQIA\_CHANNEL\_AUTO\_DEF\_EVENT
- MQIA\_CHANNEL\_EVENT
- MQIA\_CHINIT\_ADAPTERS
- MQIA\_CHINIT\_CONTROL
- MQIA\_CHINIT\_DISPATCHERS
- MQIA\_CHINIT\_SERVICE\_PARM
- MQIA\_CHINIT\_TRACE\_AUTO\_START
- MQIA\_CHINIT\_TRACE\_TABLE\_SIZE
- MQIA\_CHLAUTH\_RECORDS
- MQIA\_INTRA\_GROUP\_queuing
- MQIA\_IGQ\_PUT\_AUTHORITY
- MQIA\_IP\_ADDRESS\_VERSION
- MQIA\_LISTENER\_TIMER
- MQIA\_LU62\_CHANNELS
- MQIA\_MAX\_CHANNELS
- MQIA\_MONITORING\_CHANNEL
- MQIA\_OUTBOUND\_PORT\_MAX
- MQIA\_OUTBOUND\_PORT\_MIN
- MQIA\_RECEIVE\_TIMEOUT
- MQIA\_RECEIVE\_TIMEOUT\_MIN
- MQIA\_RECEIVE\_TIMEOUT\_TYPE
- MQIA\_SSL\_EVENT
- MQIA\_SSL\_FIPS\_REQUIRED
- MQIA\_SSL\_RESET\_COUNT
- MQIA\_SSL\_TASKS
- MQIA\_STATISTICS\_AUTO\_CLUSSDR
- MQIA\_TCP\_CHANNELS
- MQIA\_TCP\_KEEP\_ALIVE

- MQIA\_TCP\_STACK\_TYPE

#### **MQIACF\_Q\_MGR\_EVENT**

All event control attributes. These attributes are:

- MQIA\_AUTHORITY\_EVENT
- MQIA\_BRIDGE\_EVENT
- MQIA\_CHANNEL\_EVENT
- MQIA\_COMMAND\_EVENT
- MQIA\_CONFIGURATION\_EVENT
- MQIA\_INHIBIT\_EVENT
- MQIA\_LOCAL\_EVENT
- MQIA\_LOGGER\_EVENT
- MQIA\_PERFORMANCE\_EVENT
- MQIA\_REMOTE\_EVENT
- MQIA\_SSL\_EVENT
- MQIA\_START\_STOP\_EVENT



#### **MQIACF\_Q\_MGR\_PUBSUB**

All queue manager publish/subscribe attributes. These attributes are:

- MQCA\_PARENT
- MQIA\_PUBSUB\_MAXMSG\_RETRY\_COUNT
- MQIA\_PUBSUB\_MODE
- MQIA\_PUBSUB\_NP\_MSG
- MQIA\_PUBSUB\_NP\_RESP
- MQIA\_PUBSUB\_SYNC\_PT
- MQIA\_TREE\_LIFE\_TIME

#### **MQIACF\_Q\_MGR\_SYSTEM**

All queue manager system attributes. These attributes are:

- MQCA\_ALTERATION\_DATE
- MQCA\_ALTERATION\_TIME
- MQCA\_COMMAND\_INPUT\_Q\_NAME
- MQCA\_CONN\_AUTH
- MQCA\_CREATION\_DATE
- MQCA\_CREATION\_TIME
- MQCA\_CUSTOM
- MQCA\_DEAD\_LETTER\_Q\_NAME
-   MQCA\_INITIAL\_KEY
- MQCA\_Q\_MGR\_DESC
- MQCA\_Q\_MGR\_NAME
- MQCA\_QSG\_NAME
- MQCA\_VERSION
- MQIA\_ACCOUNTING\_CONN\_OVERRIDE
- MQIA\_ACCOUNTING\_INTERVAL
- MQIA\_ACCOUNTING\_MQI
- MQIA\_ACCOUNTING\_Q

- MQIA\_ACTIVITY\_CONN\_OVERRIDE
- MQIA\_ACTIVITY\_RECORDING
- MQIA\_ACTIVITY\_TRACE
- MQIA\_ADVANCED\_CAPABILITY
- MQIA\_CMD\_SERVER\_CONTROL
- MQIA\_CODED\_CHAR\_SET\_ID
- MQIA\_COMMAND\_LEVEL
- MQIA\_CPI\_LEVEL
- MQIA\_DIST\_LISTS
- MQIA\_EXPIRY\_INTERVAL
- MQIA\_GROUP\_UR
- MQIA\_MAX\_HANDLES
- MQIA\_MAX\_MSG\_LENGTH
- MQIA\_MAX\_PRIORITY
- MQIA\_MAX\_PROPERTIES\_LENGTH
- MQIA\_MAX\_UNCOMMITTED\_MSGS
- MQIA\_MEDIA\_IMAGE\_INTERVAL
- MQIA\_MEDIA\_IMAGE\_LOG\_LENGTH
- MQIA\_MEDIA\_IMAGE\_RECOVER\_OBJ
- MQIA\_MEDIA\_IMAGE\_RECOVER\_Q
- MQIA\_MEDIA\_IMAGE\_SCHEDULING
- MQIA\_MONITORING\_Q
- MQIA\_MSG\_MARK\_BROWSE\_INTERVAL
- MQIA\_PROT\_POLICY\_CAPABILITY
- MQIA\_QMGR\_CFCONLOS
- MQIA\_SECURITY\_CASE
- MQIA\_PLATFORM
- MQIA\_SHARED\_Q\_Q\_MGR\_NAME
- MQIA\_STATISTICS\_INTERVAL
- MQIA\_STATISTICS\_MQI
- MQIA\_STATISTICS\_Q
- MQIA\_SYNCPOINT
- MQIA\_TRACE\_ROUTE\_RECORDING
- MQIA\_TRIGGER\_INTERVAL
- MQIA\_XR\_CAPABILITY

### ***MQCMD\_INQUIRE\_Q\_MGR (Inquire Queue Manager) Response***

The response to the Inquire Queue Manager (MQCMD\_INQUIRE\_Q\_MGR) PCF command consists of the response header followed by the *QMgrName* structure and the requested combination of attribute parameter structures.

**Always returned:**

*QMgrName*

**Returned if requested:**

*AccountingConnOverride, AccountingInterval, ActivityConnOverride, ActivityRecording, ActivityTrace, AdoptNewMCACheck, AdoptNewMCAType,*



*AdvancedCapability, AlterationDate, AlterationTime, AMQPCapability, AuthorityEvent, z/OS BridgeEvent, CertificateLabel, CertificateValPolicy, z/OS CFConlos, ChannelAutoDef, ChannelAutoDefEvent, ChannelAutoDefExit, ChannelAuthenticationRecords, ChannelEvent, ChannelInitiatorControl, ChannelMonitoring, ChannelStatistics, z/OS ChinitAdapters, z/OS ChinitDispatchers, z/OS ChinitServiceParm, z/OS ChinitTraceAutoStart, z/OS ChinitTraceTableSize, ClusterSenderMonitoringDefault, ClusterSenderStatistics, ClusterWorkloadData, ClusterWorkloadExit, ClusterWorkloadLength, CLWLMRUChannels, CLWLUseQ, CodedCharSetId, CommandEvent, CommandInputQName, CommandLevel, CommandServerControl, ConfigurationEvent, ConnAuth, CreationDate, CreationTime, Custom, DeadLetterQName, DefClusterXmitQueueType, DefXmitQName, DistLists, DNSGroup, z/OS DNSWLM, EncryptionPolicySuiteB, ExpiryInterval, GroupUR, z/OS IGQPutAuthority, z/OS IGQUserId, ImageInterval, ImagelogLength, ImageRecoverObject, ImageRecoverQueue, ImageSchedule, InhibitEvent, V 9.3.0 InitialKey, IntraGroupQueuing, IPAddressVersion, ListenerTimer, LocalEvent, LoggerEvent, z/OS LUGroupName, z/OS LUName, z/OS LU62ARMSuffix, z/OS LU62Channels, z/OS MaxChannels, z/OS MaxActiveChannels, MaxHandles, MaxMsgLength, MaxPriority, MaxPropertiesLength, MaxUncommittedMsgs, MQIAccounting, MQIStatistics z/OS OutboundPortMax, z/OS OutboundPortMin, Parent, PerformanceEvent, Platform, PubSubClus, PubSubMaxMsgRetryCount, PubSubMode, QmgrDesc, QMgrIdentifier, z/OS QSGCertificateLabel, z/OS QSGName, QueueAccounting, QueueMonitoring, QueueStatistics, ReceiveTimeout, ReceiveTimeoutMin, ReceiveTimeoutType, RemoteEvent, RepositoryName, RepositoryNameList, RevDns, z/OS SecurityCase, SharedQMgrName, Splcap, SSLCRLNameList, SSLCryptoHardware, SSLEvent, SSLFIPSRequired, SSLKeyRepository, V 9.3.0 SSLKeyRepositoryPassword, SSLKeyResetCount, SSLTasks, StartStopEvent, StatisticsInterval, SyncPoint, TCPChannels, TCPKeepAlive, TCPName, TCPStackType, TraceRouteRecording, TreeLifeTime, TriggerInterval, Version*

## Response data

### AccountingConnOverride (MQCFIN)

Specifies whether applications can override the settings of the *QueueAccounting* and *MQIAccounting* queue manager parameters (parameter identifier: MQIA\_ACCOUNTING\_CONN\_OVERRIDE).

The value can be any of the following values:

#### **MQMON\_DISABLED**

Applications cannot override the settings of the **QueueAccounting** and **MQIAccounting** parameters.

#### **MQMON\_ENABLED**

Applications can override the settings of the **QueueAccounting** and **MQIAccounting** parameters by using the options field of the MQCNO structure of the MQCONN API call.

This parameter applies only to AIX, Linux, and Windows.

### AccountingInterval (MQCFIN)

The time interval, in seconds, at which intermediate accounting records are written (parameter identifier: MQIA\_ACCOUNTING\_INTERVAL).

It is a value in the range 1 through 604 000.

This parameter applies only to AIX, Linux, and Windows.

### **ActivityConnOverride (MQCFIN)**

Specifies whether applications can override the setting of the ACTVTRC value in the queue manager attribute (parameter identifier: MQIA\_ACTIVITY\_CONN\_OVERRIDE).

The value can be any of the following values:

#### **MQMON\_DISABLED**

Applications cannot override the setting of the ACTVTRC queue manager attribute using the Options field in the MQCNO structure on the MQCONN call. This is the default value.

#### **MQMON\_ENABLED**

Applications can override the ACTVTRC queue manager attribute using the Options field in the MQCNO structure.

Changes to this value are only effective for connections to the queue manager after the change to the attribute.

This parameter applies only to IBM i, AIX, Linux, and Windows.

### **ActivityRecording (MQCFIN)**

Whether activity reports can be generated (parameter identifier: MQIA\_ACTIVITY\_RECORDING).

The value can be:

#### **MQRECORDING\_DISABLED**

Activity reports cannot be generated.

#### **MQRECORDING\_MSG**

Activity reports can be generated and sent to the destination specified by the originator of the message causing the report.

#### **MQRECORDING\_Q**

Activity reports can be generated and sent to SYSTEM.ADMIN.ACTIVITY.QUEUE.

### **Multi ActivityTrace (MQCFIN)**

Whether activity reports can be generated (parameter identifier: MQIA\_ACTIVITY\_TRACE).

The value can be:

#### **MQMON\_OFF**

Do not collect IBM MQ MQI application activity trace. This is the default value.

If you set the queue manager attribute ACTVCON0 to ENABLED, this value might be overridden for individual connections using the Options field in the MQCNO structure.

#### **MQMON\_ON**

Collect IBM MQ MQI application activity trace.

Changes to this value are only effective for connections to the queue manager after the change to the attribute.

This parameter applies only to IBM i, AIX, Linux, and Windows.

### **z/OS AdoptNewMCACheck (MQCFIN)**

The elements checked to determine whether an MCA must be adopted (restarted) when a new inbound channel is detected. It is adopted if it has the same name as a currently active MCA (parameter identifier: MQIA\_ADOPTNEWMCA\_CHECK).

The value can be:

#### **MQADOPT\_CHECK\_Q\_MGR\_NAME**

Check the queue manager name.

#### **MQADOPT\_CHECK\_NET\_ADDR**

Check the network address.

**MQADOPT\_CHECK\_ALL**

Check the queue manager name and network address.

**MQADOPT\_CHECK\_NONE**

Do not check any elements.

This parameter is valid only on z/OS.

**z/OS** **AdoptNewMCAType (MQCFIL)**

Adoption of orphaned channel instances (parameter identifier: MQIA\_ADOPTNEWMCA\_TYPE).

The value can be:

**MQADOPT\_TYPE\_NO**

Do not adopt orphaned channel instances.

**MQADOPT\_TYPE\_ALL**

Adopt all channel types.

This parameter is valid only on z/OS.

**MQ Adv.** **AdvancedCapability (MQCFIN)**

Whether IBM MQ Advanced extended capabilities are available for a queue manager (parameter identifier: MQIA\_ADVANCED\_CAPABILITY).

**z/OS** On z/OS, the queue manager sets the value to be MQCAP\_SUPPORTED, only if the value of **QMGRPROD** is ADVANCEDVUE. For any other value of **QMGRPROD**, or if **QMGRPROD** is not set, the queue manager sets the value to MQCAP\_NOTSUPPORTED. See [“START QMGR \(start queue manager\) on z/OS” on page 953](#) for more information.

**Multi** On other platforms, from IBM MQ 9.1, the queue manager sets the value to be MQCAP\_SUPPORTED, only if you have installed Managed File Transfer, XR, or Advanced Message Security. If you have not installed Managed File Transfer, XR, or Advanced Message Security, **AdvancedCapability** is set to MQCAP\_NOTSUPPORTED. See [IBM MQ components and features](#) for more information.

**AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date, in the form yyyy-mm-dd, on which the information was last altered.

**AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time, in the form hh.mm.ss, at which the information was last altered.

**ALW** **AMQPCapability (MQCFIN)**

Whether AMQP capabilities are available on a queue manager (parameter identifier: MQIA\_AMQP\_CAPABILITY).

The value can be one of the following values:

**MQCAP\_SUPPORTED**

AMQP capability has been installed.

**MQCAP\_NOT\_SUPPORTED**

AMQP capability has not been installed.

**AuthorityEvent (MQCFIN)**

Controls whether authorization (Not Authorized) events are generated (parameter identifier: MQIA\_AUTHORITY\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**z/OS BridgeEvent (MQCFIN)**

Controls whether IMS bridge events are generated (parameter identifier: MQIA\_BRIDGE\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

This parameter is valid only on z/OS.

**CertificateLabel (MQCFST)**

Certificate label in the key repository for this queue manager to use (parameter identifier: MQCA\_CERT\_LABEL).

The maximum length of the string is MQ\_CERT\_LABEL\_LENGTH.

**ALW CertificateValPolicy (MQCFIN)**

Specifies which TLS certificate validation policy is used to validate digital certificates received from remote partner systems (parameter identifier: MQIA\_CERT\_VAL\_POLICY).

This attribute can be used to control how strictly the certificate chain validation conforms to industry security standards. This parameter is valid only on AIX, Linux, and Windows. For more information, see [Certificate validation policies in IBM MQ](#).

The value can be any of the following values:

**MQ\_CERT\_VAL\_POLICY\_ANY**

Apply each of the certificate validation policies supported by the secure sockets library and accept the certificate chain if any of the policies considers the certificate chain valid. This setting can be used for maximum backwards compatibility with older digital certificates which do not comply with the modern certificate standards.

**MQ\_CERT\_VAL\_POLICY\_RFC5280**

Apply only the RFC 5280 compliant certificate validation policy. This setting provides stricter validation than the ANY setting, but rejects some older digital certificates.

**z/OS CFConlos (MQCFIN)**

Specifies the action to be taken when the queue manager loses connectivity to the administration structure, or any CF structures with CFCONLOS set to ASQMGR (parameter identifier: MQIA\_QMGR\_CFCONLOS).

The value can be:

**MQCFCONLOS\_TERMINATE**

The queue manager terminates when connectivity to CF structures is lost.

**MQCFCONLOS\_TOLERATE**

The queue manager tolerates loss of connectivity to CF structures without terminating.

This parameter is valid only on z/OS.

**ChannelAutoDef (MQCFIN)**

Controls whether receiver and server-connection channels can be auto-defined (parameter identifier: MQIA\_CHANNEL\_AUTO\_DEF).

The value can be:

**MQCHAD\_DISABLED**

Channel auto-definition disabled.

**MQCHAD\_ENABLED**

Channel auto-definition enabled.

**ChannelAutoDefEvent (MQCFIN)**

Controls whether channel auto-definition events are generated (parameter identifier: MQIA\_CHANNEL\_AUTO\_DEF\_EVENT), when a receiver, server-connection, or cluster-sender channel is auto-defined.

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**ChannelAutoDefExit (MQCFST)**

Channel auto-definition exit name (parameter identifier: MQCA\_CHANNEL\_AUTO\_DEF\_EXIT).

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

**ChannelAuthenticationRecords (MQCFIN)**

Controls whether channel authentication records are checked (parameter identifier: MQIA\_CHLAUTH\_RECORDS).

The value can be:

**MQCHLA\_DISABLED**

Channel authentication records are not checked.

**MQCHLA\_ENABLED**

Channel authentication records are checked.

**ChannelEvent (MQCFIN)**

Controls whether channel events are generated (parameter identifier: MQIA\_CHANNEL\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**MQEVR\_EXCEPTION**

Reporting of exception channel events enabled.

**ChannelInitiatorControl (MQCFIN)**

Start the channel initiator during queue manager start (parameter identifier: MQIA\_CHINIT\_CONTROL). This parameter is not available on z/OS.

The value can be:

**MQSVC\_CONTROL\_MANUAL**

The channel initiator is not to be started automatically when the queue manager starts.

**MQSVC\_CONTROL\_Q\_MGR**

The channel initiator is to be started automatically when the queue manager starts.

**ChannelMonitoring (MQCFIN)**

Default setting for online monitoring for channels (parameter identifier: MQIA\_MONITORING\_CHANNEL).

If the *ChannelMonitoring* channel attribute is set to MQMON\_Q\_MGR, this attribute specifies the value which is assumed by the channel. The value can be any of the following values:

**MQMON\_OFF**

Online monitoring data collection is turned off.

**MQMON\_NONE**

Online monitoring data collection is turned off for channels regardless of the setting of their **ChannelMonitoring** attribute.

**MQMON\_LOW**

Online monitoring data collection is turned on, with a low ratio of data collection.

**MQMON\_MEDIUM**

Online monitoring data collection is turned on, with a moderate ratio of data collection.

**MQMON\_HIGH**

Online monitoring data collection is turned on, with a high ratio of data collection.

**z/OS ChannelStatistics (MQCFIN)**

Specifies whether statistics data is to be collected for channels (parameter identifier: MQIA\_STATISTICS\_CHANNEL).

The value can be:

**MQMON\_OFF**

Statistics data collection is turned off.

**MQMON\_LOW**

Statistics data collection is turned on, with a low ratio of data collection.

**MQMON\_MEDIUM**

Statistics data collection is turned on, with a moderate ratio of data collection.

**MQMON\_HIGH**

Statistics data collection is turned on, with a high ratio of data collection.

On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

This parameter is valid only on z/OS.

**z/OS ChinitAdapters (MQCFIN)**

Number of adapter subtasks (parameter identifier: MQIA\_CHINIT\_ADAPTERS).

The number of adapter subtasks to use for processing IBM MQ calls. This parameter is valid only on z/OS.

**z/OS ChinitDispatchers (MQCFIN)**

Number of dispatchers (parameter identifier: MQIA\_CHINIT\_DISPATCHERS).

The number of dispatchers to use for the channel initiator. This parameter is valid only on z/OS.

**z/OS ChinitServiceParm (MQCFST)**

Reserved for use by IBM (parameter identifier: MQCA\_CHINIT\_SERVICE\_PARM).

**z/OS ChinitTraceAutoStart (MQCFIN)**

Specifies whether the channel initiator trace must start automatically (parameter identifier: MQIA\_CHINIT\_TRACE\_AUTO\_START).

The value can be:

**MQTRAXSTR\_YES**

Channel initiator trace is to start automatically.

**MQTRAXSTR\_NO**

Channel initiator trace is not to start automatically.

This parameter is valid only on z/OS.

▶ z/OS

### **ChinitTraceTableSize (MQCFIN)**

The size, in megabytes, of the trace data space of the channel initiator (parameter identifier: MQIA\_CHINIT\_TRACE\_TABLE\_SIZE).

This parameter is valid only on z/OS.

### **ClusterSenderMonitoringDefault (MQCFIN)**

Setting for online monitoring for automatically defined cluster-sender channels (parameter identifier: MQIA\_MONITORING\_AUTO\_CLUSSDR).

The value can be:

#### **MQMON\_Q\_MGR**

Collection of online monitoring data is inherited from the setting of the queue manager's **ChannelMonitoring** parameter.

#### **MQMON\_OFF**

Monitoring for the channel is disabled.

#### **MQMON\_LOW**

Specifies a low rate of data collection with a minimal effect on system performance unless **ChannelMonitoring** for the queue manager is MQMON\_NONE. The data collected is not likely to be the most current.

#### **MQMON\_MEDIUM**

Specifies a moderate rate of data collection with limited effect on system performance unless **ChannelMonitoring** for the queue manager is MQMON\_NONE.

#### **MQMON\_HIGH**

Specifies a high rate of data collection with a likely effect on system performance unless **ChannelMonitoring** for the queue manager is MQMON\_NONE. The data collected is the most current available.

▶ z/OS

On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results.

### **ClusterSenderStatistics (MQCFIN)**

Specifies whether statistics data is to be collected for auto-defined cluster-sender channels (parameter identifier: MQIA\_STATISTICS\_AUTO\_CLUSSDR).

The value can be:

#### **MQMON\_Q\_MGR**

Collection of statistics data is inherited from the setting of the queue manager's **ChannelStatistics** parameter.

#### **MQMON\_OFF**

Statistics data collection for the channel is disabled.

#### **MQMON\_LOW**

Specifies a low rate of data collection with a minimal effect on system performance.

#### **MQMON\_MEDIUM**

Specifies a moderate rate of data collection.

#### **MQMON\_HIGH**

Specifies a high rate of data collection.

▶ z/OS

On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying LOW, MEDIUM, or HIGH makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

### **ClusterWorkLoadData (MQCFST)**

Data passed to the cluster workload exit (parameter identifier: MQCA\_CLUSTER\_WORKLOAD\_DATA).

**ClusterWorkLoadExit (MQCFST)**

Name of the cluster workload exit (parameter identifier: MQCA\_CLUSTER\_WORKLOAD\_EXIT).

The maximum length of the exit name depends on the environment in which the exit is running. MQ\_EXIT\_NAME\_LENGTH gives the maximum length for the environment in which your application is running. MQ\_MAX\_EXIT\_NAME\_LENGTH gives the maximum for all supported environments.

**ClusterWorkLoadLength (MQCFIN)**

Cluster workload length (parameter identifier: MQIA\_CLUSTER\_WORKLOAD\_LENGTH).

The maximum length of the message passed to the cluster workload exit.

**CLWLMRUChannels (MQCFIN)**

Cluster workload most recently used (MRU) channels (parameter identifier: MQIA\_CLWL\_MRU\_CHANNELS).

The maximum number of active most recently used outbound channels.

**CLWLUseQ (MQCFIN)**

Use of remote queue (parameter identifier: MQIA\_CLWL\_USEQ).

Specifies whether a cluster queue manager is to use remote puts to other queues defined in other queue managers within the cluster during workload management.

The value can be any of the following values:

**MQCLWL\_USEQ\_ANY**

Use remote queues.

**MQCLWL\_USEQ\_LOCAL**

Do not use remote queues.

**CodedCharSetId (MQCFIN)**

Coded character set identifier (parameter identifier: MQIA\_CODED\_CHAR\_SET\_ID).

**CommandEvent (MQCFIN)**

Controls whether command events are generated (parameter identifier: MQIA\_COMMAND\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**MQEVR\_NODISPLAY**

Event reporting enabled for all successful commands except Inquire commands.

**CommandInputQName (MQCFST)**

Command input queue name (parameter identifier: MQCA\_COMMAND\_INPUT\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**CommandLevel (MQCFIN)**

Command level supported by queue manager (parameter identifier: MQIA\_COMMAND\_LEVEL).

The value can be:

**MQCMDL\_LEVEL\_800**

Level 800 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 8.0
- IBM MQ for IBM i 8.0
- IBM MQ for Linux 8.0



- IBM MQ for Windows 8.0
- IBM MQ for z/OS 8.0

**MQCMDL\_LEVEL\_801**

Level 801 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 8.0.0 Fix Pack 2
- IBM MQ for HP-UX 8.0.0 Fix Pack 2
- IBM MQ for IBM i 8.0.0 Fix Pack 2
- IBM MQ for Linux 8.0.0 Fix Pack 2

**MQCMDL\_LEVEL\_802**

Level 802 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 8.0.0 Fix Pack 3
- IBM MQ for IBM i 8.0.0 Fix Pack 3
- IBM MQ for Linux 8.0.0 Fix Pack 3
- IBM MQ for Windows 8.0.0 Fix Pack 3

**MQCMDL\_LEVEL\_900**

Level 900 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.0
- IBM MQ for IBM i 9.0
- IBM MQ for Linux 9.0
- IBM MQ for Windows 9.0
- IBM MQ for z/OS 9.0

**MQCMDL\_LEVEL\_901**

Level 901 of system control commands.

This value is returned by the following versions:

- IBM MQ for Linux 9.0.1
- IBM MQ for Windows 9.0.1
- IBM MQ for z/OS 9.0.1

**MQCMDL\_LEVEL\_902**

Level 902 of system control commands.

This value is returned by the following versions:

- IBM MQ for Linux 9.0.2
- IBM MQ for Windows 9.0.2
- IBM MQ for z/OS 9.0.2

**MQCMDL\_LEVEL\_903**

Level 903 of system control commands.

This value is returned by the following versions:

- IBM MQ for Linux 9.0.3
- IBM MQ for Windows 9.0.3
- IBM MQ for z/OS 9.0.3

**MQCMDL\_LEVEL\_904**

Level 904 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.0.4
- IBM MQ for Linux 9.0.4
- IBM MQ for Windows 9.0.4
- IBM MQ for z/OS 9.0.4

**MQCMDL\_LEVEL\_905**

Level 905 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.0.5
- IBM MQ for Linux 9.0.5
- IBM MQ for Windows 9.0.5
- IBM MQ for z/OS 9.0.5

**MQCMDL\_LEVEL\_910**

Level 910 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

**MQCMDL\_LEVEL\_911**

Level 911 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.1.1
- IBM MQ for Linux 9.1.1
- IBM MQ for Windows 9.1.1
- IBM MQ for z/OS 9.1.1

**MQCMDL\_LEVEL\_912**

Level 912 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.1.2
- IBM MQ for Linux 9.1.2
- IBM MQ for Windows 9.1.2
- IBM MQ for z/OS 9.1.2

**MQCMDL\_LEVEL\_913**

Level 913 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.1.3
- IBM MQ for Linux 9.1.3
- IBM MQ for Windows 9.1.3
- IBM MQ for z/OS 9.1.3

**MQCMDL\_LEVEL\_914**

Level 914 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.1.4
- IBM MQ for Linux 9.1.4
- IBM MQ for Windows 9.1.4
- IBM MQ for z/OS 9.1.4

**MQCMDL\_LEVEL\_915**

Level 915 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.1.5
- IBM MQ for Linux 9.1.5
- IBM MQ for Windows 9.1.5
- IBM MQ for z/OS 9.1.5

**MQCMDL\_LEVEL\_910**

Level 910 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

**MQCMDL\_LEVEL\_920**

Level 920 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.2
- IBM MQ for IBM i 9.2
- IBM MQ for Linux 9.2
- IBM MQ for Windows 9.2
- IBM MQ for z/OS 9.2

**MQCMDL\_LEVEL\_921**

Level 921 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.2.1
- IBM MQ for Linux 9.2.1
- IBM MQ for Windows 9.2.1
- IBM MQ for z/OS 9.2.1

**MQCMDL\_LEVEL\_922**

Level 922 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.2.2
- IBM MQ for Linux 9.2.2
- IBM MQ for Windows 9.2.2

- IBM MQ for z/OS 9.2.2

#### **MQCMDL\_LEVEL\_923**

Level 923 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.2.3
- IBM MQ for Linux 9.2.3
- IBM MQ for Windows 9.2.3
- IBM MQ for z/OS 9.2.3

#### **MQCMDL\_LEVEL\_924**

Level 924 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.2.4
- IBM MQ for Linux 9.2.4
- IBM MQ for Windows 9.2.4
- IBM MQ for z/OS 9.2.4

#### **MQCMDL\_LEVEL\_925**

Level 925 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.2.5
- IBM MQ for Linux 9.2.5
- IBM MQ for Windows 9.2.5
- IBM MQ for z/OS 9.2.5

#### **MQCMDL\_LEVEL\_930**

Level 930 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.3
- IBM MQ for IBM i 9.3
- IBM MQ for Linux 9.3
- IBM MQ for Windows 9.3
- IBM MQ for z/OS 9.3

#### **MQCMDL\_LEVEL\_931**

Level 931 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.3.1
- IBM MQ for Linux 9.3.1
- IBM MQ for Windows 9.3.1
- IBM MQ for z/OS 9.3.1

#### **MQCMDL\_LEVEL\_932**

Level 932 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX 9.3.2
- IBM MQ for Linux 9.3.2
- IBM MQ for Windows 9.3.2

- IBM MQ for z/OS 9.3.2

The set of system control commands that corresponds to a particular value of the **CommandLevel** attribute varies. It varies according to the value of the **Platform** attribute; both must be used to decide which system control commands are supported.

**Note:** Support for the HP-UX operating system for all IBM MQ components, including server and clients, is removed from IBM MQ 9.1.0.

### **CommandServerControl (MQCFIN)**

Start the command server during queue manager start (parameter identifier: MQIA\_CMD\_SERVER\_CONTROL). This parameter is not available on z/OS.

The value can be:

#### **MQSVC\_CONTROL\_MANUAL**

The command server is not to be started automatically when the queue manager starts.

#### **MQSVC\_CONTROL\_Q\_MGR**

The command server is to be started automatically when the queue manager starts.

### **ConfigurationEvent (MQCFIN)**

Controls whether configuration events are generated (parameter identifier: MQIA\_CONFIGURATION\_EVENT).

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

### **ConnAuth (MQCFST)**

Name of the authentication information object that is used to provide the location of user ID and password authentication (parameter identifier: MQCA\_CONN\_AUTH).

### **CreationDate (MQCFST)**

Creation date, in the form yyyy-mm-dd (parameter identifier: MQCA\_CREATION\_DATE).

The maximum length of the string is MQ\_CREATION\_DATE\_LENGTH.

### **CreationTime (MQCFST)**

Creation time, in the form hh.mm.ss (parameter identifier: MQCA\_CREATION\_TIME).

The maximum length of the string is MQ\_CREATION\_TIME\_LENGTH.

### **Custom (MQCFST)**

Custom attribute for new features (parameter identifier: MQCA\_CUSTOM).

This attribute is reserved for the configuration of new features before separate attributes are introduced. It can contain the values of zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE).

This description is updated when features using this attribute are introduced.

### **DeadLetterQName (MQCFST)**

Dead letter (undelivered message) queue name (parameter identifier: MQCA\_DEAD\_LETTER\_Q\_NAME).

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **DefClusterXmitQueueType (MQCFIN)**

The DefClusterXmitQueueType attribute controls which transmission queue is selected by default by cluster-sender channels to get messages from, to send the messages to cluster-receiver channels. (Parameter identifier: MQIA\_DEF\_CLUSTER\_XMIT\_Q\_TYPE.)

The values of **DefClusterXmitQueueType** are MQCLXQ\_SCTQ or MQCLXQ\_CHANNEL.

#### **MQCLXQ\_SCTQ**

All cluster-sender channels send messages from SYSTEM.CLUSTER.TRANSMIT.QUEUE. The correlID of messages placed on the transmission queue identifies which cluster-sender channel the message is destined for.

SCTQ is set when a queue manager is defined.

#### **MQCLXQ\_CHANNEL**

Each cluster-sender channel sends messages from a different transmission queue. Each transmission queue is created as a permanent dynamic queue from the model queue SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

### **DefXmitQName (MQCFST)**

Default transmission queue name (parameter identifier: MQCA\_DEF\_XMIT\_Q\_NAME).

The default transmission queue is used for the transmission of messages to remote queue managers. It is used if there is no other indication of which transmission queue to use.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **DistLists (MQCFIN)**

Distribution list support (parameter identifier: MQIA\_DIST\_LISTS).

The value can be:

#### **MQDL\_SUPPORTED**

Distribution lists supported.

#### **MQDL\_NOT\_SUPPORTED**

Distribution lists not supported.

### **z/OS DNSGroup (MQCFST)**

DNS group name (parameter identifier: MQCA\_DNS\_GROUP).

This parameter is no longer used. See [z/OS: WLM/DNS no longer supported](#).

This parameter is valid only on z/OS.

### **z/OS DNSWLM (MQCFIN)**

WLM/DNS Control: (parameter identifier: MQIA\_DNS\_WLM).

This parameter is no longer used. See [z/OS: WLM/DNS no longer supported](#).

The value can be any of the following values:

#### **MQDNSWLM\_NO**

MQDNSWLM\_NO is the only value supported by the queue manager.

This parameter is valid only on z/OS.

### **EncryptionPolicySuiteB (MQCFIL)**

Specifies whether Suite B-compliant cryptography is used and what level of strength is employed (parameter identifier: MQIA\_SUITE\_B\_STRENGTH). For more information about Suite B configuration and its effect on TLS channels, see [NSA Suite B Cryptography in IBM MQ](#).

The value can be one, or more, of:

#### **MQ\_SUITE\_B\_NONE**

Suite B-compliant cryptography is not used.

**MQ\_SUITE\_B\_128\_BIT**

Suite B 128-bit strength security is used.

**MQ\_SUITE\_B\_192\_BIT**

Suite B 192-bit strength security is used.

**MQ\_SUITE\_B\_128\_BIT, MQ\_SUITE\_B\_192\_BIT**

Suite B 128-bit and Suite B 192-bit strength security is used.

**z/OS** **ExpiryInterval (MQCFIN)**

Interval between scans for expired messages (parameter identifier: MQIA\_EXPIRY\_INTERVAL).

Specifies the frequency with which the queue manager scans the queues looking for expired messages. This parameter is a time interval in seconds in the range 1 through 99 999 999, or the following special value:

**MQEXPI\_OFF**

No scans for expired messages.

This parameter is valid only on z/OS.

**z/OS** **GroupUR (MQCFIN)**

Identifies whether XA client applications can establish transactions with a GROUP unit of recovery disposition.

The value can be:

**MQGUR\_DISABLED**

XA client applications must connect using a queue manager name.

**MQGUR\_ENABLED**

XA client applications can establish transactions with a group unit of recovery disposition by specifying a queue sharing group name when they connect.

This parameter is valid only on z/OS.

**z/OS** **IGQPutAuthority (MQCFIN)**

Type of authority checking used by the intra-group queuing agent (parameter identifier: MQIA\_IGQ\_PUT\_AUTHORITY).

The attribute indicates the type of authority checking that is performed by the local intra-group queuing agent (IGQ agent). The checking is performed when the IGQ agent removes a message from the shared transmission queue and places the message on a local queue. The value can be any of the following values:

**MQIGQPA\_DEFAULT**

Default user identifier is used.

**MQIGQPA\_CONTEXT**

Context user identifier is used.

**MQIGQPA\_ONLY\_IGQ**

Only the IGQ user identifier is used.

**MQIGQPA\_ALTERNATE\_OR\_IGQ**

Alternate user identifier or IGQ-agent user identifier is used.

This parameter is valid only on z/OS.

**z/OS** **IGQUserId (MQCFST)**

User identifier used by the intra-group queuing agent (parameter identifier: MQCA\_IGQ\_USER\_ID).

The maximum length of the string is MQ\_USER\_ID\_LENGTH. This parameter is valid only on z/OS.

**ImageInterval (MQCFIN)**

The target frequency with which the queue manager automatically writes media images (parameter identifier: MQIA\_MEDIA\_IMAGE\_INTERVAL). This parameter is not valid on z/OS.

The value can be:

The time interval, at which the queue manager automatically writes media images.

**MQMEDIMGINTVL\_OFF**

Automatic media images are not written on a time interval basis.

**ImageLogLength (MQCFIN)**

The target size of the recovery log (parameter identifier: MQIA\_MEDIA\_IMAGE\_LOG\_LENGTH). This parameter is not valid on z/OS.

The value can be:

The size of the recovery log.

**MQMEDIMGLOGLN\_OFF**

Automatic media images are not written.

**ImageRecoverObject (MQCFST)**

Specifies the recoverable objects from a media image, if linear logging is being used (parameter identifier: MQIA\_MEDIA\_IMAGE\_RECOVER\_OBJ). This parameter is not valid on z/OS.

The value can be:

**MQIMGRCOV\_NO**

Automatic media images, if enabled, are not written for these objects.

**MQIMGRCOV\_YES**

These objects are recoverable.

**ImageRecoverQueue (MQCFST)**

Displays the default **ImageRecoverQueue** attribute for local and permanent dynamic queue objects, when used with this parameter (parameter identifier: MQIA\_MEDIA\_IMAGE\_RECOVER\_Q). This parameter is not valid on z/OS.

The value can be:

**MQIMGRCOV\_NO**

The **ImageRecoverQueue** attribute for local and permanent dynamic queue objects is set to MQIMGRCOV\_NO .

**MQIMGRCOV\_YES**

The **ImageRecoverQueue** attribute for local and permanent dynamic queue objects is set to MQIMGRCOV\_YES .

**ImageSchedule (MQCFST)**

Whether the queue manager automatically writes media images (parameter identifier: MQIA\_MEDIA\_IMAGE\_SCHEDULING). This parameter is not valid on z/OS.

The value can be:

**MQMEDIMGSCHEM\_AUTO**

The queue manager automatically writes a media image for an object.

**MQMEDIMGSCHEM\_MANUAL**

Automatic media images are not written.

**InhibitEvent (MQCFIN)**

Controls whether inhibit (Inhibit Get and Inhibit Put) events are generated (parameter identifier: MQIA\_INHIBIT\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.



Multi

V 9.3.0

V 9.3.0

### InitialKey (MQCFST)

The initial key for the password protection system (parameter identifier: MQCA\_INITIAL\_KEY).

The length of the string is MQ\_INITIAL\_KEY\_LENGTH. If this attribute is set to a custom value, the value is returned as \*\*\*\*\*.

A blank string is returned when the default initial key is in use.

This parameter is valid only on IBM MQ for Multiplatforms.

z/OS

### IntraGroupQueuing (MQCFIN)

Specifies whether intra-group queuing is used (parameter identifier: MQIA\_INTRA\_GROUP\_QUEUING).

The value can be:

#### MQIGQ\_DISABLED

Intra-group queuing is disabled. All messages destined for other queue managers in the queue sharing group are transmitted using conventional channels.

#### MQIGQ\_ENABLED

Intra-group queuing is enabled.

This parameter is valid only on z/OS.

### IPAddressVersion (MQCFIN)

IP address version selector (parameter identifier: MQIA\_IP\_ADDRESS\_VERSION).

Specifies which IP address version, either IPv4 or IPv6, is used. The value can be:

#### MQIPADDR\_IPV4

IPv4 is used.

#### MQIPADDR\_IPV6

IPv6 is used.

### ListenerTimer (MQCFIN)

Listener restart interval (parameter identifier: MQIA\_LISTENER\_TIMER).

The time interval, in seconds, between attempts by IBM MQ to restart the listener after an APPC or TCP/IP failure.

z/OS

### LocalEvent (MQCFIN)

Controls whether local error events are generated (parameter identifier: MQIA\_LOCAL\_EVENT).

The value can be:

#### MQEVR\_DISABLED

Event reporting disabled.

#### MQEVR\_ENABLED

Event reporting enabled.

This parameter is valid only on z/OS.

### LoggerEvent (MQCFIN)

Controls whether recovery log events are generated (parameter identifier: MQIA\_LOGGER\_EVENT).

The value can be:

#### MQEVR\_DISABLED

Event reporting disabled.

#### MQEVR\_ENABLED

Event reporting enabled.

This parameter applies only to AIX, Linux, and Windows.

z/OS

### LUGroupName (MQCFST)

Generic LU name for the LU 6.2 listener (parameter identifier: MQCA\_LU\_GROUP\_NAME).

The generic LU name to be used by the LU 6.2 listener that handles inbound transmissions for the queue sharing group. This parameter is valid only on z/OS.

**z/OS LUName (MQCFST)**

LU name to use for outbound LU 6.2 transmissions (parameter identifier: MQCA\_LU\_NAME).

The name of the LU to use for outbound LU 6.2 transmissions. This parameter is valid only on z/OS.

**z/OS LU62ARMSuffix (MQCFST)**

APPCPM suffix (parameter identifier: MQCA\_LU62\_ARM\_SUFFIX).

The suffix of the APPCPM member of SYS1.PARMLIB. This suffix nominates the LUADD for this channel initiator. This parameter is valid only on z/OS.

**z/OS LU62Channels (MQCFIN)**

Maximum number of LU 6.2 channels (parameter identifier: MQIA\_LU62\_CHANNELS).

The maximum number of channels that can be current, or clients that can be connected, that use the LU 6.2 transmission protocol. This parameter is valid only on z/OS.

**z/OS MaxActiveChannels (MQCFIN)**

Maximum number of channels (parameter identifier: MQIA\_ACTIVE\_CHANNELS).

The maximum number of channels that can be active at any time. This parameter is valid only on z/OS.

**z/OS MaxChannels (MQCFIN)**

Maximum number of current channels (parameter identifier: MQIA\_MAX\_CHANNELS).

The maximum number of channels that can be current (including server-connection channels with connected clients). This parameter is valid only on z/OS.

**MaxHandles (MQCFIN)**

Maximum number of handles (parameter identifier: MQIA\_MAX\_HANDLES).

Specifies the maximum number of handles that any one connection can have open at the same time.

**MaxMsgLength (MQCFIN)**

Maximum message length (parameter identifier: MQIA\_MAX\_MSG\_LENGTH).

**MaxPriority (MQCFIN)**

Maximum priority (parameter identifier: MQIA\_MAX\_PRIORITY).

**MaxPropertiesLength (MQCFIN)**

Maximum properties length (parameter identifier: MQIA\_MAX\_PROPERTIES\_LENGTH).

**MaxUncommittedMsgs (MQCFIN)**

Maximum number of uncommitted messages within a unit of work (parameter identifier: MQIA\_MAX\_UNCOMMITTED\_MSGS).

This number is the sum of the following number of messages under any one sync point. :

- The number of messages that can be retrieved, plus
- The number of messages that can be put on a queue, plus
- Any trigger messages generated within this unit of work

The limit does not apply to messages that are retrieved or put outside sync point.

**MQIAccounting (MQCFIN)**

Specifies whether accounting information for MQI data is to be collected (parameter identifier: MQIA\_ACCOUNTING\_MQI).

The value can be:

**MQMON\_OFF**

MQI accounting data collection is disabled.

**MQMON\_ON**

MQI accounting data collection is enabled.

This parameter applies only to AIX, Linux, and Windows.

**MQIStatistics (MQCFIN)**

Specifies whether statistics monitoring data is to be collected for the queue manager (parameter identifier: MQIA\_STATISTICS\_MQI).

The value can be:

**MQMON\_OFF**

Data collection for MQI statistics is disabled. MQMON\_OFF is the initial default value of the queue manager.

**MQMON\_ON**

Data collection for MQI statistics is enabled.

This parameter applies only to AIX, Linux, and Windows.

**MsgMarkBrowseInterval (MQCFIN)**

Mark-browse interval (parameter identifier: MQIA\_MSG\_MARK\_BROWSE\_INTERVAL).

The time interval in milliseconds after which the queue manager can automatically unmark messages.



**Attention:** This value should not be below the default of 5000.

▶ **z/OS** **OutboundPortMax (MQCFIN)**

The maximum value in the range for the binding of outgoing channels (parameter identifier: MQIA\_OUTBOUND\_PORT\_MAX).

The maximum value in the range of port numbers to be used when binding outgoing channels. This parameter is valid only on z/OS.

▶ **z/OS** **OutboundPortMin (MQCFIN)**

The minimum value in the range for the binding of outgoing channels (parameter identifier: MQIA\_OUTBOUND\_PORT\_MIN).

The minimum value in the range of port numbers to be used when binding outgoing channels. This parameter is valid only on z/OS.

**Parent (MQCFST)**

The name of the hierarchically connected queue manager nominated as the parent of this queue manager (parameter identifier: MQCA\_PARENT).

**PerformanceEvent (MQCFIN)**

Controls whether performance-related events are generated (parameter identifier: MQIA\_PERFORMANCE\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**Platform (MQCFIN)**

Platform on which the queue manager resides (parameter identifier: MQIA\_PLATFORM).

The value can be:

**MQPL\_AIX**

AIX (same value as MQPL\_UNIX).

**MQPL\_APPLIANCE**

IBM MQ Appliance

**MQPL\_OS400**

IBM i.

**MQPL\_UNIX**

UNIX.

**MQPL\_WINDOWS\_NT**

Windows.

**MQPL\_ZOS**

z/OS

**PubSubClus (MQCFIN)**

Controls whether the queue manager participates in publish/subscribe clustering (parameter identifier: MQIA\_PUBSUB\_CLUSTER).

The value can be:

**MQPSCLUS\_ENABLED**

The creating or receipt of clustered topic definitions and cluster subscriptions is permitted.

**Note:** The introduction of a clustered topic into a large IBM MQ cluster can cause a degradation in performance. This degradation occurs because all partial repositories are notified of all the other members of the cluster. Unexpected subscriptions might be created at all other nodes; for example, where `proxysub(FORCE)` is specified. Large numbers of channels might be started from a queue manager; for example, on resync after a queue manager failure.

**MQPSCLUS\_DISABLED**

The creating or receipt of clustered topic definitions and cluster subscriptions is inhibited. The creations or receipts are recorded as warnings in the queue manager error logs.

**PubSubMaxMsgRetryCount (MQCFIN)**

The number of attempts to reprocess a failed command message under sync point (parameter identifier: MQIA\_PUBSUB\_MAXMSG\_RETRY\_COUNT).

**PubSubMode (MQCFIN)**

Specifies whether the publish/subscribe engine and the queued publish/subscribe interface are running. The publish/subscribe engine enables applications to publish or subscribe by using the application programming interface. The publish/subscribe interface monitors the queues used the queued publish/subscribe interface (parameter identifier: MQIA\_PUBSUB\_MODE).

The values can be as follows:

**MQPSM\_COMPAT**

The publish/subscribe engine is running. It is therefore possible to publish or subscribe by using the application programming interface. The queued publish/subscribe interface is not running. Therefore any message that is put to the queues that are monitored by the queued publish/subscribe interface is not acted on. MQPSM\_COMPAT is used for compatibility with versions of IBM Integration Bus, (formerly known as WebSphere Message Broker) prior to version 7 that use this queue manager.

**MQPSM\_DISABLED**

The publish/subscribe engine and the queued publish/subscribe interface are not running. It is therefore not possible to publish or subscribe by using the application programming interface. Any publish/subscribe messages that are put to the queues that are monitored by the queued publish/subscribe interface are not acted on.

**MQPSM\_ENABLED**

The publish/subscribe engine and the queued publish/subscribe interface are running. It is therefore possible to publish or subscribe by using the application programming interface and the queues that are being monitored by the queued publish/subscribe interface. MQPSM\_ENABLED is the initial default value of the queue manager.

**PubSubNPInputMsg (MQCFIN)**

Specifies whether to discard or keep an undelivered input message (parameter identifier: MQIA\_PUBSUB\_NP\_MSG).

The values can be as follows:

**MQUNDELIVERED\_DISCARD**

Non-persistent input messages can be discarded if they cannot be processed. MQUNDELIVERED\_DISCARD is the default value.

**MQUNDELIVERED\_KEEP**

Non-persistent input messages are not discarded if they cannot be processed. The queued publish/subscribe interface continues to try the process again at appropriate intervals. It does not continue processing subsequent messages.

**PubSubNPResponse (MQCFIN)**

Controls the behavior of undelivered response messages (parameter identifier: MQIA\_PUBSUB\_NP\_RESP).

The values can be as follows:

**MQUNDELIVERED\_NORMAL**

Non-persistent responses that cannot be placed on the reply queue are put on the dead letter queue. If they cannot be placed on the dead letter queue, they are discarded.

**MQUNDELIVERED\_SAFE**

Non-persistent responses that cannot be placed on the reply queue are put on the dead letter queue. If the response cannot be sent and cannot be placed on the dead letter queue the queued publish/subscribe interface rolls back the current operation. The operation is tried again at appropriate intervals and does not continue processing subsequent messages.

**MQUNDELIVERED\_DISCARD**

Non-persistent responses that cannot be placed on the reply queue are discarded. MQUNDELIVERED\_DISCARD is the default value for new queue managers.

**MQUNDELIVERED\_KEEP**

Non-persistent responses are not placed on the dead letter queue or discarded. Instead, the queued publish/subscribe interface backs out the current operation and then tries it again at appropriate intervals.

**PubSubSyncPoint (MQCFIN)**

Specifies whether only persistent messages or all messages are processed under sync point (parameter identifier: MQIA\_PUBSUB\_SYNC\_PT).

The values can be as follows:

**MQSYNCPOINT\_IFPER**

This makes the queued publish/subscribe interface receive non-persistent messages outside sync point. If the daemon receives a publication outside sync point, the daemon forwards the publication to subscribers known to it outside sync point. MQSYNCPOINT\_IFPER is the default value.

**MQSYNCPOINT\_YES**

MQSYNCPOINT\_YES makes the queued publish/subscribe interface receive all messages under sync point.

**QMGrDesc (MQCFST)**

Queue manager description (parameter identifier: MQCA\_Q\_MGR\_DESC).

This parameter is text that briefly describes the object.

The maximum length of the string is MQ\_Q\_MGR\_DESC\_LENGTH.

Use characters from the character set identified by the coded character set identifier (CCSID) for the queue manager on which the command is executing. Using this character set ensures that the text is translated correctly.

**QMGrIdentifier (MQCFST)**

Queue manager identifier (parameter identifier: MQCA\_Q\_MGR\_IDENTIFIER).

The unique identifier of the queue manager.

### **QMgrName (MQCFST)**

Name of local queue manager (parameter identifier: MQCA\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

### **z/OS QSGCertificateLabel (MQCFST)**

Certificate label in the key repository for this queue sharing group to use (parameter identifier: MQCA\_QSG\_CERT\_LABEL).

The maximum length of the string is MQ\_QSG\_CERT\_LABEL\_LENGTH. This parameter is valid only on z/OS.

### **z/OS QSGName (MQCFST)**

Queue sharing group name (parameter identifier: MQCA\_QSG\_NAME).

The maximum length of the string is MQ\_QSG\_NAME\_LENGTH. This parameter is valid only on z/OS.

### **QueueAccounting (MQCFIN)**

Collection of accounting (thread-level and queue-level accounting) data for queues (parameter identifier: MQIA\_ACCOUNTING\_Q).

The value can be:

#### **MQMON\_NONE**

Accounting data collection for queues is disabled.

#### **MQMON\_OFF**

Accounting data collection is disabled for queues specifying a value of MQMON\_Q\_MGR in the **QueueAccounting** parameter.

#### **MQMON\_ON**

Accounting data collection is enabled for queues specifying a value of MQMON\_Q\_MGR in the **QueueAccounting** parameter.

### **QueueMonitoring (MQCFIN)**

Default setting for online monitoring for queues (parameter identifier: MQIA\_MONITORING\_Q).

If the **QueueMonitoring** queue attribute is set to MQMON\_Q\_MGR, this attribute specifies the value which is assumed by the channel. The value can be any of the following values:

#### **MQMON\_OFF**

Online monitoring data collection is turned off.

#### **MQMON\_NONE**

Online monitoring data collection is turned off for queues regardless of the setting of their **QueueMonitoring** attribute.

#### **MQMON\_LOW**

Online monitoring data collection is turned on, with a low ratio of data collection.

#### **MQMON\_MEDIUM**

Online monitoring data collection is turned on, with a moderate ratio of data collection.

#### **MQMON\_HIGH**

Online monitoring data collection is turned on, with a high ratio of data collection.

### **Multi QueueStatistics (MQCFIN)**

Specifies whether statistics data is to be collected for queues (parameter identifier: MQIA\_STATISTICS\_Q).

The value can be:

#### **MQMON\_NONE**

Statistics data collection is turned off for queues regardless of the setting of their **QueueStatistics** parameter.

**MQMON\_OFF**

Statistics data collection is turned off for queues specifying a value of MQMON\_Q\_MGR in their **QueueStatistics** parameter.

**MQMON\_ON**

Statistics data collection is turned on for queues specifying a value of MQMON\_Q\_MGR in their **QueueStatistics** parameter.

This parameter is valid only on [Multiplatforms](#).

**z/OS** **ReceiveTimeout (MQCFIN)**

How long a TCP/IP channel waits to receive data from its partner (parameter identifier: MQIA\_RECEIVE\_TIMEOUT).

The length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state.

This parameter is valid only on z/OS.

**z/OS** **ReceiveTimeoutMin (MQCFIN)**

The minimum length of time that a TCP/IP channel waits to receive data from its partner (parameter identifier: MQIA\_RECEIVE\_TIMEOUT\_MIN).

The minimum length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state. This parameter is valid only on z/OS.

**z/OS** **ReceiveTimeoutType (MQCFIN)**

The qualifier to apply to *ReceiveTimeout* (parameter identifier: MQIA\_RECEIVE\_TIMEOUT\_TYPE).

The qualifier to apply to *ReceiveTimeoutType* to calculate how long a TCP/IP channel waits to receive data from its partner. The wait includes heartbeats. If the wait interval expires the channel returns to the inactive state. This parameter is valid only on z/OS.

The value can be:

**MQRCVTIME\_MULTIPLY**

The *ReceiveTimeout* value is a multiplier to be applied to the negotiated value of *HeartbeatInterval* to determine how long a channel waits.

**MQRCVTIME\_ADD**

*ReceiveTimeout* is a value, in seconds, to be added to the negotiated value of *HeartbeatInterval* to determine how long a channel waits.

**MQRCVTIME\_EQUAL**

*ReceiveTimeout* is a value, in seconds, representing how long a channel waits.

**RemoteEvent (MQCFIN)**

Controls whether remote error events are generated (parameter identifier: MQIA\_REMOTE\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**RepositoryName (MQCFST)**

Repository name (parameter identifier: MQCA\_REPOSITORY\_NAME).

The name of a cluster for which this queue manager is to provide a repository service.

**RepositoryNameList (MQCFST)**

Repository name list (parameter identifier: MQCA\_REPOSITORY\_NAMELIST).

The name of a list of clusters for which this queue manager is to provide a repository service.

### RevDns (MQCFIN)

Whether reverse lookup of the host name from a Domain Name Server is carried out. (parameter identifier: MQIA\_REVERSE\_DNS\_LOOKUP).

This attribute has an effect only on channels using a transport type (TRPTYPE) of TCP.

The value can be:

#### MQRDNS\_DISABLED

DNS host names are not reverse looked-up for the IP addresses of inbound channels. With this setting any CHLAUTH rules using host names are not matched.

#### MQRDNS\_ENABLED

DNS host names are reverse looked-up for the IP addresses of inbound channels when this information is required. This setting is required for matching against CHLAUTH rules that contain host names, and for writing out error messages.

### SecurityCase (MQCFIN)

Security case supported (parameter identifier: MQIA\_SECURITY\_CASE).

Specifies whether the queue manager supports security profile names in mixed case, or in uppercase only. The value is activated when a Refresh Security command is run with *SecurityType* (MQSECTYPE\_CLASSES) specified.

The value can be:

#### MQSCYC\_UPPER

Security profile names must be in uppercase.

#### MQSCYC\_MIXED

Security profile names can be in uppercase or in mixed case.

This parameter is valid only on z/OS.

### SharedQQmgrName (MQCFIN)

Shared-queue queue manager name (parameter identifier: MQIA\_SHARED\_Q\_Q\_MGR\_NAME).

A queue manager makes an MQOPEN call for a shared queue. The queue manager that is specified in the **ObjectQmgrName** parameter of the MQOPEN call is in the same queue sharing group as the processing queue manager. The SQQMNAME attribute specifies whether the *ObjectQmgrName* is used or whether the processing queue manager opens the shared queue directly.

The value can be any of the following values:

#### MQSQQM\_USE

*ObjectQmgrName* is used and the appropriate transmission queue is opened.

#### MQSQQM\_IGNORE

The processing queue manager opens the shared queue directly.

This parameter is valid only on z/OS.

### Splcap (MQCFIN)

Specifies whether the Advanced Message Security component is installed for the version of IBM MQ that the queue manager is running under (parameter identifier: MQIA\_PROT\_POLICY\_CAPABILITY).

The value can be one of the following values:

#### MQCAP\_SUPPORTED

If the AMS component is installed for the version of IBM MQ that the queue manager is running under.

#### MQCAP\_NOT\_SUPPORTED

If the AMS component is not installed.

### SSLCRLNamelist (MQCFST)

The TLS certificate revocation location namelist (parameter identifier: MQCA\_SSL\_CRL\_NAMELIST).

The length of the string is MQ\_NAMELIST\_NAME\_LENGTH.



Indicates the name of a namelist of authentication information objects to be used for certificate revocation checking by the queue manager.

Only authentication information objects with types of CRLLDAP or OCSP are allowed in the namelist referred to by *SSLCRLNamelist* (MQCFST). Any other type results in an error message when the list is processed and is subsequently ignored.

### **Multi** **SSLCryptoHardware (MQCFST)**

Parameters to configure the TLS cryptographic hardware (parameter identifier: MQCA\_SSL\_CRYPTO\_HARDWARE).

The length of the string is MQ\_SSL\_CRYPTO\_HARDWARE\_LENGTH.

Sets the name of the parameter string required to configure the cryptographic hardware present on the system.

This parameter is valid only on [Multiplatforms](#).

### **SSLEvent (MQCFIN)**

Controls whether TLS events are generated (parameter identifier: MQIA\_SSL\_EVENT).

The value can be:

#### **MQEVR\_DISABLED**

Event reporting disabled.

#### **MQEVR\_ENABLED**

Event reporting enabled.

### **SSLFipsRequired (MQCFIN)**

Controls whether only FIPS-certified algorithms are to be used if cryptography is executed in IBM MQ itself (parameter identifier: MQIA\_SSL\_FIPS\_REQUIRED). This parameter is valid only on z/OS, AIX, Linux, and Windows.

The value can be:

#### **MQSSL\_FIPS\_NO**

Any supported CipherSpec can be used.

#### **MQSSL\_FIPS\_YES**

Only FIPS-certified cryptographic algorithms are to be used if cryptography is executed in IBM MQ rather than cryptographic hardware.

### **SSLKeyRepository (MQCFST)**

Location and name of the TLS key repository (parameter identifier: MQCA\_SSL\_KEY\_REPOSITORY).

The length of the string is MQ\_SSL\_KEY\_REPOSITORY\_LENGTH.

Indicates the name of the Secure Sockets Layer key repository.

The format of the name depends on the environment.

V 9.3.0

ALW

V 9.3.0

### **SSLKeyRepositoryPassword (MQCFST)**

The password to access the TLS key repository (parameter identifier: MQCA\_SSL\_KEY\_REPO\_PASSWORD).

The length of the string is MQ\_SSL\_ENCRYPT\_KEY\_REPO\_PWD\_LEN.

If a value is set for this attribute, it is always returned as \*\*\*\*\*.

This parameter is valid only on AIX, Linux, and Windows.

### **SSLKeyResetCount (MQCFIN)**

TLS key reset count (parameter identifier: MQIA\_SSL\_RESET\_COUNT).

The number of unencrypted bytes that initiating TLS channel MCAs send or receive before renegotiating the secret key.

**z/OS SSLTasks (MQCFIN)**

Number of server subtasks used for processing TLS calls (parameter identifier: MQIA\_SSL\_TASKS).

The number of server subtasks used for processing TLS calls. This parameter is valid only on z/OS.

**StartStopEvent (MQCFIN)**

Controls whether start and stop events are generated (parameter identifier: MQIA\_START\_STOP\_EVENT).

The value can be:

**MQEVR\_DISABLED**

Event reporting disabled.

**MQEVR\_ENABLED**

Event reporting enabled.

**Multi StatisticsInterval (MQCFIN)**

The time interval, in seconds, at which statistics monitoring data is written to the monitoring queue (parameter identifier: MQIA\_STATISTICS\_INTERVAL).

This parameter is valid only on [Multiplatforms](#).

**SyncPoint (MQCFIN)**

Sync point availability (parameter identifier: MQIA\_SYNCPOINT).

The value can be:

**MQSP\_AVAILABLE**

Units of work and sync pointing available.

**MQSP\_NOT\_AVAILABLE**

Units of work and sync pointing not available.

**z/OS TCPChannels (MQCFIN)**

The maximum number of channels that can be current, or clients that can be connected, that use the TCP/IP transmission protocol (parameter identifier: MQIA\_TCP\_CHANNELS).

This parameter is valid only on z/OS.

**z/OS TCPKeepAlive (MQCFIN)**

Specifies whether the TCP KEEPALIVE facility is to be used to check whether the other end of the connection is still available (parameter identifier: MQIA\_TCP\_KEEP\_ALIVE).

The value can be:

**MQTCPKEEP\_YES**

The TCP KEEPALIVE facility is to be used as specified in the TCP profile configuration data set. The interval is specified in the *KeepAliveInterval* channel attribute.

**MQTCPKEEP\_NO**

The TCP KEEPALIVE facility is not to be used.

This parameter is valid only on z/OS.

**z/OS TCPName (MQCFST)**

The name of the TCP/IP system that you are using (parameter identifier: MQIA\_TCP\_NAME).

This parameter is valid only on z/OS.

**TCPStackType (MQCFIN)**

Specifies whether the channel initiator can use only the TCP/IP address space specified in *TCPName*, or can optionally bind to any selected TCP/IP address (parameter identifier: MQIA\_TCP\_STACK\_TYPE).

The value can be:

**MQTCPSTACK\_SINGLE**

The channel initiator can use only the TCP/IP address space specified in *TCPName*.

**MQTCPSTACK\_MULTIPLE**

The channel initiator can use any TCP/IP address space available to it.

This parameter is valid only on z/OS.

**TraceRouteRecording (MQCFIN)**

Specifies whether trace-route information can be recorded and a reply message generated (parameter identifier: MQIA\_TRACE\_ROUTE\_RECORDING).

The value can be:

**MQRECORDING\_DISABLED**

Trace-route information cannot be recorded.

**MQRECORDING\_MSG**

Trace-route information can be recorded and sent to the destination specified by the originator of the message causing the trace route record.

**MQRECORDING\_Q**

Trace-route information can be recorded and sent to SYSTEM.ADMIN.TRACE.ROUTE.QUEUE.

**TreeLifeTime (MQCFIN)**

The lifetime in seconds of non-administrative topics (parameter identifier: MQIA\_TREE\_LIFE\_TIME).

Non-administrative topics are those topics created when an application publishes to, or subscribes on, a topic string that does not exist as an administrative node. When this non-administrative node no longer has any active subscriptions, this parameter determines how long the queue manager waits before removing that node. Only non-administrative topics that are in use by a durable subscription remain after the queue manager it recycled.

The value can be in the range 0 - 604,000. A value of 0 means that non-administrative topics are not removed by the queue manager. The initial default value of the queue manager is 1800.

**TriggerInterval (MQCFIN)**

Trigger interval (parameter identifier: MQIA\_TRIGGER\_INTERVAL).

Specifies the trigger time interval, expressed in milliseconds, for use only with queues where *TriggerType* has a value of MQTT\_FIRST.

**Version (MQCFST)**

The version of the IBM MQ code (parameter identifier: MQCA\_VERSION).

The version of the IBM MQ code is shown as VVRRMMFF:

VV: Version

RR: Release

MM: Maintenance level

FF: Fix level

 **XrCapability (MQCFIN)**

Specifies whether the MQ Telemetry capability and commands are supported by the queue manager where *XrCapability* has a value of MQCAP\_SUPPORTED or MQCAP\_NOT\_SUPPORTED (parameter identifier: MQIA\_XR\_CAPABILITY).

This parameter applies only to Multiplatforms.

**Related tasks**

[Specifying that only FIPS-certified CipherSpecs are used at run time on the MQI client](#)

**Related reference**

[Federal Information Processing Standards \(FIPS\) for AIX, Linux, and Windows](#)

## **MQCMD\_INQUIRE\_Q\_MGR\_STATUS (Inquire Queue Manager Status) on Multiplatforms**

The Inquire Queue Manager Status (MQCMD\_INQUIRE\_Q\_MGR\_STATUS) PCF command inquires about the status of the local queue manager.

### **Optional parameters**

#### **IntegerFilterCommand**

Integer filter command descriptor. The parameter identifier must be one of the following Native HA integer type parameters:

- MQIACF\_NHA\_INSTANCE\_ROLE
- MQIACF\_NHA\_INSTANCE\_BACKLOG
- MQIACF\_NHA\_INSTANCE\_ACTV\_CONNS
- MQIACF\_NHA\_INSTANCE\_IN\_SYNC

Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

#### **QMStatusAttrs (MQCFIL)**

Queue manager status attributes (parameter identifier: MQIACF\_Q\_MGR\_STATUS\_ATTRS).

The attribute list might specify the following value on its own and is the default value used if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes.

A combination of the following attributes returns status information for a queue manager:

Where **StatusType** is MQIACF\_Q\_MGR\_STATUS\_INFO\_Q\_MGR:

#### **MQCA\_Q\_MGR\_NAME**

Name of the local queue manager.

#### **MQCA\_INSTALLATION\_DESC**

Description of the installation associated with the queue manager.

#### **MQCA\_INSTALLATION\_NAME**

Name of the installation associated with the queue manager.

#### **MQCA\_INSTALLATION\_PATH**

Path of the installation associated with the queue manager.

#### **MQCACF\_ARCHIVE\_LOG\_EXTENT\_NAME)**

Name of the oldest log extent for which the queue manager is waiting for archive notification.

The maximum length of the string is MQ\_LOG\_EXTENT\_NAME\_LENGTH.

If the queue manager is not using archive log management, this attribute is blank. This parameter is not valid on IBM i.

#### **MQCACF\_CURRENT\_LOG\_EXTENT\_NAME**

Name of the log extent currently being written to by the logger.

MQCACF\_CURRENT\_LOG\_EXTENT\_NAME is available only on queue managers using linear logging. On other queue managers, MQCACF\_CURRENT\_LOG\_EXTENT\_NAME is blank.

#### **V9.3.2 MQCACF\_HOST\_NAME**

The name of the host that the queue manager is running on. The length of this attribute is given by MQ\_HOST\_NAME\_LENGTH.

#### **MQCACF\_LOG\_PATH**

Location of the recovery log extents.

### **V 9.3.2 MQCACF\_LOG\_START\_DATE**

The date for the last log record recovered when the queue manager started (in the form *yyyy-mm-dd*). The length of this attribute is given by `MQ_DATE_LENGTH`.

This attribute is not valid on IBM i.

### **V 9.3.2 MQCACF\_LOG\_START\_LSN**

The log sequence number (LSN) for the last log record recovered when the queue manager started. Given in the form *<nnnnn : nnnnn : nnnnn : nnnnn>*. The length of this attribute is given by `MQ_LSN_LENGTH`.

This attribute is not valid on IBM i.

### **V 9.3.2 MQCACF\_LOG\_START\_TIME**

The time for the last log record recovered when the queue manager started (in the form *hh.mm.ss*). The length of this attribute is given by `MQ_TIME_LENGTH`.

This attribute is not valid on IBM i.

### **MQCACF\_MEDIA\_LOG\_EXTENT\_NAME**

Name of the earliest log extent required to perform media recovery.

`MQCACF_MEDIA_LOG_EXTENT_NAME` is available only on queue managers using linear logging.

On other queue managers, `MQCACF_MEDIA_LOG_EXTENT_NAME` is blank.

### **V 9.3.2 MQCACF\_NHA\_INSTANCE\_NAME**

The name of the local Native HA instance if the queue manager is running in a Native HA environment, or blank otherwise. The length of this attribute is given by `MQ_NHA_INSTANCE_NAME_LENGTH`.

### **MQCACF\_RESTART\_LOG\_EXTENT\_NAME**

Name of the earliest log extent required to perform restart recovery.

`MQCACF_RESTART_LOG_EXTENT_NAME` is available only on queue managers using linear logging.

On other queue managers, `MQCACF_RESTART_LOG_EXTENT_NAME` is blank.

### **V 9.3.2 MQCACF\_Q\_MGR\_DATA\_PATH**

Location of the queue manager data. This parameter identifies the directory where queue manager data files are created by the queue manager. The length of this attribute is given by `MQ_Q_MGR_DATA_PATH_LENGTH`.

### **MQCACF\_Q\_MGR\_START\_DATE**

The date on which the queue manager was started (in the form *yyyy-mm-dd*). The length of this attribute is given by `MQ_DATE_LENGTH`.

### **MQCACF\_Q\_MGR\_START\_TIME**

The time at which the queue manager was started (in the form *hh.mm.ss*). The length of this attribute is given by `MQ_TIME_LENGTH`.

### **V 9.3.2 MQCACF\_UNIFORM\_CLUSTER\_NAME**

Indicates the name of the uniform cluster that the queue manager is a member of, or blank otherwise. The length of this attribute is given by `MQ_CLUSTER_NAME_LENGTH`.

### **MQIACF\_ARCHIVE\_LOG\_SIZE**

Current size of the amount of space occupied, in megabytes, by log extents no longer required for restart or media recovery but waiting to be archived.

This attribute is not valid on IBM i.

### **V 9.3.2 MQIACF\_AUTO\_CLUSTER\_TYPE**

The type of automatic cluster that the queue manager is a member of.

### **MQIACF\_CHINIT\_STATUS**

Current status of the channel initiator.

**MQIACF\_CMD\_SERVER\_STATUS**

Current status of the command server.

**MQIACF\_CONNECTION\_COUNT**

Current number of connections to the queue manager.

**V 9.3.2 MQIACF\_DATA\_FS\_SIZE**

The size of the dedicated queue manager data file system in MB, rounded up.

**V 9.3.2 MQIACF\_DATA\_FS\_IN\_USE**

The percentage of the dedicated queue manager data file system that is used, rounded up to the nearest percent.

**MQIACF\_LDAP\_CONNECTION\_STATUS**

Current status of the connection to the LDAP server.

**V 9.3.2 MQIACF\_LOG\_EXTENT\_SIZE**

The size of each log file in KB or the threshold of the currently attached journal receiver on IBM i.

**V 9.3.2 MQIACF\_LOG\_FS\_SIZE**

The size of the dedicated recovery log file system in MB, rounded up. The amount of file system space is different from the amount of the active log being used and is based on the configuration of the log, the space may not be useable.

**V 9.3.2 MQIACF\_LOG\_FS\_IN\_USE**

The percentage of the recovery log file system that is used, rounded up to the nearest percent.

**MQIACF\_LOG\_IN\_USE**

Current size of the percentage of the primary log space in use for restart recovery at this point in time.

This attribute is not valid on IBM i.

**V 9.3.2 MQIACF\_LOG\_PRIMARYES**

The number of primary log files.

This attribute is not valid on IBM i.

**V 9.3.2 MQIACF\_LOG\_SECONDARIES**

The maximum number of secondary log files.

This attribute is not valid on IBM i.

**V 9.3.2 MQIACF\_LOG\_TYPE**

The type of logging used by the queue manager.

**MQIACF\_LOG\_UTILIZATION**

Current percentage estimate of how well the queue manager workload is contained within the primary log space.

This attribute is not valid on IBM i.

**MQIACF\_MEDIA\_LOG\_SIZE**

Current size of the log data required for media recovery in megabytes.

This attribute is not valid on IBM i.

**MQIACF\_NHA\_IN\_SYNC\_INSTANCES**

How many of the configured instances are in-sync with the active instance in a Native HA configuration.

**MQIACF\_NHA\_TOTAL\_INSTANCES**

Total number of configured instances of the queue manager in a Native HA configuration.

**MQIACF\_PERMIT\_STANDBY**

Whether a standby instance is permitted.

### **V 9.3.2 MQIACF\_Q\_MGR\_FS\_ENCRYPTED**

Indicates whether the queue manager file system is encrypted or not.

**MQ Appliance** Returned on the IBM MQ Appliance only.

### **V 9.3.2 MQIACF\_Q\_MGR\_FS\_SIZE**

The size of the queue manager data and recovery log file systems in MB, rounded up.

### **V 9.3.2 MQIACF\_Q\_MGR\_FS\_IN\_USE**

The percentage of the queue manager data and recovery log file systems that are used, rounded up to the nearest percent.

### **MQIACF\_Q\_MGR\_STATUS**

Current status of the queue manager.

### **MQIACF\_Q\_MGR\_STATUS\_LOG**

Current status of all the log attributes. The attributes can be any of the following:

- MQCACF\_ARCHIVE\_LOG\_EXTENT\_NAME
- MQIACF\_ARCHIVE\_LOG\_SIZE
- MQCACF\_CURRENT\_LOG\_EXTENT\_NAME
- MQIACF\_LOG\_IN\_USE
- MQIACF\_LOG\_UTILIZATION
- MQCACF\_MEDIA\_LOG\_EXTENT\_NAME
- MQIACF\_MEDIA\_LOG\_SIZE
- MQCACF\_RESTART\_LOG\_EXTENT\_NAME
- MQIACF\_RESTART\_LOG\_SIZE
- MQIACF\_REUSABLE\_LOG\_SIZE

### **MQIACF\_RESTART\_LOG\_SIZE**

Size of the log data required for restart recovery in megabytes.

This attribute is not valid on IBM i.

### **MQIACF\_REUSABLE\_LOG\_SIZE**

The amount of space occupied, in megabytes, by log extents available to be reused.

This attribute is not valid on IBM i.

**V 9.3.2** Where **StatusType** is MQIACF\_Q\_MGR\_STATUS\_INFO\_NHA: **V 9.3.2**

### **MQCACF\_NHA\_INSTANCE\_NAME**

The name of the instance. The length of this attribute is given by MQ\_NHA\_INSTANCE\_NAME\_LENGTH.

### **MQIACF\_NHA\_INSTANCE\_ROLE**

The current role of the instance in the Native HA group.

### **MQIACF\_NHA\_INSTANCE\_BACKLOG**

How many KB of recovery log data the active instance has written that have not yet been acknowledged by the named instance.

### **MQIACF\_NHA\_INSTANCE\_ACTV\_CONNS**

Whether the instance currently has a pair of active connections to the active instance.

### **MQCACF\_NHA\_GROUP\_INITIAL\_DATE**

The date of the last log record recovered when the Native HA group initially became active. The length of this attribute is given by MQ\_DATE\_LENGTH.

### **MQCACF\_NHA\_GROUP\_INITIAL\_LSN**

The log sequence number (LSN) of the last log record recovered when the Native HA group initially became active. The length of this attribute is given by MQ\_LSN\_LENGTH.

**MQCACF\_NHA\_GROUP\_INITIAL\_TIME**

The time of the last log record recovered when the Native HA group initially became active. The length of this attribute is given by MQ\_TIME\_LENGTH.

**MQIACF\_NHA\_INSTANCE\_IN\_SYNC**

Whether this instance is currently considered in-sync with the active instance.

**MQCACF\_NHA\_REPL\_ADDRESS**

The network address and port to use when sending data to and from the specified instance. The length of this attribute is given by MQ\_NHA\_REPL\_ADDRESS\_LENGTH.


**StatusType (MQCFIN)**

Queue manager status type (parameter identifier: MQIACF\_Q\_MGR\_STATUS\_INFO\_TYPE). Specifies the type of status information required. Can be any of the following values:

**MQIACF\_Q\_MGR\_STATUS\_INFO\_Q\_MGR**

Selects general status information relating to the queue manager.

**MQIACF\_Q\_MGR\_STATUS\_INFO\_NHA**

Selects status information relating to Native HA instances.

The default value, if this parameter is not specified, is MQIACF\_Q\_MGR\_STATUS\_INFO\_Q\_MGR.

You cannot use **StatusType** as a parameter to filter on.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be one of the following Native HA string type parameters:

- MQCACF\_NHA\_INSTANCE\_NAME
- MQCACF\_NHA\_GROUP\_INITIAL\_DATE
- MQCACF\_NHA\_GROUP\_INITIAL\_LSN
- MQCACF\_NHA\_GROUP\_INITIAL\_TIME
- MQCACF\_NHA\_REPL\_ADDRESS

Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

**Error codes**


This command might return the following error code in the response format header [“Error codes applicable to all commands” on page 990](#)

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_NHA\_NOT\_AVAILABLE**

Native HA status not available.

 **MQCMD\_INQUIRE\_Q\_MGR\_STATUS (Inquire Queue Manager Status)**
**Response on Multiplatforms**

The response to the Inquire Queue Manager Status (MQCMD\_INQUIRE\_Q\_MGR\_STATUS) PCF command consists of the response header followed by a set of attribute parameter structures determined by the value of **StatusType** in the Inquire command.

**Always returned:**
 *StatusType*



Possible values of *StatusType* are:

**MQIACF\_Q\_MGR\_STATUS\_INFO\_Q\_MGR**

Returns general status information relating to the queue manager.

**MQIACF\_Q\_MGR\_STATUS\_INFO\_NHA**

Returns status information relating to Native HA instances.

*QMgrName*, *QMgrStatus* (if *StatusType* is MQIACF\_Q\_MGR\_STATUS\_INFO\_Q\_MGR)

**V 9.3.2** *Instance*, *Role* (if *StatusType* is MQIACF\_Q\_MGR\_STATUS\_INFO\_NHA)

**Returned if requested and StatusType is MQIACF\_Q\_MGR\_STATUS\_INFO\_Q\_MGR:**

*ArchiveLog*, *ArchiveLogSize*, *AutoCluster*, *ChannelInitiatorStatus*, *CommandServerStatus*, *ConnectionCount*, *CurrentLog*, *DataFSSize*, *DataFSUse*, *DataPath*, *HostName*, *InstallationDesc*, *InstallationName*, *InstallationPath*, *InSyncInstances*, *LDAPConnectionStatus*, *LogExtSize*, *LogFSSize*, *LogFSUse*, *LogInUse*, *LogPath*, *LogPrim*, *LogSec*, *LogStartDate*, *LogStartLSN*, *LogStartTime*, *LogType*, *LogUtilization*, *MediaRecoveryLog*, *MediaRecoveryLogSize*, *NativeHAInstanceName*, *PermitStandby*, *QMgrEncryption*, *QMgrFSSize*, *QMgrFSUse*, *RestartRecoveryLogSize*, *ReusableLogSize*, *StartDate*, *StartTime*, *TotalInstances*, *UniClusterName*

**V 9.3.2** **Returned if requested and StatusType is MQIACF\_Q\_MGR\_STATUS\_INFO\_NHA:**

*Backlog*, *ConnActv*, *HAInitDate*, *HAInitLSN*, *HAInitTime*, *InSync*, *Instance*, *ReplAddr*, *Role*

**Response data (if StatusType is MQIACF\_Q\_MGR\_STATUS\_INFO\_Q\_MGR)**

**ArchiveLog (MQCFST)**

Name of the oldest log extent for which the queue manager is waiting for archive notification or blank if they have all been archived (parameter identifier MQCACF\_ARCHIVE\_LOG\_EXTENT\_NAME).

**ArchiveLogSize (MQCFIN)**

Current size of the amount of space occupied, in megabytes, by log extents no longer required for restart or media recovery but waiting to be archived (parameter identifier MQIACF\_ARCHIVE\_LOG\_SIZE).

**V 9.3.2** **AutoCluster (MQCFIN)**

Whether the queue manager is a member of an automatic cluster (parameter identifier MQIACF\_AUTO\_CLUSTER\_TYPE).

The value can be:

**MQAUTOCLUS\_TYPE\_NONE**

The queue manager is not a member of an automatic cluster.

**MQAUTOCLUS\_TYPE\_UNIFORM**

The queue manager is a member of an automatic uniform cluster.

See [Using automatic cluster configuration](#)

**ChannelInitiatorStatus (MQCFIN)**

Status of the channel initiator reading SYSTEM.CHANNEL.INITQ (parameter identifier: MQIACF\_CHINIT\_STATUS).

The value can be:

**MQSVC\_STATUS\_STOPPED**

The channel initiator is not running.

**MQSVC\_STATUS\_STARTING**

The channel initiator is in the process of initializing.

**MQSVC\_STATUS\_RUNNING**

The channel initiator is fully initialized and is running.

**MQSVC\_STATUS\_STOPPING**

The channel initiator is stopping.

**CommandServerStatus (MQCFIN)**

Status of the command server (parameter identifier: MQIACF\_CMD\_SERVER\_STATUS).

The value can be:

**MQSVC\_STATUS\_STARTING**

The command server is in the process of initializing.

**MQSVC\_STATUS\_RUNNING**

The command server is fully initialized and is running.

**MQSVC\_STATUS\_STOPPING**

The command server is stopping.

**ConnectionCount (MQCFIN)**

Connection count (parameter identifier: MQIACF\_CONNECTION\_COUNT).

The current number of connections to the queue manager.

**CurrentLog (MQCFST)**

Log extent name (parameter identifier: MQCACF\_CURRENT\_LOG\_EXTENT\_NAME).

The name of the log extent that was being written to at the time of the Inquire command. If the queue manager is using circular logging, this parameter is blank.

The maximum length of the string is MQ\_LOG\_EXTENT\_NAME\_LENGTH.

**V 9.3.2 DataFSSize (MQCFIN)**

The size of the dedicated queue manager data file system in MB, rounded up. If the queue manager data and recovery log are on the same file system, the value is MQFS\_SHARED (parameter identifier: MQIACF\_DATA\_FS\_SIZE).

**V 9.3.2 DataFSUse (MQCFIN)**

The percentage of the queue manager data file system that is used, rounded up to the nearest percent. If the queue manager data and recovery log are on the same file system, the value is MQFS\_SHARED (parameter identifier: MQIACF\_DATA\_FS\_IN\_USE).

**V 9.3.2 DataPath (MQCFST)**

Location of the queue manager data. This parameter identifies the directory where queue manager data files are created by the queue manager (parameter identifier: MQCACF\_Q\_MGR\_DATA\_PATH). The length of this attribute is given by MQ\_Q\_MGR\_DATA\_PATH\_LENGTH.

**V 9.3.2 HostName (MQCFST)**

The name of the host that the queue manager is running on. Usually, this is the value reported by the O/S but it can be overridden by the MQS\_IPC\_HOST environment variable (parameter identifier: MQCACF\_HOST\_NAME). The length of this attribute is given by MQ\_HOST\_NAME\_LENGTH.

**InstallationDesc (MQCFST)**

Installation Description (parameter identifier: MQCA\_INSTALLATION\_DESC).

The installation description for this queue manager.

**InstallationName (MQCFST)**

Installation Name (parameter identifier: MQCA\_INSTALLATION\_NAME).

The installation name for this queue manager.

**InstallationPath (MQCFST)**

Installation Path (parameter identifier: MQCA\_INSTALLATION\_PATH).

The installation path for this queue manager.

### **V 9.3.2 InSyncInstances (MQCFIN)**

How many of the configured instances are in-sync with the active instance (parameter identifier: MQIACF\_NHA\_IN\_SYNC\_INSTANCES).

### **LDAPConnectionStatus (MQCFIN)**

Current status of the queue manager's connection to the LDAP server (parameter identifier: MQIACF\_LDAP\_CONNECTION\_STATUS).

The value can be:

#### **MQLDAPC\_CONNECTED**

The queue manager currently has a connection to the LDAP server.

#### **MQLDAPC\_ERROR**

The queue manager attempted to make a connection to the LDAP server and failed.

#### **MQLDAPC\_INACTIVE**

The queue manager is not configured to use an LDAP server or has not yet made a connection to the LDAP server.

### **V 9.3.2 LogExtSize (MQCFIN)**

The size of each log file in kilobytes or the threshold of the currently attached journal receiver on IBM i (parameter identifier: MQIACF\_LOG\_EXTENT\_SIZE).

### **V 9.3.2 LogFSSize (MQCFIN)**

The size of the dedicated recovery log file system in MB, rounded up. If the queue manager data and recovery log are on the same file system, the value is MQFS\_SHARED (parameter identifier: MQIACF\_LOG\_FS\_SIZE).

### **V 9.3.2 LogFSUse (MQCFIN)**

The percentage of the recovery log file system that is used, rounded up to the nearest percent. If the queue manager data and recovery log are on the same file system, the value is MQFS\_SHARED (parameter identifier: MQIACF\_LOG\_FS\_IN\_USE).

### **LogInUse (MQCFIN)**

Current size of the percentage of the primary log space in use for restart recovery at this point in time (parameter identifier MQIACF\_LOG\_IN\_USE).

### **LogPath (MQCFST)**

Location of the recovery log extents (parameter identifier: MQCACF\_LOG\_PATH).

This parameter identifies the directory where log files are created by the queue manager. The maximum length of the string is MQ\_LOG\_PATH\_LENGTH.

### **V 9.3.2 LogPrim (MQCFIN)**

The number of primary log files (parameter identifier MQIACF\_LOG\_PRIMARIES).

### **V 9.3.2 LogSec (MQCFIN)**

The maximum number of secondary log files (parameter identifier MQIACF\_LOG\_SECONDARIES).

### **V 9.3.2 LogStartDate (MQCFST)**

The date for the last log record recovered when the queue manager started (in the form *yyyy-mm-dd*) (parameter identifier MQCACF\_LOG\_START\_DATE). The length of this attribute is given by MQ\_DATE\_LENGTH.

### **V 9.3.2 LogStartLSN (MQCFST)**

The log sequence number (LSN) for the last log record recovered when the queue manager started (in the form *<nnnnn:nnnnn:nnnnn:nnnnn>*) (parameter identifier MQCACF\_LOG\_START\_LSN). The length of this attribute is given by MQ\_LSN\_LENGTH.

### **V 9.3.2 LogStartTime (MQCFST)**

The time for the last log record recovered when the queue manager started (in the form *hh.mm.ss*) (parameter identifier MQCACF\_LOG\_START\_TIME). The length of this attribute is given by MQ\_TIME\_LENGTH.

### **V 9.3.2 LogType (MQCFIN)**

The type of logging used by the queue manager (parameter identifier MQIACF\_LOG\_TYPE).

The value can be:

#### **MQLOGTYPE\_CIRCULAR**

Circular logging.

#### **MQLOGTYPE\_LINEAR**

Linear logging.

#### **MQLOGTYPE\_REPLICATED**

Replicated logging.

For information about logging types, see [Types of logging](#).

### **LogUtilization (MQCFIN)**

Current percentage estimate of how well the queue manager workload is contained within the primary log space (parameter identifier MQIACF\_LOG\_UTILIZATION).

### **MediaRecoveryLog (MQCFST)**

Name of the oldest log extent required by the queue manager to perform media recovery (parameter identifier: MQCACF\_MEDIA\_LOG\_EXTENT\_NAME). This parameter is available only on queue managers using linear logging. If the queue manager is using circular logging, this parameter is blank.

The maximum length of the string is MQ\_LOG\_EXTENT\_NAME\_LENGTH.

### **MediaRecoveryLogSize (MQCFIN)**

Current size of the log data required for media recovery in megabytes (parameter identifier MQIACF\_MEDIA\_LOG\_SIZE).

### **V 9.3.2 NativeHAInstanceName (MQCFST)**

The name of the local Native HA instance when the queue manager is part of a Native HA group. It is blank otherwise (parameter identifier: MQCACF\_NHA\_INSTANCE\_NAME). The length of this attribute is given by MQ\_NHA\_INSTANCE\_NAME\_LENGTH.

### **PermitStandby (MQCFIN)**

Whether a standby instance is permitted (parameter identifier: MQIACF\_PERMIT\_STANDBY).

The value can be:

#### **MQSTDBY\_NOT\_PERMITTED**

Standby instances are not permitted.

#### **MQSTDBY\_PERMITTED**

Standby instances are permitted.

### **QMgrName (MQCFST)**

Name of the local queue manager (parameter identifier: MQCA\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

### **QMgrStatus (MQCFIN)**

Current execution status of the queue manager (parameter identifier: MQIACF\_Q\_MGR\_STATUS).

The value can be:

#### **MQQMSTA\_STARTING**

The queue manager is initializing.

#### **MQQMSTA\_RUNNING**

The queue manager is fully initialized and is running.

## **MQQMSTA QUIESCING**

The queue manager is quiescing.

### **V 9.3.2 QMgrEncryption (MQCFIN)**

Indicates whether the queue manager file system is encrypted or not (parameter identifier: MQIACF\_Q\_MGR\_FS\_ENCRYPTED).

The value can be:

#### **MQFSENC\_NO**

File system is encrypted.

#### **MQFSENC\_YES**

File system not encrypted.

#### **MQFSENC\_UNKNOWN**

Not known if file system is encrypted.

Returned on the IBM MQ Appliance only.

### **V 9.3.2 QMgrFSSize (MQCFIN)**

The size of the queue manager data and recovery log file systems in MB, rounded up. If the queue manager data and recovery log are on the same file system, the value will be the usage of that file system (parameter identifier: MQIACF\_Q\_MGR\_FS\_SIZE).

### **V 9.3.2 QMgrFSUse (MQCFIN)**

The percentage of the queue manager data and recovery log file systems that are full, rounded up to the nearest percent. If the queue manager data and recovery log are on the same file system, the value will be the usage of that file system. (parameter identifier: MQIACF\_Q\_MGR\_FS\_IN\_USE).

### **RestartRecoveryLog (MQCFST)**

Name of the oldest log extent required by the queue manager to perform restart recovery (parameter identifier: MQCACF\_RESTART\_LOG\_EXTENT\_NAME).

This parameter is available only on queue managers using linear logging. If the queue manager is using circular logging, this parameter is blank.

The maximum length of the string is MQ\_LOG\_EXTENT\_NAME\_LENGTH.

### **RestartRecoveryLogSize (MQCFIN)**

Size of the log data required for restart recovery in megabytes (parameter identifier: MQIACF\_RESTART\_LOG\_SIZE).

### **ReusableLogSize (MQCFIN)**

The amount of space occupied, in megabytes, by log extents available to be reused (parameter identifier: MQIACF\_REUSABLE\_LOG\_SIZE).

### **StartDate (MQCFST)**

Date when this queue manager was started (in the form yyyy-mm-dd) (parameter identifier: MQCACF\_Q\_MGR\_START\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

### **StartTime (MQCFST)**

Time when this queue manager was started (in the form hh:mm:ss) (parameter identifier: MQCACF\_Q\_MGR\_START\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

### **V 9.3.2 StatusType**

The type of status being returned:

#### **MQIACF\_Q\_MGR\_STATUS\_INFO\_Q\_MGR**

Returns general status information relating to the queue manager.

### V 9.3.2 **TotalInstances (MQCFIN)**

Total number of configured instances of the queue manager (parameter identifier: MQIACF\_NHA\_TOTAL\_INSTANCES).

### V 9.3.2 **UniClusterName (MQCFST)**

The name of the uniform cluster that the queue manager is a member of, or blank otherwise. (parameter identifier: MQCACF\_UNIFORM\_CLUSTER\_NAME). The length of this attribute is given by MQ\_CLUSTER\_NAME\_LENGTH.

## **Response data (if StatusType is MQIACF\_Q\_MGR\_STATUS\_INFO\_NHA)**

### V 9.3.2

#### **Backlog (MQCFIN)**

How far 'out of sync' the instance currently is (parameter identifier: MQIACF\_NHA\_INSTANCE\_BACKLOG). For REPLICIA instances only.

#### **ConnActv (MQCFIN)**

Whether the instance currently has a pair of active connections to the active instance (parameter identifier: MQIACF\_NHA\_INSTANCE\_ACTV\_CONNS). For REPLICIA instances only.

The value can be:

##### **MQNHACONNACTV\_NO**

No pair of active connections.

##### **MQNHACONNACTV\_YES**

Has a pair of active connections.

#### **HAIInitDate (MQCFST)**

The date of the last log record recovered when the Native HA group initially became active (in the form *yyyy-mm-dd*) (parameter identifier: MQCACF\_NHA\_GROUP\_INITIAL\_DATE). For ACTIVE instances only.

#### **HAIInitLSN (MQCFST)**

The log sequence number (LSN) of the last log record recovered when the Native HA group initially became active (in the form *<nnnnn:nnnnn:nnnnn:nnnn>*) (parameter identifier: MQCACF\_NHA\_GROUP\_INITIAL\_LSN). For ACTIVE instances only.

#### **HAIInitTime (MQCFST)**

The time of the last log record recovered when the Native HA group initially became active (in the form *hh.mm.ss*) (parameter identifier: MQCACF\_NHA\_GROUP\_INITIAL\_TIME). For ACTIVE instances only.

#### **InSync (MQCFIN)**

Whether this instance is currently considered in-sync with the active instance (parameter identifier: MQIACF\_NHA\_INSTANCE\_IN\_SYNC). For REPLICIA instances only.

The value can be:

##### **MQNHAINSYNC\_NO**

Is not in sync.

##### **MQNHAINSYNC\_YES**

Is in sync.

#### **Instance (MQCFST)**

The name of the local Native HA instance (parameter identifier: MQCACF\_NHA\_INSTANCE\_NAME). The length of this attribute is given by MQ\_NHA\_INSTANCE\_NAME\_LENGTH.

#### **ReplAddr (MQCFST)**

The network address and port to use when sending data to and from the specified instance (parameter identifier: MQCACF\_NHA\_REPL\_ADDRESS). The length of this attribute is given by MQ\_NHA\_REPL\_ADDRESS\_LENGTH.

#### **Role (MQCFIN)**

The role that the instance is currently playing in the Native HA group (parameter identifier: MQIACF\_NHA\_INSTANCE\_ROLE).

The value can be:

**MQNHAROLE\_UNKNOWN**

Role is not known.

**MQNHAROLE\_ACTIVE**

Role is ACTIVE.

**MQNHAROLE\_REPLICA**

Role is REPLICA.

See [Native HA](#) for a description of these roles.

**StatusType**

The type of status being returned:

**MQIACF\_Q\_MGR\_STATUS\_INFO\_NHA**

Returns status information relating to Native HA instances.

**MQCMD\_INQUIRE\_Q\_NAMES (Inquire Queue Names)**

The Inquire Queue Names (MQCMD\_INQUIRE\_Q\_NAMES) PCF command inquires a list of queue names that match the generic queue name, and the optional queue type specified.

**Required parameters**

**QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

Generic queue names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_Q\_LENGTH.

**Optional parameters**

**z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

When a value other than blank is specified, the maximum response size is limited to 32KB from each queue manager. If the response from a queue manager would be larger than this, an error response with reason code [MQRCCF\\_COMMAND\\_LENGTH\\_ERROR \(3230\)](#) is returned by that queue manager.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

**z/OS QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

**MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

**MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_PRIVATE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED. MQQSGD\_SHARED is permitted only in a shared queue environment.

**QType (MQCFIN)**

Queue type (parameter identifier: MQIA\_Q\_TYPE).

If present, this parameter limits the queue names returned to queues of the specified type. If this parameter is not present, queues of all types are eligible. The value can be any of the following values:

**MQQT\_ALL**

All queue types.

**MQQT\_LOCAL**

Local queue.

**MQQT\_ALIAS**

Alias queue definition.

**MQQT\_REMOTE**

Local definition of a remote queue.

**MQQT\_MODEL**

Model queue definition.

The default value if this parameter is not specified is MQQT\_ALL.

**MQCMD\_INQUIRE\_Q\_NAMES (Inquire Queue Names) Response**


The response to the Inquire Queue Names (MQCMD\_INQUIRE\_Q\_NAMES) PCF command consists of the response header followed by a single parameter structure giving zero or more names that match the specified queue name. The response header is followed by the *QTypes* structure, with the same number of entries as the *QNames* structure. Each entry gives the type of the queue with the corresponding entry in the *QNames* structure.



Additionally, on z/OS only, the **QSGDispositions** parameter structure (with the same number of entries as the *QNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *QNames* structure.



**Always returned:**

*QNames*,  *QSGDispositions*, *QTypes*

**Returned if requested:**

None

**Response data****QNames (MQCFSL)**

List of queue names (parameter identifier: MQACF\_Q\_NAMES).

 **QSGDispositions (MQCFIL)**

List of queue sharing group dispositions (parameter identifier: MQIACF\_QSG\_DISPS). This parameter is valid on z/OS only. Possible values for fields in this structure are:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED.

**QTypes (MQCFIL)**

List of queue types (parameter identifier: MQIACF\_Q\_TYPES). Possible values for fields in this structure are:

**MQQT\_ALIAS**

Alias queue definition.

**MQQT\_LOCAL**

Local queue.

**MQQT\_REMOTE**

Local definition of a remote queue.

**MQQT\_MODEL**

Model queue definition.

**MQCMD\_INQUIRE\_Q\_STATUS (Inquire Queue Status)**

The Inquire Queue Status (MQCMD\_INQUIRE\_Q\_STATUS) PCF command inquires about the status of a local IBM MQ queue. You must specify the name of a local queue for which you want to receive status information.

**Required parameters****QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

Generic queue names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all queues having names that start with the selected character string. An asterisk on its own matches all possible names.

The queue name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**Optional parameters (Inquire Queue Status)****ByteStringFilterCommand (MQCFBF)**

Byte string filter command descriptor. The parameter identifier must be MQBACF\_EXTERNAL\_UOW\_ID or MQBACF\_Q\_MGR\_UOW\_ID. Use this parameter to restrict the output

from the command by specifying a filter condition. See [“MQCFBF - PCF byte string filter parameter” on page 1518](#) for information about using this filter condition.

If you specify a byte string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter, or a string filter using the **StringFilterCommand** parameter.

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is initiated when the queue manager is a member of a queue sharing group. You can specify one of the following:

- Blank (or omit the parameter altogether). The command is initiated on the queue manager on which it was entered.
- Queue manager name. The command is initiated on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be initiated.
- An asterisk (\*). The command is initiated on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *QStatusAttrs* except MQIACF\_ALL, MQIACF\_MONITORING, and MQIACF\_Q\_TIME\_INDICATOR. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter, you cannot also specify a byte string filter using the **ByteStringFilterCommand** parameter or a string filter using the **StringFilterCommand** parameter.

### **OpenType (MQCFIN)**

Queue status open type (parameter identifier: MQIACF\_OPEN\_TYPE).

It is always returned, regardless of the queue instance attributes requested.

The value can be:

#### **MQQSOT\_ALL**

Selects status for queues that are open with any type of access.

#### **MQQSOT\_INPUT**

Selects status for queues that are open for input.

#### **MQQSOT\_OUTPUT**

Selects status for queues that are open for output.

The default value if this parameter is not specified is MQQSOT\_ALL.

Filtering is not supported for this parameter.

### **QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid only on z/OS. The value can be any of the following values:

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED.

You cannot use *QSGDisposition* as a parameter to filter on.

**QStatusAttrs (MQCFIL)**

Queue status attributes (parameter identifier: MQIACF\_Q\_STATUS\_ATTRS).

The attribute list can specify the following value on its own - default value used if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

or a combination of the following:

Where *StatusType* is MQIACF\_Q\_STATUS:

**MQCA\_Q\_NAME**

Queue name.

**MQCACF\_LAST\_GET\_DATE**

Date of the last message successfully destructively read from the queue.

**MQCACF\_LAST\_GET\_TIME**

Time of the last message successfully destructively read from the queue.

**MQCACF\_LAST\_PUT\_DATE**

Date of the last message successfully put to the queue.

**MQCACF\_LAST\_PUT\_TIME**

Time of the last message successfully put to the queue.

**MQCACF\_MEDIA\_LOG\_EXTENT\_NAME**

Identity of the oldest log extent required to perform media recovery of the queue.

On IBM i, this parameter identifies the name of the oldest journal receiver require to perform media recovery of the queue.

**MQIA\_CURRENT\_Q\_DEPTH**

The current number of messages on the queue.

**MQIA\_MONITORING\_Q**

Current level of monitoring data collection.

**MQIA\_OPEN\_INPUT\_COUNT**

The number of handles that are currently open for input for the queue.

MQIA\_OPEN\_INPUT\_COUNT does not include handles that are open for browse.

**MQIA\_OPEN\_OUTPUT\_COUNT**

The number of handles that are currently open for output for the queue.

**MQIACF\_HANDLE\_STATE**

Whether an API call is in progress.

**MQIACF\_MONITORING**

All the queue status monitoring attributes. These attributes are:

- MQCACF\_LAST\_GET\_DATE
- MQCACF\_LAST\_GET\_TIME
- MQCACF\_LAST\_PUT\_DATE
- MQCACF\_LAST\_PUT\_TIME
- MQIA\_MONITORING\_Q
- MQIACF\_OLDEST\_MSG\_AGE
- MQIACF\_Q\_TIME\_INDICATOR

Filtering is not supported for this parameter.

**MQIACF\_CUR\_MAX\_FILE\_SIZE**

Current maximum queue file size

**MQIACF\_CUR\_Q\_FILE\_SIZE)**

Current queue file size

**MQIACF\_OLDEST\_MSG\_AGE**

Age of oldest message on the queue.

**MQIACF\_Q\_TIME\_INDICATOR**

Indicator of the time that messages remain on the queue.

**MQIACF\_UNCOMMITTED\_MSGS**

The number of uncommitted messages on the queue.

Where *StatusType* is MQIACF\_Q\_HANDLE:

**MQBACF\_EXTERNAL\_UOW\_ID**

Unit of recovery identifier assigned by the queue manager.

**MQBACF\_Q\_MGR\_UOW\_ID**

External unit of recovery identifier associated with the connection.

**MQCA\_Q\_NAME**

Queue name.

**MQCACF\_APPL\_TAG**

This parameter is a string containing the tag of the application connected to the queue manager.

 **MQCACF\_ASID**

Address-space identifier of the application identified by *Appl.Tag*. This parameter is valid on z/OS only.

**MQCACF\_PSB\_NAME**

Name of the program specification block (PSB) associated with the running IMS transaction. This parameter is valid on z/OS only.

**MQCACF\_PSTID**

Identifier of the IMS program specification table (PST) for the connected IMS region. This parameter is valid on z/OS only.

**MQCACF\_TASK\_NUMBER**

CICS task number. This parameter is valid on z/OS only.

**MQCACF\_TRANSACTION\_ID**

CICS transaction identifier. This parameter is valid on z/OS only.

**MQCACF\_USER\_IDENTIFIER**

The user name of the application that has opened the specified queue.

**MQCACH\_CHANNEL\_NAME**

The name of the channel that has the queue open, if any.

**MQCACH\_CONNECTION\_NAME**

The connection name of the channel that has the queue open, if any.

**MQIA\_APPL\_TYPE**

The type of application that has the queue open.

**MQIACF\_OPEN\_BROWSE**

Open browse.

Filtering is not supported for this parameter.

**MQIACF\_OPEN\_INPUT\_TYPE**

Open input type.

Filtering is not supported for this parameter.

**MQIACF\_OPEN\_INQUIRE**

Open inquire.

Filtering is not supported for this parameter.

**MQIACF\_OPEN\_OPTIONS**

The options used to open the queue.

If this parameter is requested, the following parameter structures are also returned:

- *OpenBrowse*
- *OpenInputType*
- *OpenInquire*
- *OpenOutput*
- *OpenSet*

Filtering is not supported for this parameter.

**MQIACF\_OPEN\_OUTPUT**

Open output.

Filtering is not supported for this parameter.

**MQIACF\_OPEN\_SET**

Open set.

Filtering is not supported for this parameter.

**MQIACF\_PROCESS\_ID**

The process identifier of the application that has opened the specified queue.

**MQIACF\_ASYNC\_STATE****MQIACF\_THREAD\_ID**

The thread identifier of the application that has opened the specified queue.

**MQIACF\_UOW\_TYPE**

Type of external unit of recovery identifier as seen by the queue manager.

**StatusType (MQCFIN)**

Queue status type (parameter identifier: MQIACF\_Q\_STATUS\_TYPE).

Specifies the type of status information required.

The value can be any of the following values:

**MQIACF\_Q\_STATUS**

Selects status information relating to queues.

**MQIACF\_Q\_HANDLE**

Selects status information relating to the handles that are accessing the queues.

The default value, if this parameter is not specified, is MQIACF\_Q\_STATUS.

You cannot use *StatusType* as a parameter to filter on.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *QStatusAttrs* except MQCA\_Q\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 1530 for information about using this filter condition.

If you specify a string filter, you cannot also specify a byte string filter using the **ByteStringFilterCommand** parameter or an integer filter using the **IntegerFilterCommand** parameter.

## Error codes

This command might return the following error code in the response format header “[Error codes applicable to all commands](#)” on page 990 along with any additional pertinent values.

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRCCF\_Q\_TYPE\_ERROR**

Queue type not valid.

### **MQCMD\_INQUIRE\_Q\_STATUS (Inquire Queue Status) Response**

The response to the Inquire Queue Status (MQCMD\_INQUIRE\_Q\_STATUS) PCF command consists of the response header followed by the *QName* structure and a set of attribute parameter structures determined by the value of *StatusType* in the Inquire command.

#### Always returned:

*QName*, *ApplTag*, *ApplType*, **z/OS** *QSGDisposition*, *StatusType*,  
**Multi** *UserIdentifier*

Possible values of *StatusType* are:

#### **MQIACF\_Q\_STATUS**

Returns status information relating to queues.

#### **MQIACF\_Q\_HANDLE**

Returns status information relating to the handles that are accessing the queues.

#### Returned if requested and *StatusType* is MQIACF\_Q\_STATUS:

**Multi** *CurrentMaxQFileSize*, **Multi** *CurrentQFileSize*, *CurrentQDepth*,  
*LastGetDate*, *LastGetTime*, *LastPutDate*, *LastPutTime*, **Multi** *MediaRecoveryLogExtent*,  
*OldestMsgAge*, *OnQTime*, *OpenInputCount*, *OpenOutputCount*,  
*QueueMonitoring*, *UncommittedMsgs*

#### Returned if requested and *StatusType* is MQIACF\_Q\_HANDLE:

*ApplDesc*, *ApplTag*, *ApplType*, **z/OS** *ASId*, *AsynchronousState*, *ChannelName*,  
*ConnectionName*, **z/OS** *ExternalUOWId*, *HandleState*, *OpenOptions*, **Multi** *ProcessId*,  
**z/OS** *PSBName*, **z/OS** *PSTId*, *QMgrUOWId*, **z/OS** *TaskNumber*, **Multi** *ThreadId*,  
**z/OS** *TransactionId*, *UOWIdentifier*, *UOWType*, *UserIdentifier*

## Response data if *StatusType* is MQIACF\_Q\_STATUS

### **Multi** **CurrentMaxQFileSize (MQCFIN)**

Current maximum queue file size (parameter identifier MQIACF\_CUR\_MAX\_FILE\_SIZE)

The current maximum size the queue file can grow to, rounded up to the nearest megabyte, given the current block size in use on a queue

### **Multi** **CurrentQFileSize (MQCFIN)**

Current queue file size (parameter identifier MQIACF\_CUR\_Q\_FILE\_SIZE)

The current size of the queue file in megabytes, rounded up to the nearest megabyte.

### **CurrentQDepth (MQCFIN)**

Current queue depth (parameter identifier: MQIA\_CURRENT\_Q\_DEPTH).

### **LastGetDate (MQCFST)**

Date on which the last message was destructively read from the queue (parameter identifier: MQCACF\_LAST\_GET\_DATE).

The date, in the form yyyy-mm-dd, on which the last message was successfully read from the queue. The date is returned in the time zone in which the queue manager is running.

The maximum length of the string is MQ\_DATE\_LENGTH.

#### **LastGetTime (MQCFST)**

Time at which the last message was destructively read from the queue (parameter identifier: MQCACF\_LAST\_GET\_TIME).

The time, in the form hh.mm.ss, at which the last message was successfully read from the queue. The time is returned in the time zone in which the queue manager is running.

The maximum length of the string is MQ\_TIME\_LENGTH.

#### **LastPutDate (MQCFST)**

Date on which the last message was successfully put to the queue (parameter identifier: MQCACF\_LAST\_PUT\_DATE).

The date, in the form yyyy-mm-dd, on which the last message was successfully put to the queue. The date is returned in the time zone in which the queue manager is running.

The maximum length of the string is MQ\_DATE\_LENGTH.

#### **LastPutTime (MQCFST)**

Time at which the last message was successfully put to the queue (parameter identifier: MQCACF\_LAST\_PUT\_TIME).

The time, in the form hh.mm.ss, at which the last message was successfully put to the queue. The time is returned in the time zone in which the queue manager is running.

The maximum length of the string is MQ\_TIME\_LENGTH.

#### **Multi MediaRecoveryLogExtent (MQCFST)**

Name of the oldest log extent required to perform media recovery of the queue (parameter identifier: MQCACF\_MEDIA\_LOG\_EXTENT\_NAME).

On IBM i, this parameter identifies the name of the oldest journal receiver required to perform media recovery of the queue.

The name returned is of the form Snnnnnnn.LOG and is not a fully qualified path name. The use of this parameter provides the ability for the name to be easily correlated with the messages issued, following an **rcdmqimg** command to identify those queues causing the media recovery LSN not to move forwards.

This parameter is valid only on [Multiplatforms](#).

The maximum length of the string is MQ\_LOG\_EXTENT\_NAME\_LENGTH.

#### **OldestMsgAge (MQCFIN)**

Age of the oldest message (parameter identifier: MQIACF\_OLDEST\_MSG\_AGE). Age, in seconds, of the oldest message on the queue.

If the value is unavailable, MQMON\_NOT\_AVAILABLE is returned. If the queue is empty, 0 is returned. If the value exceeds 999 999 999, it is returned as 999 999 999.

#### **OnQTime (MQCFIL)**

Indicator of the time that messages remain on the queue (parameter identifier: MQIACF\_Q\_TIME\_INDICATOR). Amount of time, in microseconds, that a message spent on the queue. Two values are returned:

- A value based on recent activity over a short period.
- A value based on activity over a longer period.

Where no measurement is available, the value MQMON\_NOT\_AVAILABLE is returned. If the value exceeds 999 999 999, it is returned as 999 999 999.

**OpenInputCount (MQCFIN)**

Open input count (parameter identifier: MQIA\_OPEN\_INPUT\_COUNT).

**OpenOutputCount (MQCFIN)**

Open output count (parameter identifier: MQIA\_OPEN\_OUTPUT\_COUNT).

**QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**z/OS QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Returns the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be any of the following values:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED.

**QueueMonitoring (MQCFIN)**

Current level of monitoring data collection for the queue (parameter identifier: MQIA\_MONITORING\_Q). The value can be any of the following values:

**MQMON\_OFF**

Monitoring for the queue is disabled.

**MQMON\_LOW**

Low rate of data collection.

**MQMON\_MEDIUM**

Medium rate of data collection.

**MQMON\_HIGH**

High rate of data collection.

**StatusType (MQCFST)**

Queue status type (parameter identifier: MQIACF\_Q\_STATUS\_TYPE).

Specifies the type of status information.

**UncommittedMsgs (MQCFIN)**

The number of uncommitted changes (puts and gets) pending for the queue (parameter identifier: MQIACF\_UNCOMMITTED\_MSGS). The value can be any of the following values:

**MQQSUM\_YES**

On z/OS, there are one or more uncommitted changes pending.

**MQQSUM\_NO**

There are no uncommitted changes pending.

n

**Multi** On Multiplatforms, an integer value indicating how many uncommitted changes are pending.

**Response data if StatusType is MQIACF\_Q\_HANDLE****ApplDesc (MQCFST)**

Application description (parameter identifier: MQCACF\_APPL\_DESC).

The maximum length is MQ\_APPL\_DESC\_LENGTH.



**ApplTag (MQCFST)**

Open application tag (parameter identifier: MQCACF\_APPL\_TAG).

The maximum length of the string is MQ\_APPL\_TAG\_LENGTH.

**ApplType (MQCFIN)**

Open application type (parameter identifier: MQIA\_APPL\_TYPE).

The value can be any of the following values:

**MQAT\_QMGR**

A queue manager process.

**MQAT\_CHANNEL\_INITIATOR**

The channel initiator.

**MQAT\_USER**

A user application.

**z/OS MQAT\_BATCH**

Application using a batch connection. MQAT\_BATCH applies only to z/OS.

**z/OS MQAT\_RRS\_BATCH**

RRS-coordinated application using a batch connection. MQAT\_RRS\_BATCH applies only to z/OS.

**z/OS MQAT\_CICS**

A CICS transaction. MQAT\_CICS applies only to z/OS.

**z/OS MQAT\_IMS**

An IMS transaction. MQAT\_IMS applies only to z/OS.

**MQAT\_SYSTEM\_EXTENSION**

Application performing an extension of function that is provided by the queue manager.

**z/OS ASId (MQCFST)**

Address-space identifier (parameter identifier: MQCACF\_ASID).

The 4-character address-space identifier of the application identified by *ApplTag*. It distinguishes duplicate values of *ApplTag*. This parameter applies only to z/OS.

The length of the string is MQ\_ASID\_LENGTH.

**AsynchronousState (MQCFIN)**

The state of the asynchronous consumer on this queue (parameter identifier: MQIACF\_ASYNC\_STATE).

The value can be any of the following values:

**MQAS\_ACTIVE**

An MQCB call has set up a function to call back to process messages asynchronously and the connection handle has been started so that asynchronous message consumption can proceed.

**MQAS\_INACTIVE**

An MQCB call has set up a function to call back to process messages asynchronously but the connection handle has not yet been started, or has been stopped or suspended, so that asynchronous message consumption cannot currently proceed.

**MQAS\_SUSPENDED**

The asynchronous consumption callback has been suspended so that asynchronous message consumption cannot currently proceed on this handle. This situation can be either because an MQCB or MQCTL call with *Operation* MQOP\_SUSPEND has been issued against this object handle by the application, or because it has been suspended by the system. If it has been suspended by the system, as part of the process of suspending asynchronous message consumption the callback function is called with the reason code that describes the problem resulting in suspension. This situation is reported in the *Reason* field in the MQCBC structure passed to the callback. In order for asynchronous message consumption to proceed, the application must issue an MQCB or MQCTL call with *Operation* MQOP\_RESUME.

### **MQAS\_SUSPENDED\_TEMPORARY**

The asynchronous consumption callback has been temporarily suspended by the system so that asynchronous message consumption cannot currently proceed on this object handle. As part of the process of suspending asynchronous message consumption the callback function is called with the reason code that describes the problem resulting in suspension. This situation is reported in the *Reason* field in the MQCBC structure passed to the callback. The callback function is called again when asynchronous message consumption is resumed by the system after the temporary condition has been resolved.

### **MQAS\_NONE**

An MQCB call has not been issued against this handle, so no asynchronous message consumption is configured on this handle.

### **ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

### **ConnectionName (MQCFST)**

Connection name (parameter identifier: MQCACH\_CONNECTION\_NAME).

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH.

### **ExternalUOWId (MQCFBS)**

RRS unit-of-recovery identifier (parameter identifier: MQBACF\_EXTERNAL\_UOW\_ID).

The RRS unit-of-recovery identifier associated with the handle. This parameter is valid only on z/OS only.

The length of the string is MQ\_EXTERNAL\_UOW\_ID\_LENGTH.

### **HandleState (MQCFIN)**

State of the handle (parameter identifier: MQIACF\_HANDLE\_STATE).

The value can be any of the following values:

#### **MQHSTATE\_ACTIVE**

An API call from a connection is currently in progress for this object. For a queue, this condition can arise when an MQGET WAIT call is in progress.

If there is an MQGET SIGNAL outstanding, it does not mean, by itself, that the handle is active.

#### **MQHSTATE\_INACTIVE**

No API call from a connection is currently in progress for this object. For a queue, this condition can arise when no MQGET WAIT call is in progress.

### **OpenBrowse (MQCFIN)**

Open browse (parameter identifier: MQIACF\_OPEN\_BROWSE).

The value can be any of the following values:

#### **MQQSO\_YES**

The queue is open for browsing.

#### **MQQSO\_NO**

The queue is not open for browsing.

### **OpenInputType (MQCFIN)**

Open input type (parameter identifier: MQIACF\_OPEN\_INPUT\_TYPE).

The value can be any of the following values:

#### **MQQSO\_NO**

The queue is not open for inputting.

#### **MQQSO\_SHARED**

The queue is open for shared input.

**MQQSO\_EXCLUSIVE**

The queue is open for exclusive input.

**OpenInquire (MQCFIN)**

Open inquire (parameter identifier: MQIACF\_OPEN\_INQUIRE).

The value can be any of the following values:

**MQQSO\_YES**

The queue is open for inquiring.

**MQQSO\_NO**

The queue is not open for inquiring.

**OpenOptions (MQCFIN)**

Open options currently in force for the queue (parameter identifier: MQIACF\_OPEN\_OPTIONS).

**OpenOutput (MQCFIN)**

Open output (parameter identifier: MQIACF\_OPEN\_OUTPUT).

The value can be any of the following values:

**MQQSO\_YES**

The queue is open for output.

**MQQSO\_NO**

The queue is not open for output.

**OpenSet (MQCFIN)**

Open set (parameter identifier: MQIACF\_OPEN\_SET).

The value can be any of the following values:

**MQQSO\_YES**

The queue is open for setting.

**MQQSO\_NO**

The queue is not open for setting.

**Multi ProcessId (MQCFIN)**

Open application process ID (parameter identifier: MQIACF\_PROCESS\_ID).

**z/OS PSBName (MQCFST)**

Program specification block (PSB) name (parameter identifier: MQCACF\_PSB\_NAME).

The 8-character name of the PSB associated with the running IMS transaction. This parameter is valid on z/OS only.

The length of the string is MQ\_PSB\_NAME\_LENGTH.

**z/OS PSTId (MQCFST)**

Program specification table (PST) identifier (parameter identifier: MQCACF\_PST\_ID).

The 4-character identifier of the PST region identifier for the connected IMS region. This parameter is valid on z/OS only.

The length of the string is MQ\_PST\_ID\_LENGTH.

**QMgrUOWId (MQCFBS)**

The unit of recovery assigned by the queue manager (parameter identifier: MQBACF\_Q\_MGR\_UOW\_ID).

On z/OS, this parameter is an 8-byte log RBA, displayed as 16 hexadecimal characters. On platforms other than z/OS, this parameter is an 8-byte transaction identifier, displayed as 16 hexadecimal characters.

The maximum length of the string is MQ\_UOW\_ID\_LENGTH.

**QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**z/OS QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Returns the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be any of the following values:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED.

**StatusType (MQCFST)**

Queue status type (parameter identifier: MQIACF\_Q\_STATUS\_TYPE).

Specifies the type of status information.

**z/OS TaskNumber (MQCFST)**

CICS task number (parameter identifier: MQCACF\_TASK\_NUMBER).

A 7-digit CICS task number. This parameter is valid on z/OS only.

The length of the string is MQ\_TASK\_NUMBER\_LENGTH.

**Multi ThreadId (MQCFIN)**

The thread ID of the open application (parameter identifier: MQIACF\_THREAD\_ID).

A value of zero indicates that the handle was opened by a shared connection. A handle created by a shared connection is logically open to all threads.

**z/OS TransactionId (MQCFST)**

CICS transaction identifier (parameter identifier: MQCACF\_TRANSACTION\_ID).

A 4-character CICS transaction identifier. This parameter is valid on z/OS only.

The length of the string is MQ\_TRANSACTION\_ID\_LENGTH.

**UOWIdentifier (MQCFBS)**

The external unit of recovery associated with the connection (parameter identifier: MQBACF\_EXTERNAL\_UOW\_ID).

This parameter is the recovery identifier for the unit of recovery. Its format is determined by the value of *UOWType*.

The maximum length of the string is MQ\_UOW\_ID\_LENGTH.

**UOWType (MQCFIN)**

Type of external unit of recovery identifier as perceived by the queue manager (parameter identifier: MQIACF\_UOW\_TYPE).

The value can be any of the following values:

**MQUOWT\_Q\_MGR****z/OS MQUOWT\_CICS**

Valid only on z/OS.

**z/OS MQUOWT\_RRS**

Valid only on z/OS.

## **z/OS** >MQUOWT\_IMS

Valid only on z/OS.

### **MQUOWT\_XA**

*UOWType* identifies the *UOWIdentifier* type and not the type of the transaction coordinator. When the value of *UOWType* is MQUOWT\_Q\_MGR, the associated identifier is in *QMgrUOWId* (and not *UOWIdentifier*).

### **UserIdentifier (MQCFST)**

Open application user name (parameter identifier: MQCACF\_USER\_IDENTIFIER).

The maximum length of the string is MQ\_MAX\_USER\_ID\_LENGTH.

## **z/OS** **MQCMD\_INQUIRE\_SECURITY (Inquire Security) on z/OS**

The Inquire Security (MQCMD\_INQUIRE\_SECURITY) PCF command returns information about the current settings for the security parameters.

### **Optional parameters**

#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

#### **SecurityAttrs (MQCFIL)**

Security parameter attributes (parameter identifier: MQIACF\_SECURITY\_ATTRS).

The attribute list might specify the following value on its own - default value used if the parameter is not specified:

#### **MQIACF\_ALL**

All attributes.

or a combination of the following:

#### **MQIACF\_SECURITY\_SWITCH**

Current setting of the switch profiles. If the subsystem security switch is off, no other switch profile settings are returned.

#### **MQIACF\_SECURITY\_TIMEOUT**

Timeout value.

#### **MQIACF\_SECURITY\_INTERVAL**

Time interval between checks.

## **z/OS** **MQCMD\_INQUIRE\_SECURITY (Inquire Security) Response on z/OS**

The response to the Inquire Security (MQCMD\_INQUIRE\_SECURITY) PCF command consists of the response header followed by the requested combination of attribute parameter structures.

One message is returned if either **SecurityTimeout** or **SecurityInterval** is specified on the command. If **SecuritySwitch** is specified, one message per security switch found is returned. This

message includes the **SecuritySwitch**, **SecuritySwitchSetting**, and **SecuritySwitchProfile** parameter structures.

**Returned if requested:**

**SecurityInterval**, **SecuritySwitch**, **SecuritySwitchProfile**, **SecuritySwitchSetting**, **SecurityTimeout**

**Response data**

**SecurityInterval (MQCFIN)**

Time interval between checks (parameter identifier: MQIACF\_SECURITY\_INTERVAL).

The interval, in minutes, between checks for user IDs and their associated resources to determine whether **SecurityTimeout** has expired.

**SecuritySwitch (MQCFIN)**

Security switch profile (parameter identifier: MQIA\_CF\_LEVEL). The value can be any of the following values:

**MQSECSW\_SUBSYSTEM**

Subsystem security switch.

**MQSECSW\_Q\_MGR**

Queue manager security switch.

**MQSECSW\_QSG**

Queue sharing group security switch.

**MQSECSW\_CONNECTION**

Connection security switch.

**MQSECSW\_COMMAND**

Command security switch.

**MQSECSW\_CONTEXT**

Context security switch.

**MQSECSW\_ALTERNATE\_USER**

Alternate user security switch.

**MQSECSW\_PROCESS**

Process security switch.

**MQSECSW\_NAMELIST**

Namelist security switch.

**MQSECSW\_TOPIC**

Topic security switch.

**MQSECSW\_Q**

Queue security switch.

**MQSECSW\_COMMAND\_RESOURCES**

Command resource security switch.

**SecuritySwitchProfile (MQCFST)**

Security switch profile (parameter identifier: MQCACF\_SECURITY\_PROFILE).

The maximum length of the string is MQ\_SECURITY\_PROFILE\_LENGTH.

**SecuritySwitchSetting (MQCFIN)**

Setting of the security switch (parameter identifier: MQIACF\_SECURITY\_SETTING).

The value can be:

**MQSECSW\_ON\_FOUND**

Switch ON, profile found.

**MQSECSW\_OFF\_FOUND**

Switch OFF, profile found.

**MQSECSW\_ON\_NOT\_FOUND**

Switch ON, profile not found.

**MQSECSW\_OFF\_NOT\_FOUND**

Switch OFF, profile not found.

**MQSECSW\_OFF\_ERROR**

Switch OFF, profile error.

**MQSECSW\_ON\_OVERRIDDEN**

Switch ON, profile overridden.

**SecurityTimeout (MQCFIN)**

Timeout value (parameter identifier: MQIACF\_SECURITY\_TIMEOUT).

How long, in minutes, security information about an unused user ID and associated resources is retained.

 **MQCMD\_INQUIRE\_SERVICE (Inquire Service) on Multiplatforms**

The Inquire Service (MQCMD\_INQUIRE\_SERVICE) PCF command inquires about the attributes of existing IBM MQ services.

**Required parameters****ServiceName (MQCFST)**

Service name (parameter identifier: MQCA\_SERVICE\_NAME).

This parameter is the name of the service whose attributes are required. Generic service names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all services having names that start with the selected character string. An asterisk on its own matches all possible names.

The service name is always returned regardless of the attributes requested.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

**Optional parameters****IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ServiceAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

**ServiceAttrs (MQCFIL)**

Service attributes (parameter identifier: MQIACF\_SERVICE\_ATTRS).

The attribute list might specify the following value on its own - default value if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

or a combination of the following:

**MQCA\_ALTERATION\_DATE**

Date on which the definition was last altered.

**MQCA\_ALTERATION\_TIME**

Time at which the definition was last altered.

**MQCA\_SERVICE\_DESC**

Description of service definition.

**MQCA\_SERVICE\_NAME**

Name of service definition.

**MQCA\_SERVICE\_START\_ARGS**

Arguments to be passed to the service program.

**MQCA\_SERVICE\_START\_COMMAND**

Name of program to run to start the service.

**MQCA\_SERVICE\_STOP\_ARGS**

Arguments to be passed to the stop program to stop the service.

**MQCA\_STDERR\_DESTINATION**

Destination of standard error for the process.

**MQCA\_STDOUT\_DESTINATION**

Destination of standard output for the process.

**MQCA\_SERVICE\_START\_ARGS**

Arguments to be passed to the service program.

**MQIA\_SERVICE\_CONTROL**

When the queue manager must start the service.

**MQIA\_SERVICE\_TYPE**

Mode in which the service is to run.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ServiceAttrs* except MQCA\_SERVICE\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

**Multi MQCMD\_INQUIRE\_SERVICE (Inquire Service) Response on Multiplatforms**

The response to the Inquire Service (MQCMD\_INQUIRE\_SERVICE) PCF command consists of the response header followed by the *ServiceName* structure and the requested combination of attribute parameter structures.

If a generic service name was specified, one such message is generated for each service found.

**Always returned:**

*ServiceName*

**Returned if requested:**

*AlterationDate, AlterationTime, Arguments, ServiceDesc, ServiceType, StartArguments, StartCommand, StartMode, StderrDestination, StdoutDestination, StopArguments, StopCommand*

**Response data****AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date on which the information was last altered in the form yyyy-mm-dd.

**AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time at which the information was last altered in the form hh.mm.ss.

**ServiceDesc (MQCFST)**

Description of service definition (parameter identifier: MQCA\_SERVICE\_DESC).

The maximum length of the string is MQ\_SERVICE\_DESC\_LENGTH.



**ServiceName (MQCFST)**

Name of service definition (parameter identifier: MQCA\_SERVICE\_NAME).

The maximum length of the string is MQ\_SERVICE\_NAME\_LENGTH.

**ServiceType (MQCFIN)**

The mode in which the service is to run (parameter identifier: MQIA\_SERVICE\_TYPE).

The value can be:

**MQSVC\_TYPE\_SERVER**

Only one instance of the service can be executed at a time, with the status of the service made available by the Inquire Service Status command.

**MQSVC\_TYPE\_COMMAND**

Multiple instances of the service can be started.

**StartArguments (MQCFST)**

The arguments to be passed to the user program at queue manager startup (parameter identifier: MQCA\_SERVICE\_START\_ARGS).

The maximum length of the string is MQ\_SERVICE\_ARGS\_LENGTH.

**StartCommand (MQCFST)**

Service program name (parameter identifier: MQCA\_SERVICE\_START\_COMMAND).

The name of the program which is to run.

The maximum length of the string is MQ\_SERVICE\_COMMAND\_LENGTH.

**StartMode (MQCFIN)**

Service mode (parameter identifier: MQIA\_SERVICE\_CONTROL).

Specifies how the service is to be started and stopped. The value can be any of the following values:

**MQSVC\_CONTROL\_MANUAL**

The service is not to be started automatically or stopped automatically. It is to be controlled by user command.

**MQSVC\_CONTROL\_Q\_MGR**

The service is to be started and stopped at the same time as the queue manager is started and stopped.

**MQSVC\_CONTROL\_Q\_MGR\_START**

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**StderrDestination (MQCFST)**

The path to a file to which the standard error (stderr) of the service program is to be redirected (parameter identifier: MQCA\_STDERR\_DESTINATION).

The maximum length of the string is MQ\_SERVICE\_PATH\_LENGTH.

**StdoutDestination (MQCFST)**

The path to a file to which the standard output (stdout) of the service program is to be redirected (parameter identifier: MQCA\_STDOUT\_DESTINATION).

The maximum length of the string is MQ\_SERVICE\_PATH\_LENGTH.

**StopArguments (MQCFST)**

The arguments to be passed to the stop program when instructed to stop the service (parameter identifier: MQCA\_SERVICE\_STOP\_ARGS).

The maximum length of the string is MQ\_SERVICE\_ARGS\_LENGTH.

**StopCommand (MQCFST)**

Service program stop command (parameter identifier: MQCA\_SERVICE\_STOP\_COMMAND).

This parameter is the name of the program that is to run when the service is requested to stop.

The maximum length of the string is MQ\_SERVICE\_COMMAND\_LENGTH.

## **Multi** **MQCMD\_INQUIRE\_SERVICE\_STATUS (Inquire Service Status) on Multiplatforms**

The Inquire Service Status (MQCMD\_INQUIRE\_SERVICE\_STATUS) PCF command inquires about the status of one or more IBM MQ service instances.

### **Required parameters**

#### **ServiceName (MQCFST)**

Service name (parameter identifier: MQCA\_SERVICE\_NAME).

Generic service names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all services having names that start with the selected character string. An asterisk on its own matches all possible names.

The service name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

### **Optional parameters (Inquire Service Status)**

#### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ServiceStatusAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

#### **ServiceStatusAttrs (MQCFIL)**

Service status attributes (parameter identifier: MQIACF\_SERVICE\_STATUS\_ATTRS).

The attribute list can specify the following value on its own - is the default value used if the parameter is not specified:

##### **MQIACF\_ALL**

All attributes.

or a combination of the following:

##### **MQCA\_SERVICE\_DESC**

Description of service definition.

##### **MQCA\_SERVICE\_NAME**

Name of service definition.

##### **MQCA\_SERVICE\_START\_ARGS**

The arguments to pass to the service program.

##### **MQCA\_SERVICE\_START\_COMMAND**

The name of the program to run to start the service.

##### **MQCA\_SERVICE\_STOP\_ARGS**

The arguments to pass to the stop command to stop the service.

##### **MQCA\_SERVICE\_STOP\_COMMAND**

The name of the program to run to stop the service.

##### **MQCA\_STDERR\_DESTINATION**

Destination of standard error for the process.

##### **MQCA\_STDOUT\_DESTINATION**

Destination of standard output for the process.

**MQCACF\_SERVICE\_START\_DATE**

The date on which the service was started.

**MQCACF\_SERVICE\_START\_TIME**

The time at which the service was started.

**MQIA\_SERVICE\_CONTROL**

How the service is to be started and stopped.

**MQIA\_SERVICE\_TYPE**

The mode in which the service is to run.

**MQIACF\_PROCESS\_ID**

The process identifier of the operating system task under which this service is executing.

**MQIACF\_SERVICE\_STATUS**

Status of the service.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ServiceStatusAttrs* except MQCA\_SERVICE\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

**Error codes**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_SERV\_STATUS\_NOT\_FOUND**

Service status not found.

**Multi**

**MQCMD\_INQUIRE\_SERVICE\_STATUS (Inquire Service Status) Response on****Multiplatforms**

The response to the Inquire Service Status (MQCMD\_INQUIRE\_SERVICE\_STATUS) PCF command consists of the response header followed by the *ServiceName* structure and the requested combination of attribute parameter structures.

If a generic service name was specified, one such message is generated for each service found.

**Always returned:**

*ServiceName*

**Returned if requested:**

*ProcessId, ServiceDesc, StartArguments, StartCommand, StartDate, StartMode, StartTime, Status, StderrDestination, StdoutDestination, StopArguments, StopCommand*

**Response data****ProcessId (MQCFIN)**

Process identifier (parameter identifier: MQIACF\_PROCESS\_ID).

The operating system process identifier associated with the service.

**ServiceDesc (MQCFST)**

Description of service definition (parameter identifier: MQCACH\_SERVICE\_DESC).

The maximum length of the string is MQ\_SERVICE\_DESC\_LENGTH.

**ServiceName (MQCFST)**

Name of the service definition (parameter identifier: MQCA\_SERVICE\_NAME).

The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

**StartArguments (MQCFST)**

Arguments to be passed to the program on startup (parameter identifier: MQCA\_SERVICE\_START\_ARGS).

The maximum length of the string is MQ\_SERVICE\_ARGS\_LENGTH.

**StartCommand (MQCFST)**

Service program name (parameter identifier: MQCA\_SERVICE\_START\_COMMAND).

Specifies the name of the program which is to run.

The maximum length of the string is MQ\_SERVICE\_COMMAND\_LENGTH.

**StartDate (MQCFST)**

Start date (parameter identifier: MQIACF\_SERVICE\_START\_DATE).

The date, in the form yyyy-mm-dd, on which the service was started.

The maximum length of the string is MQ\_DATE\_LENGTH.

**StartMode (MQCFIN)**

Service mode (parameter identifier: MQIA\_SERVICE\_CONTROL).

How the service is to be started and stopped. The value can be:

**MQSVC\_CONTROL\_MANUAL**

The service is not to be started automatically or stopped automatically. It is to be controlled by user command.

**MQSVC\_CONTROL\_Q\_MGR**

The service is to be started and stopped at the same time as the queue manager is started and stopped.

**MQSVC\_CONTROL\_Q\_MGR\_START**

The service is to be started at the same time as the queue manager is started, but is not request to stop when the queue manager is stopped.

**StartTime (MQCFST)**

Start date (parameter identifier: MQIACF\_SERVICE\_START\_TIME).

The time, in the form hh.mm.ss, at which the service was started.

The maximum length of the string is MQ\_TIME\_LENGTH.

**Status (MQCFIN)**

Service status (parameter identifier: MQIACF\_SERVICE\_STATUS).

The status of the service. The value can be any of the following values:

**MQSVC\_STATUS\_STARTING**

The service is in the process of initializing.

**MQSVC\_STATUS\_RUNNING**

The service is running.

**MQSVC\_STATUS\_STOPPING**

The service is stopping.

**StderrDestination (MQCFST)**

Specifies the path to a file to which the standard error (stderr) of the service program is to be redirected (parameter identifier: MQCA\_STDERR\_DESTINATION).

The maximum length of the string is MQ\_SERVICE\_PATH\_LENGTH.

**StdoutDestination (MQCFST)**

Specifies the path to a file to which the standard output (stdout) of the service program is to be redirected (parameter identifier: MQCA\_STDOUT\_DESTINATION).

The maximum length of the string is MQ\_SERVICE\_PATH\_LENGTH.

**StopArguments (MQCFST)**

Specifies the arguments to be passed to the stop program when instructed to stop the service (parameter identifier: MQCA\_SERVICE\_STOP\_ARGS).

The maximum length of the string is MQ\_SERVICE\_ARGS\_LENGTH.

**StopCommand (MQCFST)**

Service program stop command (parameter identifier: MQCA\_SERVICE\_STOP\_COMMAND).

This parameter is the name of the program that is to run when the service is requested to stop.

The maximum length of the string is MQ\_SERVICE\_COMMAND\_LENGTH.

**z/OS MQCMD\_INQUIRE\_SMDS (Inquire SMDS) on z/OS**

The Inquire SMDS (MQCMD\_INQUIRE\_SMDS) PCF command inquires about the attributes of shared message data sets for a CF application structure.

**Required parameters****SMDS (qmgr\_name)**

Specifies the queue manager for which the shared message data set properties are to be displayed, or an asterisk to display the properties for all shared message data sets associated with the specified CFSTRUCT (parameter identifier: MQCACF\_CF\_SMDS).

**CFStrucName (MQCFST)**

The name of the CF application structure with SMDS properties that you want to inquire on (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

**Optional parameters****CFSMDSAttrs (MQCFIL)**

CF application structure SMDS attributes (parameter identifier: MQIACF\_SMDS\_ATTRS).

The default value used if this parameter is not specified is:

**MQIACF\_ALL**

All attributes.

The attribute list might specify MQIACF\_ALL on its own, or may specify a combination of the following:

**MQIA\_CF\_SMDS\_BUFFERS**

The shared message data set DSBUFS property.

**MQIACF\_CF\_SMDS\_EXPAND**

The shared message data set DSEXPAND property.

**z/OS MQCMD\_INQUIRE\_SMDS (Inquire SMDS) Response on z/OS**

The response to the Inquire SMDS (MQCMD\_INQUIRE\_SMDS) PCF command returns the attribute parameters of the shared message data set connection.

**Response data****SMDS (MQCFST)**

The queue manager name for which the shared message data set properties are displayed (parameter identifier: MQCACF\_CF\_SMDS).

**CFStrucName (MQCFST)**

CF Structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length is MQ\_CF\_STRUC\_NAME\_LENGTH.

**DSBUFS (MQCFIN)**

The CF DSBUFS property (parameter identifier: MQIA\_CF\_SMDS\_BUFFERS).

The returned value is in the range 0 - 9999.

The value is the number of buffers to be allocated in each queue manager for accessing shared message data sets. The size of each buffer is equal to the logical block size.

**DSEXPAND (MQCFIN)**

The CF DSEXPAND property (parameter identifier: MQIACF\_CF\_SMDS\_EXPAND).

**MQDSE\_YES**

The data set can be expanded.

**MQDSE\_NO**

The data set cannot be expanded.

**MQDSE\_DEFAULT**

Only returned on Inquire CF Struct when not explicitly set


**MQCMD\_INQUIRE\_SMDSCONN (Inquire SMDS Connection) on z/OS**

The response to the Inquire SMDS Connection (MQCMD\_INQUIRE\_SMDSCONN) PCF command returns status and availability information about the connection between the queue manager and the shared message data sets for the specified *CFStrucName*.

**Required parameters****SMDSCONN (MQCFST)**

Specify the queue manager which owns the SMDS for which the connection information is to be returned, or an asterisk to return the connection information for all shared message data sets associated with the specified *CFStrucName* (parameter identifier: MQCACF\_CF\_SMDSCONN).

**CFStrucName (MQCFST)**

The name of the CF application structure with SMDS connections properties that you want to inquire on (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

**CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## **MQCMD\_INQUIRE\_SMDSCONN (Inquire SMDS Connection) Response on**

### **z/OS**

The response to the Inquire SMDS Connection (MQCMD\_INQUIRE\_SMDSCONN) PCF command returns status and availability information about the connection between the queue manager and the shared message data sets for the specified *CFStrucName*.

### **Response data**

#### **SMDSCONN (MQCFST)**

The queue manager which owns the SMDS for which the connection information is returned (parameter identifier: MQCACF\_CF\_SMDSCONN).

#### **CFStrucName (MQCFST)**

The name of the CF application structure with SMDS connections properties that you want to inquire on (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

#### **Avail (MQCFIN)**

The availability of this data set connection as seen by this queue manager (parameter identifier MQIACF\_SMDS\_AVAIL).

This is one of the following values:

##### **MQS\_AVAIL\_NORMAL**

The connection can be used and no error has been detected.

##### **MQS\_AVAIL\_ERROR**

The connection is unavailable because of an error.

The queue manager may try to enable access again automatically if the error may no longer be present, for example when recovery completes or the status is manually set to RECOVERED. Otherwise, it can be enabled again using the START SMDSCONN command in order to retry the action which originally failed.

##### **MQS\_AVAIL\_STOPPED**

The connection cannot be used because it has been explicitly stopped using the STOP SMDSCONN command. It can only be made available again by using a START SMDSCONN command to enable it.

#### **ExpandST (MQCFIN)**

The data set automatic expansion status (parameter identifier MQIACF\_SMDS\_EXPANDST).

This is one of the following values:

##### **MQS\_EXPANDST\_NORMAL**

No problem has been noted which would affect automatic expansion.

##### **MQS\_EXPANDST\_FAILED**

A recent expansion attempt failed, causing the DSEXPAND option to be set to NO for this specific data set. This status is cleared when ALTER SMDS is used to set the DSEXPAND option back to YES or DEFAULT.

##### **MQS\_EXPANDST\_MAXIMUM**

The maximum number of extents has been reached, so future expansion is not possible (except by taking the data set out of service and copying it to larger extents).

#### **OpenMode (MQCFIN)**

Indicates the mode in which the shared message data set is currently open by this queue manager (parameter identifier MQIACF\_SMDS\_OPENMODE).

This is one of the following values:

##### **MQS\_OPENMODE\_NONE**

The shared message data set is not open.

**MQS\_OPENMODE\_READONLY**

The shared message data set is owned by another queue manager, and is open for read-only access.

**MQS\_OPENMODE\_UPDATE**

The shared message data set is owned by this queue manager, and is open for update access.

**MQS\_OPENMODE\_RECOVERY**

The shared message data set is open for recovery processing

**Status (MQCFIN)**

Indicates the shared message data set connection status as seen by this queue manager parameter identifier MQIACF\_SMDS\_STATUS).

This is one of the following values:

**MQS\_STATUS\_CLOSED**

This data set is not currently open.

**MQS\_STATUS\_CLOSING**

This queue manager is currently in the process of closing this data set, including quiescing normal I/O activity and storing the saved space map if necessary.

**MQS\_STATUS\_OPENING**

This queue manager is currently in the process of opening and validating this data set (including space map restart processing when necessary).

**MQS\_STATUS\_OPEN**

This queue manager has successfully opened this data set and it is available for normal use.

**MQS\_STATUS\_NOTENABLED**

The SMDS definition is not in the ACCESS(ENABLED) state so the data set is not currently available for normal use. This status is only set when the SMDSCONN status does not already indicate some other form of failure.

**MQS\_STATUS\_ALLOCFAIL**

This queue manager was unable to locate or allocate this data set.

**MQS\_STATUS\_OPENFAIL**

This queue manager was able to allocate the data set but was unable to open it, so it has now been deallocated.

**MQS\_STATUS\_STGFAIL**

The data set could not be used because the queue manager was unable to allocate associated storage areas for control blocks, or for space map or header record processing.

**MQS\_STATUS\_DATAFAIL**

The data set was successfully opened but the data was found to be invalid or inconsistent, or a permanent I/O error occurred, so it has now been closed and deallocated.

This might result in the shared message data set itself being marked as STATUS(FAILED).

## **MQCMD\_INQUIRE\_STG\_CLASS (Inquire Storage Class) on z/OS**

The Inquire Storage Class (MQCMD\_INQUIRE\_STG\_CLASS) PCF command returns information about storage classes.

**Required parameters****StorageClassName (MQCFST)**

Storage class name (parameter identifier: MQCA\_STORAGE\_CLASS).

Generic storage class names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all storage classes having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.



## Optional parameters

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

### IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *StgClassAttrs* except MQIACF\_ALL. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter for *PageSetId*, you cannot also specify the **PageSetId** parameter.

If you specify an integer filter, you cannot also specify a string filter using the **StringFilterCommand** parameter.

### PageSetId (MQCFIN)

Page set identifier that the storage class is associated with (parameter identifier: MQIA\_PAGESET\_ID).

If you omit this parameter, storage classes with any page set identifiers qualify.

### QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). The value can be:

#### MQQSGD\_LIVE

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

#### MQQSGD\_ALL

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### MQQSGD\_COPY

The object is defined as MQQSGD\_COPY.

#### MQQSGD\_GROUP

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

#### MQQSGD\_Q\_MGR

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_PRIVATE**

The object is defined with either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

**StgClassAttrs (MQCFIL)**

Storage class parameter attributes (parameter identifier: MQIACF\_STORAGE\_CLASS\_ATTRS).

The attribute list might specify the following value on its own - is the default value used if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

or a combination of the following:

**MQCA\_STORAGE\_CLASS**

Storage class name.

**MQCA\_STORAGE\_CLASS\_DESC**

Description of the storage class.

**MQIA\_PAGESET\_ID**

The page set identifier to which the storage class maps.

**MQCA\_XCF\_GROUP\_NAME**

The name of the XCF group of which IBM MQ is a member.

**MQIA\_XCF\_MEMBER\_NAME**

The XCF member name of the IMS system within the XCF group specified in MQCA\_XCF\_GROUP\_NAME.

**MQCA\_ALTERATION\_DATE**

The date on which the definition was last altered.

**MQCA\_ALTERATION\_TIME**


The time at which the definition was last altered.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *StgClassAttrs* except MQCA\_STORAGE\_CLASS. Use this parameter to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 1530 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the

**IntegerFilterCommand** parameter.

 **MQCMD\_INQUIRE\_STG\_CLASS (Inquire Storage Class) Response on z/OS**

The response to the Inquire Storage Class (MQCMD\_INQUIRE\_STG\_CLASS) PCF command consists of the response header followed by the *StgClassName* structure, the *PageSetId* structure and the *QSGDisposition* structure which are followed by the requested combination of attribute parameter structures.

**Always returned:**

*PageSetId, QSGDisposition, StgClassName*

**Returned if requested:**

*AlterationDate, AlterationTime, PassTicketApplication, StorageClassDesc, XCFGroupName, XCFMemberName,*

**Response data****AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

This parameter is the date, in the form yyyy-mm-dd, on which the definition was last altered.

The maximum length of the string is MQ\_DATE\_LENGTH.

**AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

This parameter is the time, in the form hh.mm.ss, at which the definition was last altered.

The maximum length of the string is MQ\_TIME\_LENGTH.

**PageSetId (MQCFIN)**

Page set identifier (parameter identifier: MQIA\_PAGESET\_ID).

The page set identifier to which the storage class maps.

**PassTicketApplication (MQCFST)**

PassTicket application (parameter identifier: MQCA\_PASS\_TICKET\_APPL).

The application name that is passed to RACF when authenticating the PassTicket specified in the MQIIH header.

The maximum length is MQ\_PASS\_TICKET\_APPL\_LENGTH.

**QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). The value can be any of the following values:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**StorageClassDesc (MQCFST)**

Description of the storage class (parameter identifier: MQCA\_STORAGE\_CLASS\_DESC).

The maximum length is MQ\_STORAGE\_CLASS\_DESC\_LENGTH.

**StgClassName (MQCFST)**

Name of the storage class (parameter identifier: MQCA\_STORAGE\_CLASS).

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.

**XCFGroupName (MQCFST)**

Name of the XCF group of which IBM MQ is a member (parameter identifier: MQCA\_XCF\_GROUP\_NAME).

The maximum length is MQ\_XCF\_GROUP\_NAME\_LENGTH.

**XCFMemberName (MQCFST)**

Name of the XCF group of which IBM MQ is a member (parameter identifier: MQCA\_XCF\_MEMBER\_NAME).

The maximum length is MQ\_XCF\_MEMBER\_NAME\_LENGTH.

 **MQCMD\_INQUIRE\_STG\_CLASS\_NAMES (Inquire Storage Class Names) on z/OS**

The Inquire Storage Class Names (MQCMD\_INQUIRE\_STG\_CLASS\_NAMES) PCF command inquires a list of storage class names that match the generic storage class name specified.

**Required parameters**

**StorageClassName (MQCFST)**

Storage class name (parameter identifier: MQCA\_STORAGE\_CLASS).

Generic storage class names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all storage classes having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_STORAGE\_CLASS\_LENGTH.

## Optional parameters

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

#### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

#### **MQQSGD\_PRIVATE**

The object is defined with either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

## **MQCMD\_INQUIRE\_STG\_CLASS\_NAMES (Inquire Storage Class Names)**

### **Response on z/OS**

The response to the Inquire Storage Class Names (MQCMD\_INQUIRE\_STG\_CLASS\_NAMES) PCF command consists of the response header followed by a parameter structure giving zero or more names that match the specified namelist name.

In addition to this, the *QSGDispositions* structure (with the same number of entries as the *StorageClassNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *StorageClassNames* structure.

#### **Always returned:**

*StorageClassNames, QSGDispositions*

#### **Returned if requested:**

None

### **Response data**

#### **StorageClassNames (MQCFSL)**

List of storage class names (parameter identifier: MQCACF\_STORAGE\_CLASS\_NAMES).

#### **QSGDispositions (MQCFIL)**

List of queue sharing group dispositions (parameter identifier: MQIACF\_QSG\_DISPS). Possible values for fields in this structure are those permitted for the *QSGDisposition* parameter (MQQSGD\_\*). Possible values for fields in this structure are:

##### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

##### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

##### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

## **MQCMD\_INQUIRE\_SUBSCRIPTION (Inquire Subscription)**

The Inquire Subscription (MQCMD\_INQUIRE\_SUBSCRIPTION) PCF command inquires about the attributes of a subscription.

### **Required parameters**

#### **SubName (MQCFST)**

The unique identifier of the application for a subscription (parameter identifier: MQCACF\_SUB\_NAME).

If *SubName* is not provided, *SubId* must be specified to identify the subscription to be inquired.

The maximum length of the string is MQ\_SUB\_NAME\_LENGTH.

#### **SubId (MQCFBS)**

Subscription identifier (parameter identifier: MQBACF\_SUB\_ID).

Specifies the unique internal subscription identifier. If the queue manager is generating the *CorrelId* for a subscription, then the *SubId* is used as the *DestinationCorrelId*.

You must supply a value for *SubId* if you have not supplied a value for *SubName*.

The maximum length of the string is MQ\_CORREL\_ID\_LENGTH.

### **Optional parameters**

#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- Blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- A queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- An asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

### **Durable (MQCFIN)**

Specify this attribute to restrict the type of subscriptions which are displayed (parameter identifier: MQIACF\_DURABLE\_SUBSCRIPTION).

#### **MQSUB\_DURABLE\_YES**

Information about durable subscriptions only is displayed.

#### **MQSUB\_DURABLE\_NO**

Information about nondurable subscriptions only is displayed.

#### **MQSUB\_DURABLE\_ALL**

Information about all subscriptions is displayed.

### **SubscriptionAttrs (MQCFIL)**

Subscription attributes (parameter identifier: MQIACF\_SUB\_ATTRS).

Use one of the following parameters to select the attributes you want to display:

- ALL to display all attributes.
- SUMMARY to display a subset of the attributes (see MQIACF\_SUMMARY for a list).
- Any of the following parameters individually or in combination.

#### **MQIACF\_ALL**

All attributes.

#### **MQIACF\_SUMMARY**

Use this parameter to display:

- MQBACF\_DESTINATION\_CORREL\_ID
- MQBACF\_SUB\_ID
- MQCACF\_DESTINATION
- MQCACF\_DESTINATION\_Q\_MGR
- MQCACF\_SUB\_NAME
- MQCA\_TOPIC\_STRING
- MQIACF\_SUB\_TYPE

#### **MQBACF\_ACCOUNTING\_TOKEN**

The accounting token passed by the subscriber for propagation into messages sent to this subscription in the AccountingToken field of the MQMD.

#### **MQBACF\_DESTINATION\_CORREL\_ID**

The CorrelId used for messages sent to this subscription.

#### **MQBACF\_SUB\_ID**

The internal unique key identifying a subscription.

#### **MQCA\_ALTERATION\_DATE**

The date of the most recent MQSUB with MQSO\_ALTER or ALTER SUB command.

**MQCA\_ALTERATION\_TIME**

The time of the most recent MQSUB with MQSO\_ALTER or ALTER SUB command.

**MQCA\_CREATION\_DATE**

The date of the first MQSUB command that caused this subscription to be created.

**MQCA\_CREATION\_TIME**

The time of the first MQSUB that caused this subscription to be created.

**MQCA\_TOPIC\_STRING**

The resolved topic string the subscription is for.

**MQCACF\_APPL\_IDENTITY\_DATA**

The identity data passed by the subscriber for propagation into messages sent to this subscription in the ApplIdentity field of the MQMD.

**MQCACF\_DESTINATION**

The destination for messages published to this subscription.

**MQCACF\_DESTINATION\_Q\_MGR**

The destination queue manager for messages published to this subscription.

**MQCACF\_SUB\_NAME**

The unique identifier of an application for a subscription.

**MQCACF\_SUB\_SELECTOR**

The SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

**MQCACF\_SUB\_USER\_DATA**

The user data associated with the subscription.

**MQCACF\_SUB\_USER\_ID**

The userid that owns the subscription. MQCACF\_SUB\_USER\_ID is either the userid associated with the creator of the subscription, or, if subscription takeover is permitted, the userid which last took over the subscription.

**MQCA\_TOPIC\_NAME**

The name of the topic object that identifies a position in the topic hierarchy to which the topic string is concatenated.

**MQIACF\_DESTINATION\_CLASS**

Indicated whether this subscription is a managed subscription.

**MQIACF\_DURABLE\_SUBSCRIPTION**

Whether the subscription is durable, persisting over queue manager restart.

**MQIACF\_EXPIRY**

The time to live from creation date and time.

**MQIACF\_PUB\_PRIORITY**

The priority of the messages sent to this subscription.

**MQIACF\_PUBSUB\_PROPERTIES**

The manner in which publish/subscribe related message properties are added to messages sent to this subscription.

**MQIACF\_REQUEST\_ONLY**

Indicates whether the subscriber polls for updates by using MQSUBRQ API, or whether all publications are delivered to this subscription.

**MQIACF\_SUB\_TYPE**

The type of subscription - how it was created.

**MQIACF\_SUBSCRIPTION\_SCOPE**

Whether the subscription forwards messages to all other queue managers directly connected by using a Publish/Subscribe collective or hierarchy, or the subscription forwards messages on this topic within this queue manager only.

**MQIACF\_SUB\_LEVEL**

The level within the subscription interception hierarchy at which this subscription is made.

**MQIACF\_VARIABLE\_USER\_ID**

Users other than the creator of this subscription that can connect to it (subject to topic and destination authority checks).

**MQIACF\_WILDCARD\_SCHEMA**

The schema to be used when interpreting wildcard characters in the topic string.

**MQIA\_DISPLAY\_TYPE**

Controls the output returned in the **TOPICSTR** and **TOPICOBJ** attributes.

**SubscriptionType (MQCFIN)**

Specify this attribute to restrict the type of subscriptions which are displayed (parameter identifier: MQIACF\_SUB\_TYPE).

**MQSUBTYPE\_ADMIN**

Subscriptions which have been created by an admin interface or modified by an admin interface are selected.

**MQSUBTYPE\_ALL**

All subscription types are displayed.

**MQSUBTYPE\_API**

Subscriptions created by applications by way of the IBM MQ API are displayed.

**MQSUBTYPE\_PROXY**

System created subscriptions relating to inter-queue manager subscriptions are displayed.

**MQSUBTYPE\_USER**

USER subscriptions (with SUBTYPE of either ADMIN or API) are displayed. MQSUBTYPE\_USER is the default value.

**DisplayType (MQCFIN)**

Controls the output returned in the **MQCA\_TOPIC\_STRING** and **MQCA\_TOPIC\_NAME** attributes (parameter identifier: MQIA\_DISPLAY\_TYPE).

**MQDOPT\_RESOLVED**

Returns the resolved (full) topic string in the **MQCA\_TOPIC\_STRING** attribute. The value of the **MQCA\_TOPIC\_NAME** attribute is also returned.

**MQDOPT\_DEFINED**

Returns the values of the **MQCA\_TOPIC\_NAME** and **MQCA\_TOPIC\_STRING** attributes provided when the subscription was created. The **MQCA\_TOPIC\_STRING** attribute will contain the application part of the topic string only. You can use the values returned with **MQCA\_TOPIC\_NAME** and **MQCA\_TOPIC\_STRING** to fully re-create the subscription by using **MQDOPT\_DEFINED**.

**MQCMD\_INQUIRE\_SUBSCRIPTION (Inquire Subscription) Response**

The response to the Inquire Subscription (MQCMD\_INQUIRE\_SUBSCRIPTION) PCF command consists of the response header followed by the *SubId* and *SubName* structures, and the requested combination of attribute parameter structures (where applicable).

**Always returned**

*SubID, SubName*

**Returned if requested**

*AlterationDate, AlterationTime, CreationDate, CreationTime, Destination, DestinationClass, DestinationCorrelId, DestinationQueueManager, Expiry, PublishedAccountingToken, PublishedApplicationIdentityData, PublishPriority, PublishSubscribeProperties, Requestonly, Selector, SelectorType, SubscriptionLevel, SubscriptionScope, SubscriptionType, SubscriptionUser, TopicObject, TopicString, Userdata, VariableUser, WildcardSchema*

**Response Data****AlterationDate (MQCFST)**

The date of the most recent **MQSUB** or **Change Subscription** command that modified the properties of the subscription (parameter identifier: MQCA\_ALTERATION\_DATE).



**AlterationTime (MQCFST)**

The time of the most recent **MQSUB** or **Change Subscription** command that modified the properties of the subscription (parameter identifier: MQCA\_ALTERATION\_TIME).

**CreationDate (MQCFST)**

The creation date of the subscription, in the form yyyy-mm-dd (parameter identifier: MQCA\_CREATION\_DATE).

**CreationTime (MQCFST)**

The creation time of the subscription, in the form hh.mm.ss (parameter identifier: MQCA\_CREATION\_TIME).

**Destination (MQCFST)**

Destination (parameter identifier: MQCACF\_DESTINATION).

Specifies the name of the alias, local, remote, or cluster queue to which messages for this subscription are put.

**DestinationClass (MQCFIN)**

Destination class (parameter identifier: MQIACF\_DESTINATION\_CLASS).

Whether the destination is managed.

The value can be any of the following values:

**MQDC\_MANAGED**

The destination is managed.

**MQDC\_PROVIDED**

The destination queue is as specified in the *Destination* field.

**DestinationCorrelId (MQCFBS)**

Destination correlation identifier (parameter identifier: MQBACF\_DESTINATION\_CORREL\_ID).

A correlation identifier that is placed in the *CorrelId* field of the message descriptor for all the messages sent to this subscription.

The maximum length is MQ\_CORREL\_ID\_LENGTH.

**DestinationQueueManager (MQCFST)**

Destination queue manager (parameter identifier: MQCACF\_DESTINATION\_Q\_MGR).

Specifies the name of the destination queue manager, either local or remote, to which messages for the subscription are forwarded.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

**DisplayType (MQCFIN)**

The type of output requested for **MQCA\_TOPIC\_STRING** and **MQCA\_TOPIC\_NAME** is returned (parameter identifier: MQIA\_DISPLAY\_TYPE).

**MQDOPT\_RESOLVED**

Returns the resolved (full) topic string in the **MQCA\_TOPIC\_STRING** attribute. The value of the **MQCA\_TOPIC\_NAME** attribute is also returned.

**MQDOPT\_DEFINED**

The application portion of the topic string is returned in the **MQCA\_TOPIC\_STRING** attribute. **MQCA\_TOPIC\_NAME** contains the name of the **TOPIC** Object used when defining the subscription.

**Durable (MQCFIN)**

Whether this subscription is a durable subscription (parameter identifier: MQIACF\_DURABLE\_SUBSCRIPTION).

The value can be any of the following values:

**MQSUB\_DURABLE\_YES**

The subscription persists, even if the creating application disconnects from the queue manager or issues an MQCLOSE call for the subscription. The queue manager reinstates the subscription during restart.

**MQSUB\_DURABLE\_NO**

The subscription is non-durable. The queue manager removes the subscription when the creating application disconnects from the queue manager or issues an MQCLOSE call for the subscription. If the subscription has a destination class (DESTCLAS) of MANAGED, the queue manager removes any messages not yet consumed when it closes the subscription.

**Expiry (MQCFIN)**

The time, in tenths of a second, at which a subscription expires after its creation date and time (parameter identifier: MQIACF\_EXPIRY).

A value of unlimited means that the subscription never expires.

After a subscription has expired it becomes eligible to be discarded by the queue manager and receives no further publications.

**PublishedAccountingToken (MQCFBS)**

Value of the accounting token used in the *AccountingToken* field of the message descriptor (parameter identifier: MQBACF\_ACCOUNTING\_TOKEN).

The maximum length of the string is MQ\_ACCOUNTING\_TOKEN\_LENGTH.

**PublishedApplicationIdentityData (MQCFST)**

Value of the application identity data used in the *AppIdentityData* field of the message descriptor (parameter identifier: MQCACF\_APPL\_IDENTITY\_DATA).

The maximum length of the string is MQ\_APPL\_IDENTITY\_DATA\_LENGTH.

**PublishPriority (MQCFIN)**

The priority of messages sent to this subscription (parameter identifier: MQIACF\_PUB\_PRIORITY).

The value can be any of the following values:

**MQPRI\_PRIORITY\_AS\_PUBLISHED**

The priority of messages sent to this subscription is taken from that priority supplied to the published message. MQPRI\_PRIORITY\_AS\_PUBLISHED is the supplied default value.

**MQPRI\_PRIORITY\_AS\_QDEF**

The priority of messages sent to this subscription is determined by the default priority of the queue defined as a destination.

**0-9**

An integer value providing an explicit priority for messages sent to this subscription.

**PublishSubscribeProperties (MQCFIN)**

Specifies how publish/subscribe related message properties are added to messages sent to this subscription (parameter identifier: MQIACF\_PUBSUB\_PROPERTIES).

The value can be any of the following values:

**MQPSPROP\_NONE**

Publish/subscribe properties are not added to the messages. MQPSPROP\_NONE is the supplied default value.

**MQPSPROP\_MSGPROP**

Publish/subscribe properties are added as PCF attributes.

**MQPSPROP\_COMPAT**

If the original publication is a PCF message, then the publish/subscribe properties are added as PCF attributes. Otherwise, publish/subscribe properties are added within an MQRFH version 1 header. This method is compatible with applications coded for use with previous versions of IBM MQ.

**MQPSPROP\_RFH2**

Publish/subscribe properties are added within an MQRFH version 2 header. This method is compatible with applications coded for use with IBM Integration Bus brokers.

**Requestonly (MQCFIN)**

Indicates whether the subscriber polls for updates using the MQSUBRQ API call, or whether all publications are delivered to this subscription (parameter identifier: MQIACF\_REQUEST\_ONLY).

The value can be:

**MQRU\_PUBLISH\_ALL**

All publications on the topic are delivered to this subscription.

**MQRU\_PUBLISH\_ON\_REQUEST**

Publications are only delivered to this subscription in response to an MQSUBRQ API call.

**Selector (MQCFST)**

Specifies the selector applied to messages published to the topic (parameter identifier: MQCACF\_SUB\_SELECTOR).

Only those messages that satisfy the selection criteria are put to the destination specified by this subscription.

**SelectorType (MQCFIN)**

The type of selector string that has been specified (parameter identifier: MQIACF\_SELECTOR\_TYPE).

The value can be any of the following values:

**MQSELTYPE\_NONE**

No selector has been specified.

**MQSELTYPE\_STANDARD**

The selector references only the properties of the message, not its content, using the standard IBM MQ selector syntax. Selectors of this type are to be handled internally by the queue manager.

**MQSELTYPE\_EXTENDED**

The selector uses extended selector syntax, typically referencing the content of the message. Selectors of this type cannot be handled internally by the queue manager; extended selectors can be handled only by another program, such as IBM Integration Bus.

**SubID (MQCFBS)**

The internal, unique key identifying a subscription (parameter identifier: MQBACF\_SUB\_ID).

**SubscriptionLevel (MQCFIN)**

The level within the subscription interception hierarchy at which this subscription is made (parameter identifier: MQIACF\_SUB\_LEVEL).

The value can be:

**0 - 9**

An integer in the range 0-9. The default value is 1. Subscribers with a subscription level of 9 will intercept publications before they reach subscribers with lower subscription levels.

**SubscriptionScope (MQCFIN)**

Determines whether this subscription is passed to other queue managers in the network (parameter identifier: MQIACF\_SUBSCRIPTION\_SCOPE).

The value can be:

**MQTSCOPE\_ALL**

The subscription is forwarded to all queue managers directly connected through a publish/subscribe collective or hierarchy. MQTSCOPE\_ALL is the supplied default value.

**MQTSCOPE\_QMGR**

The subscription only forwards messages published on the topic within this queue manager.

**SubscriptionType (MQCFIN)**

Indicates how the subscription was created (parameter identifier: MQIACF\_SUB\_TYPE).

**MQSUBTYPE\_PROXY**

An internally created subscription used for routing publications through a queue manager.

**MQSUBTYPE\_ADMIN**

Created using **DEF SUB** MQSC or PCF command. This **SUBTYPE** also indicates that a subscription has been modified using an administrative command.

**MQSUBTYPE\_API**

Created using an **MQSUB** API request.

**SubscriptionUser (MQCFST)**

The userid that 'owns' this subscription. This parameter is either the userid associated with the creator of the subscription, or, if subscription takeover is permitted, the userid which last took over the subscription. (parameter identifier: MQCACF\_SUB\_USER\_ID).

The maximum length of the string is MQ\_USER\_ID\_LENGTH.

**TopicObject (MQCFST)**

The name of a previously defined topic object from which is obtained the topic name for the subscription (parameter identifier: MQCA\_TOPIC\_NAME).

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

**TopicString (MQCFST)**

The resolved topic string (parameter identifier: MQCA\_TOPIC\_STRING).

The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

**Userdata (MQCFST)**

User data (parameter identifier: MQCACF\_SUB\_USER\_DATA).

Specifies the user data associated with the subscription

The maximum length of the string is MQ\_USER\_DATA\_LENGTH.

**VariableUser (MQCFIN)**

Specifies whether a user other than the one who created the subscription, that is, the user shown in *SubscriptionUser* can take over the ownership of the subscription (parameter identifier: MQIACF\_VARIABLE\_USER\_ID).

The value can be any of the following values:

**MQVU\_ANY\_USER**

Any user can take over the ownership. MQVU\_ANY\_USER is the supplied default value.

**MQVU\_FIXED\_USER**

No other user can take over the ownership.

**WildcardSchema (MQCFIN)**

Specifies the schema to be used when interpreting any wildcard characters contained in the *TopicString* (parameter identifier: MQIACF\_WILDCARD\_SCHEMA).

The value can be any of the following values:

**MQWS\_CHAR**

Wildcard characters represent portions of strings; it is for compatibility with IBM MQ V6.0 broker.

**MQWS\_TOPIC**

Wildcard characters represent portions of the topic hierarchy; this is for compatibility with IBM Integration Bus brokers. MQWS\_TOPIC is the supplied default value.

## ***MQCMD\_INQUIRE\_SUB\_STATUS (Inquire Subscription Status)***

The Inquire Subscription Status (MQCMD\_INQUIRE\_SUB\_STATUS) PCF command inquires about the status of a subscription.

### **Required parameters**

#### **SubName (MQCFST)**

The unique identifier of an application for a subscription (parameter identifier: MQCACF\_SUB\_NAME).

If *SubName* is not provided, *SubId* must be specified to identify the subscription to be inquired.

The maximum length of the string is MQ\_SUB\_NAME\_LENGTH.

#### **SubId (MQCFBS)**

Subscription identifier (parameter identifier: MQBACF\_SUB\_ID).

Specifies the unique internal subscription identifier. If the queue manager is generating the CorrelId for a subscription, then the *SubId* is used as the *DestinationCorrelId*.

You must supply a value for *SubId* if you have not supplied a value for *SubName*.

The maximum length of the string is MQ\_CORREL\_ID\_LENGTH.

### **Optional parameters**

#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- Blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- A queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- An asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter on which to filter.

#### **Durable (MQCFIN)**

Specify this attribute to restrict the type of subscriptions which are displayed (parameter identifier: MQIACF\_DURABLE\_SUBSCRIPTION).

##### **MQSUB\_DURABLE\_YES**

Information about durable subscriptions only is displayed. MQSUB\_DURABLE\_YES is the default.

##### **MQSUB\_DURABLE\_NO**

Information about non-durable subscriptions only is displayed.

#### **SubscriptionType (MQCFIN)**

Specify this attribute to restrict the type of subscriptions which are displayed (parameter identifier: MQIACF\_SUB\_TYPE).

##### **MQSUBTYPE\_ADMIN**

Subscriptions which have been created by an admin interface or modified by an admin interface are selected.

##### **MQSUBTYPE\_ALL**

All subscription types are displayed.

**MQSUBTYPE\_API**

Subscriptions created by applications through an IBM MQ API call are displayed.

**MQSUBTYPE\_PROXY**

System created subscriptions relating to inter-queue manager subscriptions are displayed.

**MQSUBTYPE\_USER**

USER subscriptions (with SUBTYPE of either ADMIN or API) are displayed. MQSUBTYPE\_USER is the default value.

**StatusAttrs (MQCFIL)**

Subscription status attributes (parameter identifier: MQIACF\_SUB\_STATUS\_ATTRS).

To select the attributes you want to display you can specify;

- ALL to display all attributes.
- any of the following parameters individually or in combination.

**MQIACF\_ALL**

All attributes.

**MQBACF\_CONNECTION\_ID**

The currently active *ConnectionID* that has opened the subscription.

**MQIACF\_DURABLE\_SUBSCRIPTION**

Whether the subscription is durable, persisting over queue manager restart.

**MQCACF\_LAST\_MSG\_DATE**

The date that a message was last sent to the destination specified by the subscription.

**MQCACF\_LAST\_MSG\_TIME**

The time when a message was last sent to the destination specified by the subscription.

**MQIACF\_MESSAGE\_COUNT**

The number of messages put to the destination specified by the subscription.

**MQCA\_RESUME\_DATE**

The date of the most recent MQSUB command that connected to the subscription.

**MQCA\_RESUME\_TIME**

The time of the most recent MQSUB command that connected to the subscription.

**MQIACF\_SUB\_TYPE**

The type of subscription - how it was created.

**MQCACF\_SUB\_USER\_ID**

The userid owns the subscription.

**MQCA\_TOPIC\_STRING**

Returns the fully resolved topic string of the subscription.

**MQCMD\_INQUIRE\_SUB\_STATUS (Inquire Subscription Status) Response**

The response to the Inquire Subscription Status (MQCMD\_INQUIRE\_SUB\_STATUS) PCF command consists of the response header followed by the *SubId* and *SubName* structures, and the requested combination of attribute parameter structures (where applicable).

**Always returned**

*SubID, SubName*

**Returned if requested**

*ActiveConnection, Durable, LastPublishDate, LastPublishTime, MCastRelIndicator, NumberMsgs, ResumeDate, ResumeTime, SubType, TopicString*

**Response Data****ActiveConnection (MQCFBS)**

The *ConnId* of the *HConn* that currently has this subscription open (parameter identifier: MQBACF\_CONNECTION\_ID).

**Durable (MQCFIN)**

A durable subscription is not deleted when the creating application closes its subscription handle (parameter identifier: MQIACF\_DURABLE\_SUBSCRIPTION).

**MQSUB\_DURABLE\_NO**

The subscription is removed when the application that created it is closed or disconnected from the queue manager.

**MQSUB\_DURABLE\_YES**

The subscription persists even when the creating application is no longer running or has been disconnected. The subscription is reinstated when the queue manager restarts.

**LastMessageDate (MQCFST)**

The date that a message was last sent to the destination specified by the subscription (parameter identifier: MQCACF\_LAST\_MSG\_DATE).

**LastMessageTime (MQCFST)**

The time when a message was last sent to the destination specified by the subscription (parameter identifier: MQCACF\_LAST\_MSG\_TIME).

**MCastRelIndicator (MQCFIN)**

The multicast reliability indicator (parameter identifier: MQIACF\_MCAST\_REL\_INDICATOR).

**NumberMsgs (MQCFIN)**

The number of messages put to the destination specified by this subscription (parameter identifier: MQIACF\_MESSAGE\_COUNT).

**ResumeDate (MQCFST)**

The date of the most recent **MQSUB** API call that connected to the subscription (parameter identifier: MQCA\_RESUME\_DATE).

**ResumeTime (MQCFST)**

The time of the most recent **MQSUB** API call that connected to the subscription (parameter identifier: MQCA\_RESUME\_TIME).

**SubscriptionUser (MQCFST)**

The userid that 'owns' this subscription. This parameter is either the userid associated with the creator of the subscription, or, if subscription takeover is permitted, the userid which last took over the subscription. (parameter identifier: MQCACF\_SUB\_USER\_ID).

The maximum length of the string is MQ\_USER\_ID\_LENGTH.

**SubID (MQCFBS)**

The internal, unique key identifying a subscription (parameter identifier: MQBACF\_SUB\_ID).

**SubName (MQCFST)**

The unique identifier of a subscription (parameter identifier: MQCACF\_SUB\_NAME).

**SubType (MQCFIN)**

Indicates how the subscription was created (parameter identifier: MQIACF\_SUB\_TYPE).

**MQSUBTYPE\_PROXY**

An internally created subscription used for routing publications through a queue manager.

**MQSUBTYPE\_ADMIN**

Created using the **DEF SUB MQSC** or **Create Subscription** PCF command. This Subtype also indicates that a subscription has been modified using an administrative command.

**MQSUBTYPE\_API**

Created using an **MQSUB** API call.

**TopicString (MQCFST)**

The resolved topic string (parameter identifier: MQCA\_TOPIC\_STRING). The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

## **MQCMD\_INQUIRE\_SYSTEM (Inquire System) on z/OS**

The Inquire System (MQCMD\_INQUIRE\_SYSTEM) PCF command returns general system parameters and information.

### Optional parameters

#### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## **MQCMD\_INQUIRE\_SYSTEM (Inquire System) Response on z/OS**

The response to the Inquire System (MQCMD\_INQUIRE\_SYSTEM) PCF command consists of the response header followed by the *ParameterType* structure and the combination of attribute parameter structures determined by the value of the parameter type.

#### Always returned:

*ParameterType*

Possible values of *ParameterType* are:


#### **MQSYSP\_TYPE\_INITIAL**

The initial settings of the system parameters.

#### **MQSYSP\_TYPE\_SET**

The settings of the system parameters if they have been altered since their initial setting.

#### Returned if *ParameterType* is **MQSYSP\_TYPE\_INITIAL** or **MQSYSP\_TYPE\_SET** (and a value is set):

 From IBM MQ for z/OS 9.3.0, the following parameters are returned: *CheckpointCount*, *ClusterCacheType*, *CodedCharSetId*, *CommandUserId*, *DB2BlobTasks*, *DB2Name*, *DB2Tasks*, *DSGName*, *Exclmsg*, *ExitInterval*, *ExitTasks*, *MaximumAcePool*, *MULCCapture*, *OTMADruExit*, *OTMAGroup*, *OTMAInterval*, *OTMAMember*, *OTMStpipePrefix*, *QIndexDefer*, *QSGName*, *RESLEVELAudit*, *RoutingCode*, *Service*, *SMFAccounting*, *SMFAcctIntervalMins*, *SMFAcctIntervalSecs*, *SMFStatistics*, *SMFstatsIntervalMins*, *SMFstatsIntervalSecs*, *Splcap*, *TraceClass*, *TraceSize*, *WLMInterval*, *WLMIntervalUnits*

### Response data

#### CheckpointCount (MQCFIN)

The number of log records written by IBM MQ between the start of one checkpoint and the next (parameter identifier: MQIACF\_SYSP\_CHKPOINT\_COUNT).

#### ClusterCacheType (MQCFIN)

The type of the cluster cache (parameter identifier: MQIACF\_SYSP\_CLUSTER\_CACHE).

The value can be any of the following values:

#### **MQCLCT\_STATIC**

Static cluster cache.



**MQCLCT\_DYNAMIC**

Dynamic cluster cache.

**CodedCharSetId (MQCFIN)**

Archive retention period (parameter identifier: MQIA\_CODED\_CHAR\_SET\_ID).

The coded character set identifier for the queue manager.

**CommandUserId (MQCFST)**

Command user ID (parameter identifier: MQCACF\_SYSP\_CMD\_USER\_ID).

Specifies the default user ID for command security checks.

The maximum length of the string is MQ\_USER\_ID\_LENGTH.

**DB2BlobTasks (MQCFIN)**

The number of Db2 server tasks to be used for BLOBs (parameter identifier: MQIACF\_SYSP\_DB2\_BLOB\_TASKS).

**DB2Name (MQCFST)**

The name of the Db2 subsystem or group attachment to which the queue manager is to connect (parameter identifier: MQCACF\_DB2\_NAME).

The maximum length of the string is MQ\_DB2\_NAME\_LENGTH.

**DB2Tasks (MQCFIN)**

The number of Db2 server tasks to use (parameter identifier: MQIACF\_SYSP\_DB2\_TASKS).

**DSGName (MQCFST)**

The name of the Db2 data-sharing group to which the queue manager is to connect (parameter identifier: MQCACF\_DSG\_NAME).

The maximum length of the string is MQ\_DSG\_NAME\_LENGTH.

**Exclmsg (MQCFSL)**

A list of message identifiers to be excluded from being written to any log (parameter identifier: MQCACF\_EXCL\_OPERATOR\_MESSAGES).

The maximum length of each message identifier is MQ\_OPERATOR\_MESSAGE\_LENGTH.

The list can contain a maximum of 16 message identifiers.

**ExitInterval (MQCFIN)**

The time, in seconds, for which queue manager exits can execute during each invocation (parameter identifier: MQIACF\_SYSP\_EXIT\_INTERVAL).

**ExitTasks (MQCFIN)**

Specifies how many started server tasks to use to run queue manager exits (parameter identifier: MQIACF\_SYSP\_EXIT\_TASKS).

**MaximumAcePool (MQCFIN)**

The maximum ACE storage pool size in 1 KB blocks (parameter identifier: MQIACF\_SYSP\_MAX\_ACE\_POOL).

**MULCCapture (MQCFIN)**

The Measured Usage Pricing property is used to control the algorithm for gathering data used by Measured Usage License Charging (MULC) (parameter identifier: MQIACF\_MULC\_CAPTURE).

The returned values can be MQMULC\_STANDARD or MQMULC\_REFINED.

**OTMADruExit (MQCFST)**

The name of the OTMA destination resolution user exit to be run by IMS (parameter identifier: MQCACF\_SYSP\_OTMA\_DRU\_EXIT).

The maximum length of the string is MQ\_EXIT\_NAME\_LENGTH.

**OTMAGroup (MQCFST)**

The name of the XCF group to which this instance of IBM MQ belongs (parameter identifier: MQCACF\_SYSP\_OTMA\_GROUP).

The maximum length of the string is MQ\_XCF\_GROUP\_NAME\_LENGTH.

**OTMAInterval (MQCFIN)**

The length of time, in seconds, that a user ID from IBM MQ is considered previously verified by IMS (parameter identifier: MQIACF\_SYSP\_OTMA\_INTERVAL).

**OTMAMember (MQCFST)**

The name of the XCF member to which this instance of IBM MQ belongs (parameter identifier: MQCACF\_SYSP\_OTMA\_MEMBER).

The maximum length of the string is MQ\_XCF\_MEMBER\_NAME\_LENGTH.

**OTMSTpipePrefix (MQCFST)**

The prefix to be used for Tpipe names (parameter identifier: MQCACF\_SYSP\_OTMA\_TPIPE\_PFX).

The maximum length of the string is MQ\_TPIPE\_PFX\_LENGTH.

**QIndexDefer (MQCFIN)**

Specifies whether queue manager restart completes before all indexes are built deferring building to later, or waits until all indexes are built (parameter identifier: MQIACF\_SYSP\_Q\_INDEX\_DEFER).

The value can be any of the following values:

**MQSYSP\_YES**

Queue manager restart completes before all indexes are built.

**MQSYSP\_NO**

Queue manager restart waits until all indexes are built.

**QSGName (MQCFST)**

The name of the queue sharing group to which the queue manager belongs (parameter identifier: MQCA\_QSG\_NAME).

The maximum length of the string is MQ\_QSG\_NAME\_LENGTH.

**RESLEVELAudit (MQCFIN)**

Specifies whether RACF audit records are written for RESLEVEL security checks performed during connection processing (parameter identifier: MQIACF\_SYSP\_RESLEVEL\_AUDIT).

The value can be any of the following values:

**MQSYSP\_YES**

RACF audit records are written.

**MQSYSP\_NO**

RACF audit records are not written.

**RoutingCode (MQCFIL)**

z/OS routing code list (parameter identifier: MQIACF\_SYSP\_ROUTING\_CODE).

Specifies the list of z/OS routing codes for messages that are not sent in direct response to an MQSC command. There can be in the range 1 through 16 entries in the list.

**Service (MQCFST)**

Service parameter setting (parameter identifier: MQCACF\_SYSP\_SERVICE).

The maximum length of the string is MQ\_SERVICE\_NAME\_LENGTH.

**SMFAccounting (MQCFIN)**

Specifies whether IBM MQ sends accounting data to SMF automatically when the queue manager starts (parameter identifier: MQIACF\_SYSP\_SMF\_ACCOUNTING).

The value can be any of the following values:

**MQSYSP\_YES**

Accounting data is sent automatically.

**MQSYSP\_NO**

Accounting data is not sent automatically.

### **V 9.3.0 SMFAcctIntervalMins (MQCFIN)**

From IBM MQ for z/OS 9.2.4 onwards, the minutes value of the default time between each gathering of accounting data (parameter identifier: MQIACF\_SYSP\_SMF\_ACCT\_TIME\_MINS).

### **V 9.3.0 SMFAcctIntervalSecs (MQCFIN)**

From IBM MQ for z/OS 9.2.4 onwards, the seconds value of the default time between each gathering of accounting data (parameter identifier: MQIACF\_SYSP\_SMF\_ACCT\_TIME\_SECS).

### **SMFInterval (MQCFIN)**

The default time, in minutes, between each gathering of statistics (parameter identifier: MQIACF\_SYSP\_SMF\_INTERVAL).

### **SMFStatistics (MQCFIN)**

Specifies whether IBM MQ sends statistics data to SMF automatically when the queue manager starts (parameter identifier: MQIACF\_SYSP\_SMF\_STATS).

The value can be any of the following values:

#### **MQSYSP\_YES**

Statistics data is sent automatically.

#### **MQSYSP\_NO**

Statistics data is not sent automatically.

### **V 9.3.0 SMFStatsIntervalMins (MQCFIN)**

From IBM MQ for z/OS 9.2.4 onwards, the minutes value of the default time between each gathering of statistics data (parameter identifier: MQIACF\_SYSP\_SMF\_STAT\_TIME\_MINS and parameter identifier: MQIACF\_SYSP\_SMF\_INTERVAL).

### **V 9.3.0 SMFStatsIntervalSecs (MQCFIN)**

From IBM MQ for z/OS 9.2.4 onwards, the seconds value of the default time between each gathering of statistics data (parameter identifier: MQIACF\_SYSP\_SMF\_STAT\_TIME\_SECS).

### **Splcap (MQCFIN)**

If the AMS component is installed for the version of IBM MQ that the queue manager is running under, the attribute has a value YES (MQCAP\_SUPPORTED). If the AMS component is not installed, the value is NO (MQCAP\_NOT\_SUPPORTED) (parameter identifier MQIA\_PROT\_POLICY\_CAPABILITY).

The value can be one of the following values:

#### **MQCAP\_SUPPORTED**

If the AMS component is installed for the version of IBM MQ that the queue manager is running under.

#### **MQCAP\_NOT\_SUPPORTED**

If the AMS component is not installed.

### **TraceClass (MQCFIL)**

Classes for which tracing is started automatically (parameter identifier: MQIACF\_SYSP\_TRACE\_CLASS). There can be in the range 1 through 4 entries in the list.

### **TraceSize (MQCFIN)**

The size of the trace table, in 4 KB blocks, to be used by the global trace facility (parameter identifier: MQIACF\_SYSP\_TRACE\_SIZE).

### **WLMInterval (MQCFIN)**

The time between scans of the queue index for WLM-managed queues (parameter identifier: MQIACF\_SYSP\_WLM\_INTERVAL).

### **WLMIntervalUnits (MQCFIN)**

Whether the value of *WLMInterval* is given in seconds or minutes (parameter identifier: MQIACF\_SYSP\_WLM\_INT\_UNITS). The value can be any of the following values:

#### **MQTIME\_UNITS\_SEC**

The value of *WLMInterval* is given in seconds.

## MQTIME\_UNITS\_MINS

The value of *WLMInterval* is given in minutes.

## MQCMD\_INQUIRE\_TOPIC (Inquire Topic)

The Inquire Topic (MQCMD\_INQUIRE\_TOPIC) PCF command inquires about the attributes of existing IBM MQ administrative topic objects

### Required parameters

#### TopicName (MQCFST)

Administrative topic object name (parameter identifier: MQCA\_TOPIC\_NAME).

Specifies the name of the administrative topic object about which information is to be returned. Generic topic object names are supported. A generic name is a character string followed by an asterisk (\*). For example, ABC\* selects all administrative topic objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

### Optional parameters

V 9.3.1

Multi

#### CapExpiry (MQCFIN)

Capped expiry processing (parameter identifier MQIA\_CAP\_EXPIRY) which can be an integer value or take values of MQCEX\_NOLIMIT or MQCEX\_AS\_PARENT.

Specifies a lifetime limit for messages put using the object, expressed in 10ths of a second. A value of -1, displayed as NOLIMIT, has no effect on processing.

Note that in addition to the CapExpiry attribute itself, you can use an attribute of the **CUSTOM** parameter, which is a string. Therefore passing a string PCF parameter MQCA\_CUSTOM, has a string value of CAPEXPY(*integer*).

CapExpiry provides, or limits, the value in the MQMD [Expiry](#) field of any message put.

An application provided MQMD **Expiry** that is less than any resolved CapExpiry value is passed. This value is not replaced by that resolved CapExpiry value.

This process allows an IBM MQ administrator to limit the life of messages put by an application that overlooked (or was unable to provide, in the case of MQTT) message expiration criteria.

However, this option does not allow an administrator to override application behavior where the required lifetime of messages was under-estimated.

If you specify **CAPEXPY(ASPARENT)**, the value is established by moving up the tree towards the root, until the point of first resolution to a non *ASPARENT* value. Additionally you can use the **CUSTOM CAPEXPY** option. See [Making CAPEXPY a first-class MQSC attribute in MQ 9.3.1](#) for more information.

As for queues, the smallest **CAPEXPY** found during a put operation, is used. Further limiting can be set for specific subscribers, based on **CAPEXPY** resolution applied on the path to resolve the destination for a subscription.

The new capped value for expiry is used during the put processing as if it had been provided by the application in the MQMD structure.

The *capped* value is evaluated for each put being performed, and so is sensitive to the resolution of the put operation. For example, in a cluster, where the put operation is performed with BIND NOT FIXED, messages might pick up different expiry values depending on the CapExpiry value set for the transmission queue used by the channel.

#### ClusterInfo (MQCFIN)

Cluster information (parameter identifier: MQIACF\_CLUSTER\_INFO).

This parameter requests that, in addition to information about attributes of topics defined on this queue manager, cluster information about these topics and other topics in the repository that match the selection criteria is returned.

In this case, there might be multiple topics with the same name returned.

You can set this parameter to any integer value: the value used does not affect the response to the command.

The cluster information is obtained locally from the queue manager.

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

### **IntegerFilterCommand (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *TopicAttrs* except MQIACF\_ALL.

Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFIF - PCF integer filter parameter” on page 1523](#) for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the

**StringFilterCommand** parameter.

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

#### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**MQQSGD\_PRIVATE**

The object is defined as either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

**StringFilterCommand (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *TopicAttrs* except MQCA\_TOPIC\_NAME. Use this parameter to restrict the output from the command by specifying a filter condition. See [“MQCFSF - PCF string filter parameter” on page 1530](#) for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

**TopicAttrs (MQCFIL)**

Topic object attributes (parameter identifier: MQIACF\_TOPIC\_ATTRS).

The attribute list can specify the following value on its own - default value if the parameter is not specified:

**MQIACF\_ALL**

All attributes.

or a combination of the following:

**MQCA\_ALTERATION\_DATE**

The date on which the information was last altered.

**MQCA\_ALTERATION\_TIME**

The time at which the information was last altered.

**MQCA\_CLUSTER\_NAME**

The cluster that is to be used for the propagation of publications and subscription to publish/subscribe cluster-connected queue managers for this topic.

**MQCA\_CLUSTER\_DATE**

The date on which this information became available to the local queue manager.

**MQCA\_CLUSTER\_TIME**

The time at which this information became available to the local queue manager.

**MQCA\_CLUSTER\_Q\_MGR\_NAME**

Queue manager that hosts the topic.

**MQCA\_CUSTOM**

The custom attribute for new features.

**MQCA\_MODEL\_DURABLE\_Q**

Name of the model queue for durable managed subscriptions.

**MQCA\_MODEL\_NON\_DURABLE\_Q**

Name of the model queue for non-durable managed subscriptions.

**MQCA\_TOPIC\_DESC**

Description of the topic object.

**MQCA\_TOPIC\_NAME**

Name of the topic object.

**MQCA\_TOPIC\_STRING**

The topic string for the topic object.

**MQIA\_CLUSTER\_OBJECT\_STATE**

The current state of the clustered topic definition.

**MQIA\_CLUSTER\_PUB\_ROUTE**

The routing behavior of publications between queue managers in a cluster.

**MQIA\_DEF\_PRIORITY**

Default message priority.

**MQIA\_DEF\_PUT\_RESPONSE\_TYPE**

Default put response.

**MQIA\_DURABLE\_SUB**

Whether durable subscriptions are permitted.

**MQIA\_INHIBIT\_PUB**

Whether publications are allowed.

**MQIA\_INHIBIT\_SUB**

Whether subscriptions are allowed.

**MQIA\_NPM\_DELIVERY**

The delivery mechanism for non-persistent messages.

**MQIA\_PM\_DELIVERY**

The delivery mechanism for persistent messages.

**MQIA\_PROXY\_SUB**

Whether a proxy subscription is to be sent for this topic, even if no local subscriptions exist.

**MQIA\_PUB\_SCOPE**

Whether this queue manager propagates publications to queue managers as part of a hierarchy or a publish/subscribe cluster.

**MQIA\_SUB\_SCOPE**

Whether this queue manager propagates subscriptions to queue managers as part of a hierarchy or a publish/subscribe cluster.

**MQIA\_TOPIC\_DEF\_PERSISTENCE**

Default message persistence.

**MQIA\_USE\_DEAD\_LETTER\_Q**

Determines whether the dead-letter queue is used when publication messages cannot be delivered to their correct subscriber queue.

**TopicType (MQCFIN)**

Cluster information (parameter identifier: MQIA\_TOPIC\_TYPE).

If this parameter is present, eligible queues are limited to the specified type. Any attribute selector that is specified in the TopicAttrs list and that is valid only for topics of different type is ignored; no error is raised.

If this parameter is not present (or if MQIACF\_ALL is specified), queues of all types are eligible. Each attribute specified must be a valid topic attribute selector (that is, it must be in the following list), but it need not be applicable to all or any of the topics returned. Topic attribute selectors that are valid but not applicable to the queue are ignored; no error messages occur and no attribute is returned.

The value can be any of the following values:

**MQTOPT\_ALL**

All topic types are displayed. MQTOPT\_ALL includes cluster topics, if ClusterInfo is also specified. MQTOPT\_ALL is the default value.

**MQTOPT\_CLUSTER**

Topics that are defined in publish/subscribe clusters are returned.

**MQTOPT\_LOCAL**

Locally defined topics are displayed.

## **MQCMD\_INQUIRE\_TOPIC (Inquire Topic) Response**

The response to the Inquire Topic (MQCMD\_INQUIRE\_TOPIC) PCF command consists of the response header followed by the *TopicName* structure (and on z/OS only, the *QSG Disposition* structure), and the requested combination of attribute parameter structures (where applicable).

### **Always returned:**

*TopicName*, *TopicType*,  *QSGDisposition*

### **Returned if requested:**

*AlterationDate*, *AlterationTime*, *CapExpiry*, *ClusterName*, *ClusterObjectState*, *ClusterPubRoute*, *CommInfo*, *Custom*, *DefPersistence*, *DefPriority*, *DefPutResponse*, *DurableModelQName*, *DurableSubscriptions*, *InhibitPublications*, *InhibitSubscriptions*, *Multicast*, *NonDurableModelQName*, *NonPersistentMsgDelivery*, *PersistentMsgDelivery*, *ProxySubscriptions*, *PublicationScope*, *QMGrName*, *SubscriptionScope*, *TopicDesc*, *TopicString*, *UseDLQ*, *WildcardOperation*

## **Response data**

### **AlterationDate (MQCFST)**

Alteration date (parameter identifier: MQCA\_ALTERATION\_DATE).

The date when the information was last altered, in the form yyyy-mm-dd.

### **AlterationTime (MQCFST)**

Alteration time (parameter identifier: MQCA\_ALTERATION\_TIME).

The time when the information was last altered, in the form hh.mm.ss.

V 9.3.1

Multi

### **CapExpiry (MQCFIN)**

Capped message expiry processing (parameter identifier MQIA\_CAP\_EXPIRY).

Specifies a lifetime limit for messages put using the object, expressed in 10ths of a second.

### **ClusterName (MQCFST)**

The name of the cluster to which this topic belongs. (parameter identifier: MQCA\_CLUSTER\_NAME).

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH. Setting this parameter to a cluster that this queue manager is a member of makes all queue managers in the cluster aware of this topic. Any publication to this topic or a topic string below it put to any queue manager in the cluster is propagated to subscriptions on any other queue manager in the cluster. For more details, see [Distributed publish/subscribe networks](#).

The value can be any of the following values:

#### **Blank**

If no topic object above this topic in the topic tree has set this parameter to a cluster name, then this topic does not belong to a cluster. Publications and subscriptions for this topic are not propagated to publish/subscribe cluster-connected queue managers. If a topic node higher in the topic tree has a cluster name set, publications and subscriptions to this topic are also propagated throughout the cluster.

This value is the default value for this parameter if no value is specified.

#### **String**

The topic belongs to this cluster. It is not recommended that this is set to a different cluster from a topic object above this topic object in the topic tree. Other queue managers in the cluster will honor this object's definition unless a local definition of the same name exists on those queue managers.

Additionally, if **PublicationScope** or **SubscriptionScope** are set to MQSCOPE\_ALL, this value is the cluster to be used for the propagation of publications and subscriptions, for this topic, to publish/subscribe cluster-connected queue managers.



**ClusterObjectState (MQCFIN)**

The current state of the clustered topic definition (parameter identifier: MQIA\_CLUSTER\_OBJECT\_STATE).

The value can be any of the following values:

**MQCLST\_ACTIVE**

The cluster topic is correctly configured and being adhered to by this queue manager.

**MQCLST\_PENDING**

Only seen by a hosting queue manager, this state is reported when the topic has been created but the full repository has not yet propagated it to the cluster. This might be because the host queue manager is not connected to a full repository, or because the full repository has deemed the topic to be invalid.

**MQCLST\_INVALID**

This clustered topic definition conflicts with an earlier definition in the cluster and is therefore not currently active.

**MQCLST\_ERROR**

An error has occurred with respect to this topic object.

This parameter is typically used to aid diagnosis when multiple definitions of the same clustered topic are defined on different queue managers, and the definitions are not identical. See [Routing for publish/subscribe clusters: Notes on behavior](#).

**ClusterPubRoute (MQCFIN)**

The routing behavior of publications between queue managers in a cluster (parameter identifier: MQIA\_CLUSTER\_PUB\_ROUTE).

The value can be any of the following values:

**MQCLROUTE\_DIRECT**

When you configure a direct routed clustered topic on a queue manager, all queue managers in the cluster become aware of all other queue managers in the cluster. When performing publish and subscribe operations, each queue manager can connect direct to any other queue manager in the cluster.

**MQCLROUTE\_TOPIC\_HOST**

When you use topic host routing, all queue managers in the cluster become aware of the cluster queue managers that host the routed topic definition (that is, the queue managers on which you have defined the topic object). When performing publish and subscribe operations, queue managers in the cluster connect only to these topic host queue managers, and not directly to each other. The topic host queue managers are responsible for routing publications from queue managers on which publications are published to queue managers with matching subscriptions.

**CommInfo (MQCFST)**

The name of the communication information object (parameter identifier: MQCA\_COMM\_INFO\_NAME).

Shows the resolved value of the name of the communication information object to be used for this topic node.

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

**Custom (MQCFST)**

Custom attribute for new features (parameter identifier: MQCA\_CUSTOM).

This attribute is reserved for the configuration of new features before separate attributes have been introduced. It can contain the values of zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs have the form NAME (VALUE).

This description will be updated when features using this attribute are introduced.

**DefPersistence (MQCFIN)**

Default persistence (parameter identifier: MQIA\_TOPIC\_DEF\_PERSISTENCE).

The value can be:

**MQPER\_PERSISTENCE\_AS\_PARENT**

The default persistence is based on the setting of the closest parent administrative topic object in the topic tree.

**MQPER\_PERSISTENT**

Message is persistent.

**MQPER\_NOT\_PERSISTENT**

Message is not persistent.

**DefPriority (MQCFIN)**

Default priority (parameter identifier: MQIA\_DEF\_PRIORITY).

**DefPutResponse (MQCFIN)**

Default put response (parameter identifier: MQIA\_DEF\_PUT\_RESPONSE\_TYPE).

The value can be:

**MQPRT\_ASYNC\_RESPONSE**

The put operation is issued asynchronously, returning a subset of MQMD fields.

**MQPRT\_RESPONSE\_AS\_PARENT**

The default put response is based on the setting of the closest parent administrative topic object in the topic tree.

**MQPRT\_SYNC\_RESPONSE**

The put operation is issued synchronously, returning a response.

**DurableModelQName (MQCFST)**

Name of the model queue to be used for durable managed subscriptions (parameter identifier: MQCA\_MODEL\_DURABLE\_Q).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**DurableSubscriptions (MQCFIN)**

Whether applications are permitted to make durable subscriptions (parameter identifier: MQIA\_DURABLE\_SUB).

The value can be:

**MQSUB\_DURABLE\_AS\_PARENT**

Whether durable subscriptions are permitted is based on the setting of the closest parent administrative topic object in the topic tree.

**MQSUB\_DURABLE\_ALLOWED**

Durable subscriptions are permitted.

**MQSUB\_DURABLE\_INHIBITED**

Durable subscriptions are not permitted.

**InhibitPublications (MQCFIN)**

Whether publications are allowed for this topic (parameter identifier: MQIA\_INHIBIT\_PUB).

The value can be:

**MQTA\_PUB\_AS\_PARENT**

Whether messages can be published to this topic is based on the setting of the closest parent administrative topic object in the topic tree.

**MQTA\_PUB\_INHIBITED**

Publications are inhibited for this topic.

**MQTA\_PUB\_ALLOWED**

Publications are allowed for this topic.

**InhibitSubscriptions (MQCFIN)**

Whether subscriptions are allowed for this topic (parameter identifier: MQIA\_INHIBIT\_SUB).

The value can be:

**MQTA\_SUB\_AS\_PARENT**

Whether applications can subscribe to this topic is based on the setting of the closest parent administrative topic object in the topic tree.

**MQTA\_SUB\_INHIBITED**

Subscriptions are inhibited for this topic.

**MQTA\_SUB\_ALLOWED**

Subscriptions are allowed for this topic.

**Multicast (MQCFIN)**

Whether multicast is used for this topic (parameter identifier: MQIA\_MULTICAST).

Returned value:

**MQMC\_ENABLED**

Multicast can be used.

**MQMC\_DISABLED**

Multicast is not used.

**MQMC\_ONLY**

Only Multicast publish/subscribe can be used on this topic.

**NonDurableModelQName (MQCFST)**

Name of the model queue to be used for non-durable managed subscriptions (parameter identifier: MQCA\_MODEL\_NON\_DURABLE\_Q).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**NonPersistentMsgDelivery (MQCFIN)**

The delivery mechanism for non-persistent messages published to this topic (parameter identifier: MQIA\_NPM\_DELIVERY).

The value can be:

**MQDLV\_AS\_PARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**MQDLV\_ALL**

Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT fails.

**MQDLV\_ALL\_DUR**

Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a non-persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no other subscribers receive the message and the MQPUT fails.

**MQDLV\_ALL\_AVAIL**

Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**PersistentMsgDelivery (MQCFIN)**

The delivery mechanism for persistent messages published to this topic (parameter identifier: MQIA\_PM\_DELIVERY).

The value can be:

**MQDLV\_AS\_PARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**MQDLV\_ALL**

Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT fails.

**MQDLV\_ALL\_DUR**

Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no other subscribers receive the message and the MQPUT fails.

**MQDLV\_ALL\_AVAIL**

Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**ProxySubscriptions (MQCFIN)**

Whether a proxy subscription is to be sent for this topic, even if no local subscriptions exist, to directly connected queue managers (parameter identifier: MQIA\_PROXY\_SUB).

The value can be:

**MQTA\_PROXY\_SUB\_FORCE**

A proxy subscription is sent to connected queue managers even if no local subscriptions exist.

**MQTA\_PROXY\_SUB\_FIRSTUSE**

A proxy subscription is sent for this topic only when a local subscription exists.

**PublicationScope (MQCFIN)**

Whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster (parameter identifier: MQIA\_PUB\_SCOPE).

The value can be:

**MQSCOPE\_ALL**

Publications for this topic are propagated to hierarchically connected queue managers and to publish/subscribe cluster-connected queue managers.

**MQSCOPE\_AS\_PARENT**

Whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

MQSCOPE\_AS\_PARENT is the default value for this parameter if no value is specified.

**MQSCOPE\_QMGR**

Publications for this topic are not propagated to other queue managers.

**Note:** You can override this behavior on a publication-by-publication basis, using MQPMO\_SCOPE\_QMGR on the Put Message Options.

**QMgrName (MQCFST)**

Name of local queue manager (parameter identifier: MQCA\_CLUSTER\_Q\_MGR\_NAME).

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH

**SubscriptionScope (MQCFIN)**

Whether this queue manager propagates subscriptions to queue managers as part of a hierarchy or as part of a publish/subscribe cluster (parameter identifier: MQIA\_SUB\_SCOPE).

The value can be:

**MQSCOPE\_ALL**

Subscriptions for this topic are propagated to hierarchically connected queue managers and to publish/subscribe cluster-connected queue managers.

**MQSCOPE\_AS\_PARENT**

Whether this queue manager propagates subscriptions to queue managers as part of a hierarchy or as part of a publish/subscribe cluster is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

MQSCOPE\_AS\_PARENT is the default value for this parameter if no value is specified.

**MQSCOPE\_QMGR**

Subscriptions for this topic are not propagated to other queue managers.

**Note:** You can override this behavior on a subscription-by-subscription basis, using MQSO\_SCOPE\_QMGR on the Subscription Descriptor or SUBSCOPE(QMGR) on DEFINE SUB.

**TopicDesc (MQCFST)**

Topic description (parameter identifier: MQCA\_TOPIC\_DESC).

The maximum length is MQ\_TOPIC\_DESC\_LENGTH.

**TopicName (MQCFST)**

Topic object name (parameter identifier: MQCA\_TOPIC\_NAME).

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

**TopicString (MQCFST)**

The topic string (parameter identifier: MQCA\_TOPIC\_STRING).

The '/' character within this string has special meaning. It delimits the elements in the topic tree. A topic string can start with the '/' character but is not required to. A string starting with the '/' character is not the same as the string which starts without the '/' character. A topic string cannot end with the '/' character.

The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

**TopicType (MQCFIN)**

Whether this object is a local or cluster topic (parameter identifier: MQIA\_TOPIC\_TYPE).

The value can be:

**MQTOPT\_LOCAL**

This object is a local topic.

**MQTOPT\_CLUSTER**

This object is a cluster topic.

**UseDLQ (MQCFIN)**

Whether the dead-letter queue (or undelivered message queue) should be used when publication messages cannot be delivered to their correct subscriber queue (parameter identifier: MQIA\_USE\_DEAD\_LETTER\_Q).

The value might be:

**MQUSEDLQ\_NO**

Publication messages that cannot be delivered to their correct subscriber queue are treated as a failure to put the message and the application's MQPUT to a topic will fail in accordance with the settings of NPMSGDLV and PMSGDLV.

**MQUSEDLQ\_YES**

If the queue manager DEADQ attribute provides the name of a dead-letter queue then it will be used, otherwise the behaviour will be as for MQUSEDLQ\_NO.

**MQUSEDLQ\_AS\_PARENT**

Whether to use the dead-letter queue is based on the setting of the closest administrative topic object in the topic tree.

**WildcardOperation (MQCFIN)**

Behavior of subscriptions including wildcards made to this topic (parameter identifier: MQIA\_WILDCARD\_OPERATION).

The value can be:

## **MQTA\_PASSTHRU**

Subscriptions made using wildcard topic names that are less specific than the topic string at this topic object receive publications made to this topic and to topic strings more specific than this topic. MQTA\_PASSTHRU is the default supplied with IBM MQ.

## **MQTA\_BLOCK**

Subscriptions made using wildcard topic names that are less specific than the topic string at this topic object do not receive publications made to this topic or to topic strings more specific than this topic.

## **MQCMD\_INQUIRE\_TOPIC\_NAMES (Inquire Topic Names)**

The Inquire Topic Names (MQCMD\_INQUIRE\_TOPIC\_NAMES) PCF command inquires a list of administrative topic names that match the generic topic name specified.

## **Required parameters**

### **TopicName (MQCFST)**

Administrative topic object name (parameter identifier: MQCA\_TOPIC\_NAME).

Specifies the name of the administrative topic object that information is to be returned for.

Generic topic object names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

## **Optional parameters**



### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_LIVE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_LIVE is the default value if the parameter is not specified.

#### **MQQSGD\_ALL**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD\_GROUP.

If MQQSGD\_LIVE is specified or defaulted, or if MQQSGD\_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP. MQQSGD\_GROUP is permitted only in a shared queue environment.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

#### **MQQSGD\_PRIVATE**

The object is defined as MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE returns the same information as MQQSGD\_LIVE.

### **MQCMD\_INQUIRE\_TOPIC\_NAMES (Inquire Topic Names) Response**

The response to the Inquire Topic Names (MQCMD\_INQUIRE\_TOPIC\_NAMES) PCF command consists of the response header followed by a parameter structure giving zero or more names that match the specified administrative topic name.

**z/OS** Additionally, on z/OS only, the **QSGDispositions** parameter structure (with the same number of entries as the *TopicNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *TopicNames* structure.

#### **Always returned:**

*TopicNames*, **z/OS** *QSGDispositions*

#### **Returned if requested:**

None

### **Response data**

#### **TopicNames (MQCFSL)**

List of topic object names (parameter identifier: MQCACF\_TOPIC\_NAMES).

#### **z/OS QSGDispositions (MQCFIL)**

List of queue sharing group dispositions (parameter identifier: MQIACF\_QSG\_DISPS). This parameter is valid on z/OS only. The value can be:

#### **MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

#### **MQQSGD\_GROUP**

The object is defined as MQQSGD\_GROUP.

#### **MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

### **MQCMD\_INQUIRE\_TOPIC\_STATUS (Inquire Topic Status)**

The Inquire Topic Status (MQCMD\_INQUIRE\_TOPIC\_STATUS) PCF command inquires the status of a particular topic, or of a topic and its child topics. The Inquire Topic Status command has a required parameter. The Inquire Topic Status command has optional parameters.

### **Required parameters**

#### **TopicString (MQCFST)**

The topic string (parameter identifier: MQCA\_TOPIC\_STRING).

The name of the topic string to display. IBM MQ uses the topic wildcard characters ('#' and '+') and does not treat a trailing asterisk as a wildcard. For more information about using wildcard characters, refer to the related topic.

The maximum length of the string is MQ\_TOPIC\_STR\_LENGTH.

## Optional parameters

z/OS

### CommandScope (MQCFST)

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- Blank (or omit the parameter altogether). The command runs on the queue manager on which you enter it.
- A queue manager name. The command runs on the queue manager that you specify, if it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which you entered the command, you must be using a queue sharing group environment, and the command server must be enabled.
- An asterisk (\*). The command runs on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

You cannot use CommandScope as a filter parameter.

### IntegerFilterCommand (MQCFIF)

Integer filter command descriptor that you use to restrict the output from the command. The parameter identifier must be an integer type and must be one of the values allowed for *MQIACF\_TOPIC\_SUB\_STATUS*, *MQIACF\_TOPIC\_PUB\_STATUS* or *MQIACF\_TOPIC\_STATUS*, except *MQIACF\_ALL*.

If you specify an integer filter, you cannot also specify a string filter with the **StringFilterCommand** parameter.

### StatusType (MQCFIN)

The type of status to return (parameter identifier: MQIACF\_TOPIC\_STATUS\_TYPE).

The value can be:

**MQIACF\_TOPIC\_STATUS**  
**MQIACF\_TOPIC\_SUB**  
**MQIACF\_TOPIC\_PUB**

This command ignores any attribute selectors specified in the *TopicStatusAttrs* list that are not valid for the selected *StatusType* and the command raises no error.

The default value if this parameter is not specified is **MQIACF\_TOPIC\_STATUS**.

### StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter allowed for *MQIACF\_TOPIC\_SUB\_STATUS*, *MQIACF\_TOPIC\_PUB\_STATUS* or *MQIACF\_TOPIC\_STATUS*, except *MQIACF\_ALL*, or the identifier *MQCA\_TOPIC\_STRING\_FILTER* to filter on the topic string.

Use the parameter identifier to restrict the output from the command by specifying a filter condition. Ensure that the parameter is valid for the type selected in *StatusType*. If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

### TopicStatusAttrs (MQCFIL)

Topic status attributes (parameter identifier: MQIACF\_TOPIC\_STATUS\_ATTRS)

The default value used if the parameter is not specified is:



## MQIACF\_ALL

You can specify any of the parameter values listed in “MQCMD\_INQUIRE\_TOPIC\_STATUS (Inquire Topic Status) Response” on page 1441. It is not an error to request status information that is not relevant for a particular status type, but the response contains no information for the value concerned.

### **MQCMD\_INQUIRE\_TOPIC\_STATUS (Inquire Topic Status) Response**

The response of the Inquire topic (MQCMD\_INQUIRE\_TOPIC\_STATUS) PCF command consists of the response header, followed by the *TopicString* structure, and the requested combination of attribute parameter structures (where applicable). The Inquire Topic Status command returns the values requested when the *StatusType* is MQIACF\_TOPIC\_STATUS. The Inquire Topic Status command returns the values requested when the *StatusType* is MQIACF\_TOPIC\_STATUS\_SUB. The Inquire Topic Status command returns the values requested when the *StatusType* is MQIACF\_TOPIC\_STATUS\_PUB.

#### **Always returned:**

*TopicString*

#### **Returned if requested and StatusType is MQIACF\_TOPIC\_STATUS:**

*CapExpiry, Cluster, ClusterPubRoute, CommInfo, DefPriority, DefaultPutResponse, DefPersistence, DurableSubscriptions, InhibitPublications, InhibitSubscriptions, AdminTopicName, Multicast, DurableModelQName, NonDurableModelQName, PersistentMessageDelivery, NonPersistentMessageDelivery, RetainedPublication, PublishCount, SubscriptionScope, SubscriptionCount, PublicationScope, UseDLQ*

**Note:** The Inquire Topic Status command returns only resolved values for the topic, and no AS\_PARENT values.

#### **Returned if requested and StatusType is MQIACF\_TOPIC\_SUB:**

*SubscriptionId, SubscriptionUserId, Durable, SubscriptionType, ResumeDate, ResumeTime, LastMessageDate, LastMessageTime, NumberOfMessages, ActiveConnection*

#### **Returned if requested and StatusType is MQIACF\_TOPIC\_PUB:**

*LastPublishDate, LastPublishTime, NumberOfPublishes, ActiveConnection*

### **Response data (TOPIC\_STATUS)**

V9.3.1

Multi

#### **CapExpiry (MQCFIN)**

Capped message expiry processing (parameter identifier MQIA\_CAP\_EXPIRY).

Specifies a lifetime limit for messages put using the object, expressed in 10ths of a second.

#### **ClusterName (MQCFST)**

The name of the cluster to which this topic belongs. (parameter identifier: MQCA\_CLUSTER\_NAME).

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH. Setting this parameter to a cluster that this queue manager is a member of makes all queue managers in the cluster aware of this topic. Any publication to this topic or a topic string below it put to any queue manager in the cluster is propagated to subscriptions on any other queue manager in the cluster. For more details, see [Distributed publish/subscribe networks](#).

The value can be any of the following values:

##### **Blank**

If no topic object above this topic in the topic tree has set this parameter to a cluster name, then this topic does not belong to a cluster. Publications and subscriptions for this topic are not propagated to publish/subscribe cluster-connected queue managers. If a topic node higher in the topic tree has a cluster name set, publications and subscriptions to this topic are also propagated throughout the cluster.

This value is the default value for this parameter if no value is specified.

##### **String**

The topic belongs to this cluster. It is not recommended that this is set to a different cluster from a topic object above this topic object in the topic tree. Other queue managers in the cluster will

honor this object's definition unless a local definition of the same name exists on those queue managers.

Additionally, if **PublicationScope** or **SubscriptionScope** are set to MQSCOPE\_ALL, this value is the cluster to be used for the propagation of publications and subscriptions, for this topic, to publish/subscribe cluster-connected queue managers.

### **ClusterPubRoute (MQCFIN)**

The routing behavior to use for this topic in the cluster (parameter identifier: MQIA\_CLUSTER\_PUB\_ROUTE).

The values can be as follows:

#### **MQCLROUTE\_DIRECT**

A publication on this topic string, originating from this queue manager, is sent direct to any queue manager in the cluster with a matching subscription.

#### **MQCLROUTE\_TOPIC\_HOST**

A publication on this topic string, originating from this queue manager, is sent to one of the queue managers in the cluster that hosts a definition of the corresponding clustered topic object, and from there to any queue manager in the cluster with a matching subscription.

#### **MQCLROUTE\_NONE**

This topic node is not clustered.

### **CommInfo (MQCFST)**

The name of the communication information object (parameter identifier: MQCA\_COMM\_INFO\_NAME).

Shows the resolved value of the name of the communication information object to be used for this topic node.

The maximum length of the string is MQ\_COMM\_INFO\_NAME\_LENGTH.

### **DefPersistence (MQCFIN)**

Default persistence (parameter identifier: MQIA\_TOPIC\_DEF\_PERSISTENCE).

Returned value:

#### **MQPER\_PERSISTENT**

Message is persistent.

#### **MQPER\_NOT\_PERSISTENT**

Message is not persistent.

### **DefaultPutResponse (MQCFIN)**

Default put response (parameter identifier: MQIA\_DEF\_PUT\_RESPONSE\_TYPE).

Returned value:

#### **MQPRT\_SYNC\_RESPONSE**

The put operation is issued synchronously, returning a response.

#### **MQPRT\_ASYNC\_RESPONSE**

The put operation is issued asynchronously, returning a subset of MQMD fields.

### **DefPriority (MQCFIN)**

Default priority (parameter identifier: MQIA\_DEF\_PRIORITY).

Shows the resolved default priority of messages published to the topic.

### **DurableSubscriptions (MQCFIN)**

Whether applications are permitted to make durable subscriptions (parameter identifier: MQIA\_DURABLE\_SUB).

Returned value:

#### **MQSUB\_DURABLE\_ALLOWED**

Durable subscriptions are permitted.

**MQSUB\_DURABLE\_INHIBITED**

Durable subscriptions are not permitted.

**InhibitPublications (MQCFIN)**

Whether publications are allowed for this topic (parameter identifier: MQIA\_INHIBIT\_PUB).

Returned value:

**MQTA\_PUB\_INHIBITED**

Publications are inhibited for this topic.

**MQTA\_PUB\_ALLOWED**

Publications are allowed for this topic.

**InhibitSubscriptions (MQCFIN)**

Whether subscriptions are allowed for this topic (parameter identifier: MQIA\_INHIBIT\_SUB).

Returned value:

**MQTA\_SUB\_INHIBITED**

Subscriptions are inhibited for this topic.

**MQTA\_SUB\_ALLOWED**

Subscriptions are allowed for this topic.

**AdminTopicName (MQCFST)**

Topic object name (parameter identifier: MQCA\_ADMIN\_TOPIC\_NAME).

If the topic is an admin-node, the command displays the associated topic object name containing the node configuration. If the field is not an admin-node the command displays a blank.

The maximum length of the string is MQ\_TOPIC\_NAME\_LENGTH.

**Multicast (MQCFIN)**

Whether multicast is used for this topic (parameter identifier: MQIA\_MULTICAST).

Returned value:

**MQMC\_ENABLED**

Multicast can be used.

**MQMC\_DISABLED**

Multicast is not used.

**MQMC\_ONLY**

Only Multicast publish/subscribe can be used on this topic.

**DurableModelQName (MQCFST)**

The name of the model queue used for managed durable subscriptions (parameter identifier: MQCA\_MODEL\_DURABLE\_Q).

Shows the resolved value of the name of the model queue to be used for durable subscriptions that request the queue manager to manage the destination of publications.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**NonDurableModelQName (MQCFST)**

The name of the model queue for managed non-durable subscriptions (parameter identifier: MQCA\_MODEL\_NON\_DURABLE\_Q).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**PersistentMessageDelivery (MQCFIN)**

Delivery mechanism for persistent messages published to this topic (parameter identifier: MQIA\_PM\_DELIVERY).

Returned value:

**MQDLV\_ALL**

Persistent messages must be delivered to all subscribers, irrespective of durability, for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**MQDLV\_ALL\_DUR**

Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

**MQDLV\_ALL\_AVAIL**

Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**NonPersistentMessageDelivery (MQCFIN)**

Delivery mechanism for non-persistent messages published to this topic (parameter identifier: MQIA\_NPM\_DELIVERY).

Returned value:

**MQDLV\_ALL**

Non-persistent messages must be delivered to all subscribers, irrespective of durability, for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**MQDLV\_ALL\_DUR**

Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a non-persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

**MQDLV\_ALL\_AVAIL**

Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**RetainedPublication (MQCFIN)**

Whether there is a retained publication for this topic (parameter identifier: MQIACF\_RETAINED\_PUBLICATION).

Returned value:

**MQQSO\_YES**

There is a retained publication for this topic.

**MQQSO\_NO**

There is no retained publication for this topic.

**PublishCount (MQCFIN)**

Publish count (parameter identifier: MQIA\_PUB\_COUNT).

The number of applications currently publishing to the topic.

**SubscriptionCount (MQCFIN)**

Subscription count (parameter identifier: MQIA\_SUB\_COUNT).

The number of subscribers for this topic string, including durable subscribers who are not currently connected.

**SubscriptionScope (MQCFIN)**

Determines whether this queue manager propagates subscriptions for this topic to queue managers as part of a hierarchy or as part of a publish/subscribe cluster (parameter identifier: MQIA\_SUB\_SCOPE).

Returned value:

**MQSCOPE\_QMGR**

The queue manager does not propagate subscriptions for this topic to other queue managers.

**MQSCOPE\_ALL**

The queue manager propagates subscriptions for this topic to hierarchically connected queue managers and to publish/subscribe cluster connected queues.

**PublicationScope (MQCFIN)**

Determines whether this queue manager propagates publications for this topic to queue managers as part of a hierarchy or as part of a publish/subscribe cluster (parameter identifier: MQIA\_PUB\_SCOPE).

Returned value:

**MQSCOPE\_QMGR**

The queue manager does not propagate publications for this topic to other queue managers.

**MQSCOPE\_ALL**

The queue manager propagates publications for this topic to hierarchically connected queue managers and to publish/subscribe cluster connected queues.

**UseDLQ (MQCFIN)**

Determines whether the dead-letter queue is used when publication messages cannot be delivered to their correct subscriber queue (parameter identifier: MQIA\_USE\_DEAD\_LETTER\_Q).

The value can be any of the following values:

**MQUSEDLQ\_NO**

Publication messages that cannot be delivered to their correct subscriber queue are treated as a failure to put the message. The MQPUT of an application to a topic fails in accordance with the settings of MQIA\_NPM\_DELIVERY and MQIA\_PM\_DELIVERY.

**MQUSEDLQ\_YES**

If the DEADQ queue manager attribute provides the name of a dead-letter queue then it is used, otherwise the behavior is as for MQUSEDLQ\_NO.

**Response data (TOPIC\_STATUS\_SUB)****SubscriptionId (MQCFBS)**

Subscription identifier (parameter identifier: MQBACF\_SUB\_ID).

The queue manager assigns *SubscriptionId* as an all time unique identifier for this subscription.

The maximum length of the string is MQ\_CORREL\_ID\_LENGTH.

**SubscriptionUserId (MQCFST)**

The user ID that owns this subscription (parameter identifier: MQCACF\_SUB\_USER\_ID).

The maximum length of the string is MQ\_USER\_ID\_LENGTH.

**Durable (MQCFIN)**

Whether this subscription is a durable subscription (parameter identifier: MQIACF\_DURABLE\_SUBSCRIPTION).

**MQSUB\_DURABLE\_YES**

The subscription persists, even if the creating application disconnects from the queue manager or issues an MQCLOSE call for the subscription. The queue manager reinstates the subscription during restart.

**MQSUB\_DURABLE\_NO**

The subscription is non-durable. The queue manager removes the subscription when the creating application disconnects from the queue manager or issues an MQCLOSE call for the subscription. If the subscription has a destination class (DESTCLAS) of MANAGED, the queue manager removes any messages not yet consumed when it closes the subscription.

**SubscriptionType (MQCFIN)**

The type of subscription (parameter identifier: MQIACF\_SUB\_TYPE).

The value can be:

MQSUBTYPE\_ADMIN

MQSUBTYPE\_API

MQSUBTYPE\_PROXY

**ResumeDate (MQCFST)**

Date of the most recent MQSUB call that connected to this subscription (parameter identifier: MQCA\_RESUME\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

**ResumeTime (MQCFST)**

Time of the most recent MQSUB call that connected to this subscription (parameter identifier: MQCA\_RESUME\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**LastMessageDate (MQCFST)**

Date on which an MQPUT call last sent a message to this subscription. The queue manager updates the date field after the MQPUT call successfully puts a message to the destination specified by this subscription (parameter identifier: MQCACF\_LAST\_MSG\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

**Note:** An MQSUBRQ call updates this value.

**LastMessageTime (MQCFST)**

Time at which an MQPUT call last sent a message to this subscription. The queue manager updates the time field after the MQPUT call successfully puts a message to the destination specified by this subscription (parameter identifier: MQCACF\_LAST\_MSG\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**Note:** An MQSUBRQ call updates this value.

**NumberOfMessages (MQCFIN)**

Number of messages put to the destination specified by this subscription (parameter identifier: MQIACF\_MESSAGE\_COUNT).

**Note:** An MQSUBRQ call updates this value.

**ActiveConnection (MQCFBS)**

The currently active *ConnectionId* (CONNID) that opened this subscription (parameter identifier: MQBACF\_CONNECTION\_ID).

The maximum length of the string is MQ\_CONNECTION\_ID\_LENGTH.

**Response data (TOPIC\_STATUS\_PUB)**

**LastPublicationDate (MQCFST)**

Date on which this publisher last sent a message (parameter identifier: MQCACF\_LAST\_PUB\_DATE).

The maximum length of the string is MQ\_DATE\_LENGTH.

**LastPublicationTime (MQCFST)**

Time at which this publisher last sent a message (parameter identifier: MQCACF\_LAST\_PUB\_TIME).

The maximum length of the string is MQ\_TIME\_LENGTH.

**NumberOfPublishes (MQCFIN)**

Number of publishes made by this publisher (parameter identifier: MQIACF\_PUBLISH\_COUNT).

**ActiveConnection (MQCFBS)**

The currently active *ConnectionId* (CONNID) associated with the handle that has this topic open for publish (parameter identifier: MQBACF\_CONNECTION\_ID).

The maximum length of the string is MQ\_CONNECTION\_ID\_LENGTH.

## **MQCMD\_INQUIRE\_USAGE (Inquire Usage) on z/OS**

The Inquire Usage (MQCMD\_INQUIRE\_USAGE) PCF command inquires about the current state of a page set, or information about the log data sets.

### Optional parameters

#### CommandScope (MQCFST)

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

#### PageSetId (MQCFIN)

Page set identifier (parameter identifier: MQIA\_PAGESET\_ID). If you omit this parameter, all page set identifiers are returned.

#### UsageType (MQCFIN)

The type of information to be returned (parameter identifier: MQIACF\_USAGE\_TYPE).

The value can be any of the following values:

##### **MQIACF\_USAGE\_PAGESET**

Return page set (MQIACF\_USAGE\_PAGESET) and buffer pool information (MQIACF\_USAGE\_BUFFER\_POOL).

##### **MQIACF\_USAGE\_DATA\_SET**

Return data set information for log data sets (MQIACF\_USAGE\_DATA\_SET).

##### **MQIACF\_ALL**

Return page set, buffer pool, and data set information (MQIACF\_USAGE\_PAGESET), (MQIACF\_USAGE\_BUFFER\_POOL), and (MQIACF\_USAGE\_DATA\_SET).

##### **MQIACF\_USAGE\_SMDS**

Return shared message data set usage (MQIACF\_USAGE\_SMDS) and buffer pool information (MQIACF\_USAGE\_BUFFER\_POOL).

This includes the allocated, and used space for each data set, and information about the number of buffers currently active, the number with valid contents, and the number of free buffers.

## **MQCMD\_INQUIRE\_USAGE (Inquire Usage) Response on z/OS**

The response to the Inquire Usage (MQCMD\_INQUIRE\_USAGE) PCF command consists of the response header followed by one or more *UsageType* structure and a set of attribute parameter structures determined by the value of *UsageType* in the Inquire command.

### Always returned:

*UsageType*

Possible values of *ParameterType* are:

#### **MQIACF\_USAGE\_PAGESET**

Page set information.

#### **MQIACF\_USAGE\_BUFFER\_POOL**

Buffer pool information.

**MQIACF\_USAGE\_DATA\_SET**

Data set information for log data sets.

**MQIACF\_USAGE\_SMDS**

Return shared message data set usage and buffer pool information.

This includes the allocated, and used space for each data set, and information about the number of buffers currently active, the number with valid contents, and the number of free buffers.

**Returned if *UsageType* is MQIACF\_USAGE\_PAGESET:**

*BufferPoolId, Encrypted, ExpandCount, ExpandType, LogRBA, NonPersistentDataPages, PageSetId, PageSetStatus, PersistentDataPages, TotalPages, UnusedPages*

**Returned if *UsageType* is MQIACF\_USAGE\_BUFFER\_POOL:**

*BufferPoolId, FreeBuffers, FreeBuffersPercentage, TotalBuffers, BufferPoolLocation, PageClass*

**Returned if *UsageType* is MQIACF\_USAGE\_DATA\_SET:**

*DataSetName, DataSetType, LogRBA, LogLRSN*

**Returned if *UsageType* is MQIACF\_USAGE\_SMDS:**

*DataSetName, DataSetType, Encrypted*

**Response data if *UsageType* is MQIACF\_USAGE\_PAGESET****BufferPoolId (MQCFIN)**

Buffer pool identifier (parameter identifier: MQIACF\_BUFFER\_POOL\_ID).

This parameter identifies the buffer pool being used by the page set.

**Encrypted (MQCFIN)**

Shows whether the page set is encrypted (parameter identifier: MQIACF\_DS\_ENCRYPTED)

The value can be one of the following values:

**MQSYSP\_YES**

The page set is encrypted.

**MQSYSP\_NO**

The page set is not encrypted.

**ExpandCount (MQCFIN)**

The number of times the page set has been dynamically expanded since restart (parameter identifier: MQIACF\_USAGE\_EXPAND\_COUNT).

**ExpandType (MQCFIN)**

How the queue manager expands a page set when it becomes nearly full, and further pages are required within it (parameter identifier: MQIACF\_USAGE\_EXPAND\_TYPE).

The value can be:

**MQUSAGE\_EXPAND\_NONE**

No further page set expansion is to take place.

**MQUSAGE\_EXPAND\_USER**

The secondary extent size that was specified when the page set was defined is used. If no secondary extent size was specified, or it was specified as zero, then no dynamic page set expansion can take place.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set. Only one extent is required to reach this size.

**MQUSAGE\_EXPAND\_SYSTEM**

A secondary extent size that is approximately 10 per cent of the current size of the page set is used. MQUSAGE\_EXPAND\_SYSTEM can be rounded up to the nearest cylinder of DASD.



**NonPersistentDataPages (MQCFIN)**

The number of pages holding nonpersistent data (parameter identifier: MQIACF\_USAGE\_NONPERSIST\_PAGES).

These pages are being used to store nonpersistent message data.

**PageSetId (MQCFIN)**

Page set identifier (parameter identifier: MQIA\_PAGESET\_ID).

The string consists of two numeric characters, in the range 00 through 99.

**PageSetStatus (MQCFIN)**

Current status of the page set (parameter identifier: MQIACF\_PAGESET\_STATUS).

The value can be any of the following values:

**MQUSAGE\_PS\_AVAILABLE**

The page set is available.

**MQUSAGE\_PS\_DEFINED**

The page set has been defined but has never been used.

**MQUSAGE\_PS\_OFFLINE**

The page set is currently not accessible by the queue manager, for example because the page set has not been defined to the queue manager.

**MQUSAGE\_PS\_NOT\_DEFINED**

The command was issued for a specific page set that is not defined to the queue manager.

**MQUSAGE\_PS\_SUSPENDED**

The page set has been suspended.

**PersistentDataPages (MQCFIN)**

The number of pages holding persistent data (parameter identifier: MQIACF\_USAGE\_PERSIST\_PAGES).

These pages are being used to store object definitions and persistent message data.

**TotalPages (MQCFIN)**

The total number of 4 KB pages in the page set (parameter identifier: MQIACF\_USAGE\_TOTAL\_PAGES).

**UnusedPages (MQCFIN)**

The number of pages that are not used (that is, available page sets) (parameter identifier: MQIACF\_USAGE\_UNUSED\_PAGES).

**LogRBA (MQCFST)**

Log RBA (parameter identifier: MQCACF\_USAGE\_LOG\_RBA).

The maximum length is MQ\_RBA\_LENGTH.

This response is returned only if PageSetStatus is set to MQUSAGE\_PS\_NOT\_DEFINED or MQUSAGE\_SUSPENDED. However, the response is not always returned if PageSetStatus is set to MQUSAGE\_PS\_NOT\_DEFINED.

A value of 'FFFFFFFFFFFFFFFF' indicates that the page set has never been online.

**Response data if UsageType is MQIACF\_USAGE\_BUFFER\_POOL****BufferPoolId (MQCFIN)**

Buffer pool identifier (parameter identifier: MQIACF\_BUFFER\_POOL\_ID).

This parameter identifies the buffer pool being used by the page set.

**FreeBuffers (MQCFIN)**

Number of free buffers (parameter identifier: MQIACF\_USAGE\_FREE\_BUFF).

**FreeBuffersPercentage (MQCFIN)**

Number of free buffers as a percentage of all buffers in the buffer pool (parameter identifier: MQIACF\_USAGE\_FREE\_BUFF\_PERC).

**TotalBuffers (MQCFIN)**

The number of buffers defined for specified buffer pool (parameter identifier: MQIACF\_USAGE\_TOTAL\_BUFFERS).

**BufferPoolLocation (MQCFIN)**

The location of the buffers in this buffer pool relative to the bar. This is one of the following values:

**MQBPLOCATION\_ABOVE**

All buffer pool buffers are above the bar.

**MQBPLOCATION\_BELOW**

All buffer pool buffers are below the bar.

**MQBPLOCATION\_SWITCHING\_ABOVE**

Buffer pool buffers are being moved above the bar.

**MQBPLOCATION\_SWITCHING\_BELOW**

Buffer pool buffers are being moved below the bar.

**PageClass (MQCFIN)**

The type of virtual storage pages used for backing the buffers in the buffer pool. This is one of the following values:

**MQPAGECLAS\_4KB**

Pageable 4 KB pages are used.

**MQPAGECLAS\_FIXED4KB**

Fixed 4 KB pages are used.

**Response data if UsageType is MQIACF\_USAGE\_DATA\_SET****DataSetName (MQCFST)**

Data set name (parameter identifier: MQCACF\_DATA\_SET\_NAME).

The maximum length is MQ\_DATA\_SET\_NAME\_LENGTH.

**DataSetType (MQCFIN)**

The type of data set, and circumstance (parameter identifier: MQIACF\_USAGE\_DATA\_SET\_TYPE).

The value can be:

**MQUSAGE\_DS\_OLDEST\_ACTIVE\_UOW**

The log data set containing the start RBA of the oldest active unit of work for the queue manager

**MQUSAGE\_DS\_OLDEST\_PS\_RECOVERY**

The log data set containing the oldest restart RBA of any page set for the queue manager.

**MQUSAGE\_DS\_OLDEST\_CF\_RECOVERY**

The log data set containing the LRSN which matches the time of the oldest current backup of any CF structure in the queue sharing group.

**LogRBA (MQCFST)**

Log RBA (parameter identifier: MQCACF\_USAGE\_LOG\_RBA).

The maximum length is MQ\_RBA\_LENGTH.

**LogLRSN (MQCFST)**

Log LRSN (parameter identifier: MQIACF\_USAGE\_LOG\_LRSN).

The length of the string is MQ\_LRSN\_LENGTH.

**Response data if UsageType is MQIACF\_USAGE\_SMDS****Encrypted (MQCFIN)**

Shows whether the SMDS is encrypted (parameter identifier: MQIACF\_DS\_ENCRYPTED)

The value can be one of the following values:

**MQSYSP\_YES**

The SMDS is encrypted.

**MQSYSP\_NO**

The SMDS is not encrypted.

**SMDSStatus (MQCFIN)**

SMDS status (parameter identifier: MQIACF\_SMDS\_STATUS).

**MQUSAGE\_SMDS\_NO\_DATA**

There is no SMDS data available. Nothing further is returned.

**MQUSAGE\_SMDS\_AVAILABLE**

For each CF structure two sets of PCF data are returned:

**A**

**CFStrucNames (MQCFSL)**

List of CF application structure names (parameter identifier: MQCACF\_CF\_STRUC\_NAME).

**MQIACF\_USAGE\_OFFLOAD\_MSGS (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_OFFLOAD\_MSGS).

**MQIACF\_USAGE\_TOTAL\_BLOCKS (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_TOTAL\_BLOCKS).

**MQIACF\_USAGE\_DATA\_BLOCKS (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_DATA\_BLOCKS).

**MQIACF\_USAGE\_USED\_BLOCKS (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_USED\_BLOCKS).

**MQIACF\_USAGE\_USED\_RATE (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_USED\_RATE).

**MQIACF\_SMDS\_STATUS (MQCFIN)**

Description required (parameter identifier: MQIACF\_SMDS\_STATUS). The value is MQUSAGE\_SMDS\_AVAILABLE.

**MQIACF\_USAGE\_TYPE (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_TYPE).

**B**

**CFStrucNames (MQCFSL)**

List of CF application structure names (parameter identifier: MQCACF\_CF\_STRUC\_NAME).

**MQIACF\_USAGE\_BLOCK\_SIZE (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_BLOCK\_SIZE).

**MQIACF\_USAGE\_TOTAL\_BUFFERS (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_TOTAL\_BUFFERS).

**MQIACF\_USAGE\_INUSE\_BUFFERS (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_INUSE\_BUFFERS).

**MQIACF\_USAGE\_SAVED\_BUFFERS (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_SAVED\_BUFFERS).

**MQIACF\_USAGE\_EMPTY\_BUFFERS (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_EMPTY\_BUFFERS).

**MQIACF\_USAGE\_READS\_SAVED (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_READS\_SAVED).

**MQIACF\_USAGE\_LOWEST\_FREE (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_LOWEST\_FREE).

**MQIACF\_USAGE\_WAIT\_RATE (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_WAIT\_RATE).

**MQIACF\_SMDS\_STATUS (MQCFIN)**

Description required (parameter identifier: MQIACF\_SMDS\_STATUS). The value is MQUSAGE\_SMDS\_AVAILABLE.

**MQIACF\_USAGE\_TYPE (MQCFIN)**

Description required (parameter identifier: MQIACF\_USAGE\_TYPE).

## **MQCMD\_MOVE\_Q (Move Queue) on z/OS**

The Move Queue (MQCMD\_MOVE\_Q) PCF command moves all the messages from one local queue to another.

**Required parameters****FromQName (MQCFST)**

From queue name (parameter identifier: MQCACF\_FROM\_Q\_NAME).

The name of the local queue from which messages are moved. The name must be defined to the local queue manager.

The command fails if the queue contains uncommitted messages.

If an application has this queue open, or has open a queue that eventually resolves to this queue, the command fails. For example, the command fails if this queue is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open.

An application can open this queue while the command is in progress but the application waits until the command has completed.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**Optional parameters (Move Queue)****CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

**MoveType (MQCFIN)**

Move type (parameter identifier: MQIA\_QSG\_DISP).

Specifies how the messages are moved. The value can be any of the following values:

**MQIACF\_MOVE\_TYPE\_MOVE**

Move the messages from the source queue to the empty target queue.

The command fails if the target queue already contains one or more messages. The messages are deleted from the source queue. MQIACF\_MOVE\_TYPE\_MOVE is the default value.

**MQIACF\_MOVE\_TYPE\_ADD**

Move the messages from the source queue and add them to any messages already on the target queue.

The messages are deleted from the source queue.

### **QSGDisposition (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be any of the following values:

#### **MQQSGD\_PRIVATE**

The object is defined as either MQQSGD\_Q\_MGR or MQQSGD\_COPY. MQQSGD\_PRIVATE is the default value.

#### **MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED. MQQSGD\_SHARED is valid only in a shared queue environment.

### **ToQName (MQCFST)**

To queue name (parameter identifier: MQCACF\_TO\_Q\_NAME).

The name of the local queue to which messages are moved. The name must be defined to the local queue manager.

The name of the target queue can be the same as the name of the source queue only if the queue exists as both a shared and a private queue. In this case, the command moves messages to the queue that has the opposite disposition (shared or private) from that disposition specified for the source queue on the **QSGDisposition** parameter.

If an application has this queue open, or has open a queue that eventually resolves to this queue, the command fails. The command also fails if this queue is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open.

No application can open this queue while the command is in progress.

If you specify a value of MQIACF\_MOVE\_TYPE\_MOVE on the **MoveType** parameter, the command fails if the target queue already contains one or more messages.

The **DefinitionType**, **HardenGetBackout**, **Usage** parameters of the target queue must be the same as those parameters of the source queue.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

### **MQCMD\_PING\_CHANNEL (Ping Channel)**

The Ping Channel (MQCMD\_PING\_CHANNEL) PCF command tests a channel by sending data as a special message to the remote message queue manager and checking that the data is returned. The data is generated by the local queue manager.

This command can only be used for channels with a *ChannelType* value of MQCHT\_SENDER, MQCHT\_SERVER, or MQCHT\_CLUSSDR.

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

The command is not valid if the channel is running; however it is valid if the channel is stopped or in retry mode.

## **Required parameters**

### **ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The name of the channel to be tested. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

## Optional parameters

### DataCount (MQCFIN)

Data count (parameter identifier: MQIACH\_DATA\_COUNT).

Specifies the length of the data.

Specify a value in the range 16 through 32 768. The default value is 64 bytes.

### z/OS CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### z/OS ChannelDisposition (MQCFIN)

Channel disposition (parameter identifier: MQIACH\_CHANNEL\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be tested.

If this parameter is omitted, then the value for the channel disposition is taken from the default channel disposition attribute of the channel object.

The value can be any of the following values:

#### MQCHLD\_PRIVATE

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD\_SHARED.

#### MQCHLD\_SHARED

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD\_SHARED.

#### MQCHLD\_FIXSHARED

Tests shared channels, tied to a specific queue manager.

The combination of the **ChannelDisposition** and **CommandScope** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in [Table 211 on page 1455](#)

Table 211. ChannelDisposition and CommandScope for PING CHANNEL

<b>ChannelDisposition</b>	<b>CommandScope blank or local-qmgr</b>	<b>CommandScope qmgr-name</b>	<b>CommandScope (*)</b>
MQCHLD_PRIVATE	Ping private channel on the local queue manager	Ping private channel on the named queue manager	Ping private channel on all active queue managers
MQCHLD_SHARED	<p>Ping a shared channel on the most suitable queue manager in the group</p> <p>MQCHLD_SHARED might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted	Not permitted
MQCHLD_FIXSHARED	Ping a shared channel on the local queue manager	Ping a shared channel on the named queue manager	Not permitted

## Error codes

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

### Reason (MQLONG)

The value can be any of the following values:

**MQRCCF\_ALLOCATE\_FAILED**

Allocation failed.

**MQRCCF\_BIND\_FAILED**

Bind failed.

**MQRCCF\_CCSID\_ERROR**

Coded character-set identifier error.

**MQRCCF\_CHANNEL\_CLOSED**

Channel closed.

**MQRCCF\_CHANNEL\_IN\_USE**

Channel in use.

**MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

<b>MQRCCF_CHANNEL_TYPE_ERROR</b>	Channel type not valid.
<b>MQRCCF_CONFIGURATION_ERROR</b>	Configuration error.
<b>MQRCCF_CONNECTION_CLOSED</b>	Connection closed.
<b>MQRCCF_CONNECTION_REFUSED</b>	Connection refused.
<b>MQRCCF_DATA_TOO_LARGE</b>	Data too large.
<b>MQRCCF_ENTRY_ERROR</b>	Connection name not valid.
<b>MQRCCF_HOST_NOT_AVAILABLE</b>	Remote system not available.
<b>MQRCCF_NO_COMMS_MANAGER</b>	Communications manager not available.
<b>MQRCCF_PING_DATA_COMPARE_ERROR</b>	Ping Channel command failed.
<b>MQRCCF_PING_DATA_COUNT_ERROR</b>	Data count not valid.
<b>MQRCCF_PING_ERROR</b>	Ping error.
<b>MQRCCF_RECEIVE_FAILED</b>	Receive failed.
<b>MQRCCF_RECEIVED_DATA_ERROR</b>	Received data error.
<b>MQRCCF_REMOTE_QM_TERMINATING</b>	Remote queue manager terminating.
<b>MQRCCF_REMOTE_QM_UNAVAILABLE</b>	Remote queue manager not available.
<b>MQRCCF_SEND_FAILED</b>	Send failed.
<b>MQRCCF_STRUCTURE_TYPE_ERROR</b>	Structure type not valid.
<b>MQRCCF_TERMINATED_BY_SEC_EXIT</b>	Channel terminated by security exit.
<b>MQRCCF_UNKNOWN_REMOTE_CHANNEL</b>	Remote channel not known.
<b>MQRCCF_USER_EXIT_NOT_AVAILABLE</b>	User exit not available.

**Multi**

### ***MQCMD\_PING\_Q\_MGR (Ping Queue Manager) on Multiplatforms***

The Ping Queue Manager (MQCMD\_PING\_Q\_MGR) PCF command tests whether the queue manager and its command server is responsive to commands. If the queue manager is responding a positive reply is returned.

**Required parameters:**

None

**Optional parameters:**

None



## ALW MQCMD\_PURGE\_CHANNEL (Purge Channel) on AIX, Linux, and Windows

The Purge Channel (MQCMD\_PURGE\_CHANNEL) PCF command stops and purges an IBM MQ telemetry or AMQP channel.

This command can only be issued to channel of type MQTT or AMQP.

Purging a telemetry or AMQP channel disconnects all the MQTT or AMQP clients connect to it, cleans up the state of the MQTT or AMQP clients, and stops the telemetry or AMQP channel. Cleaning the state of a client deletes all the pending publications and removes all the subscriptions from the client.

### Required parameters

#### ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The name of the channel to be stopped and purged. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

### Optional parameters

#### ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH\_CHANNEL\_TYPE).

This parameter is required to purge an MQTT channel. It can not be specified for other types of channels. If specified, this parameter must follow immediately after the **ChannelName** parameter, and the value must be MQCHT\_MQTT.

#### ClientIdentifier (MQCFST)

Client identifier (parameter identifier: MQCACH\_CLIENT\_ID).

The client identifier is a 23 byte string that identifies an MQ Telemetry Transport or AMQP client. When the Purge Channel command specifies a *ClientIdentifier*, only the connection for the specified client identifier is purged. If the *ClientIdentifier* is not specified, all the connections on the channel are purged.

The maximum length of the string is MQ\_CLIENT\_ID\_LENGTH.

## z/OS MQCMD\_RECOVER\_CF\_STRUC (Recover CF Structure) on z/OS

The Recover CF Structure (MQCMD\_RECOVER\_CF\_STRUC) PCF command initiates recovery of CF application structures.

**Note:** This command is valid only on z/OS when the queue manager is a member of a queue sharing group.

### Required parameters

#### CFStrucName (MQCFST)

CF application structure name (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

### Optional parameters

#### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.

- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_Q\_MGR\_NAME\_LENGTH.

### **Purge (MQCFIN)**

Recover to empty CF structure (parameter identifier: MQIACF\_PURGE).

Specifies whether the CF application structure is emptied. The value can be any of the following values:

#### **MQPO\_YES**

Recover to empty CF structure. Any messages in the CF structure are lost.

#### **MQPO\_NO**

Performs a true recovery of the CF structure. MQPO\_NO is the default value.

### **MQCMD\_REFRESH\_CLUSTER (Refresh Cluster)**

The Refresh Cluster (MQCMD\_REFRESH\_CLUSTER) PCF command discards all locally held cluster information, including any auto-defined channels that are not in doubt, and forces the repository to be rebuilt.

**Note:** For large clusters, use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See [Refreshing in a large cluster can affect performance and availability of the cluster.](#)

## **Required parameters**

### **ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

The name of the cluster to be refreshed.

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

This parameter is the name of the cluster to be refreshed. If an asterisk (\*) is specified for the name, the queue manager is refreshed in all the clusters to which it belongs.

If an asterisk (\*) is specified with *RefreshRepository* set to MQCFO\_REFRESH\_REPOSITORY\_YES, the queue manager restarts its search for repository queue managers, using information in the local cluster-sender channel definitions.

## **Optional parameters**

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## RefreshRepository (MQCFIN)

Whether repository information is refreshed (parameter identifier: MQIACF\_REFRESH\_REPOSITORY).

This parameter indicates whether the information about repository queue managers is refreshed.

The value can be:

### MQCFO\_REFRESH\_REPOSITORY\_YES

Refresh repository information.

This value cannot be specified if the queue manager is itself a repository queue manager.

MQCFO\_REFRESH\_REPOSITORY\_YES specifies that in addition to MQCFO\_REFRESH\_REPOSITORY\_NO behavior, objects representing full repository cluster queue managers are also refreshed. Do not use this option if the queue manager is itself a full repository.

If it is a full repository, you must first alter it so that it is not a full repository for the cluster in question.

The full repository location is recovered from the manually defined cluster-sender channel definitions. After the refresh with MQCFO\_REFRESH\_REPOSITORY\_YES has been issued the queue manager can be altered so that it is once again a full repository.

### MQCFO\_REFRESH\_REPOSITORY

Do not refresh repository information. MQCFO\_REFRESH\_REPOSITORY is the default.

If you select MQCFO\_REFRESH\_REPOSITORY\_YES, check that all cluster-sender channels in the relevant cluster are inactive or stopped before you issue the Refresh Cluster command. If there are cluster-sender channels running at the time when the Refresh is processed, and they are used exclusively by the cluster or clusters being refreshed and MQCFO\_REFRESH\_REPOSITORY\_YES is used, the channels are stopped, by using the Stop Channel command with a value of MQMODE\_FORCE in the **Mode** parameter if necessary.

This scenario ensures that the Refresh can remove the channel state and that the channel will run with the refreshed version after the Refresh has completed. If the state of a channel cannot be deleted, for example because it is in doubt, or because it is also running as part of another cluster, its state is not new after the refresh and it does not automatically restart if it was stopped.

## Related information

[Clustering: Using REFRESH CLUSTER best practices](#)

## MQCMD\_REFRESH\_Q\_MGR (Refresh Queue Manager)

Use the Refresh Queue Manager (MQCMD\_REFRESH\_Q\_MGR) PCF command to perform special operations on queue managers.

## Required parameters

### RefreshType (MQCFIN)

Type of information to be refreshed (parameter identifier: MQIACF\_REFRESH\_TYPE).

Use this parameter to specify the type of information to be refreshed. The value can be any of the following values:

### MQRT\_CONFIGURATION

MQRT\_CONFIGURATION causes the queue manager to generate configuration event messages for every object definition that matches the selection criteria specified by the **ObjectType**, **ObjectName**, and **RefreshInterval** parameters.

A Refresh Queue Manager command with a **RefreshType** value of MQRT\_CONFIGURATION is generated automatically when the value of the queue manager's **ConfigurationEvent** parameter changes from MQEVR\_DISABLED to MQEVR\_ENABLED.

Use this command with a **RefreshType** of MQRT\_CONFIGURATION to recover from problems such as errors on the event queue. In such cases, use appropriate selection criteria, to avoid excessive processing time and event message generation.

## **MQRT\_EXPIRY**


This requests that the queue manager performs a scan to discard expired messages for every queue that matches the selection criteria specified by the **ObjectName** parameter.

**Note:**  Valid only on z/OS.

## **MQRT\_EARLY**

Requests that the subsystem function routines (generally known as early code) for the queue manager replace themselves with the corresponding routines in the linkpack area (LPA).

You need to use this command only after you install new subsystem function routines (provided as corrective maintenance or with a new version or release of IBM MQ). This command instructs the queue manager to use the new routines.

 See [Task 3: Update the z/OS link list and LPA](#) for more information about IBM MQ early code routines.

## **MQRT\_PROXYSUB**

Requests that the queue manager resynchronizes the proxy subscriptions that are held with, and on behalf of, queue managers that are connected in a hierarchy or publish/subscribe cluster.

You should only resynchronize the proxy subscriptions in exceptional circumstances. See [Resynchronization of proxy subscriptions](#).

## **Optional parameters (Refresh Queue Manager)**

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **ObjectName (MQCFST)**

Name of object to be included in the processing of this command (parameter identifier: MQCACF\_OBJECT\_NAME).

Use this parameter to specify the name of the object to be included in the processing of this command.

Generic names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length is MQ\_OBJECT\_NAME\_LENGTH.

### **ObjectType (MQCFIN)**

Object type for which configuration data is to be refreshed (parameter identifier: MQIACF\_OBJECT\_TYPE).

Use this parameter to specify the object type for which configuration data is to be refreshed. This parameter is valid only if the value of *RefreshType* is MQRT\_CONFIGURATION. The default value, in that case, is MQOT\_ALL. The value can be one of:

**MQOT\_AUTH\_INFO**

Authentication information object.

**MQOT\_CF\_STRUC**

CF structure.

**MQOT\_CHANNEL**

Channel.

**MQOT\_CHLAUTH**

Channel authentication

**MQOT\_LISTENER**

Listener.

**MQOT\_NAMELIST**

Namelist.

**MQOT\_PROCESS**

Process definition.

**MQOT\_Q**

Queue.

**MQOT\_LOCAL\_Q**

Local queue.

**MQOT\_MODEL\_Q**

Model queue.

**MQOT\_ALIAS\_Q**

Alias queue.

**MQOT\_REMOTE\_Q**

Remote queue.

**MQOT\_Q\_MGR**

Queue manager.

**MQOT\_CFSTRUC**

CF structure.

**MQOT\_SERVICE**

Service.

**Note:**  z/OS Not valid on z/OS.

**MQOT\_STORAGE\_CLASS**

Storage class.

**MQOT\_TOPIC**

Topic name.

**RefreshInterval (MQCFIN)**

Refresh interval (parameter identifier: MQIACF\_REFRESH\_INTERVAL).

Use this parameter to specify a value, in minutes, defining a period immediately before the current time. This requests that only objects that have been created or altered within that period (as defined by their *AlterationDate* and **AlterationTime** attributes) are included.

Specify a value in the range zero through 999 999. A value of zero means there is no time limit (0 is the default).

This parameter is valid only if the value of *RefreshType* is MQRT\_CONFIGURATION.

**Usage Notes for Refresh Queue Manager**

1. Issue this command with *RefreshType*(MQRT\_CONFIGURATION) after setting the MQRT\_CONFIGURATION queue manager attribute to ENABLED, to bring the queue manager configuration up to date. To ensure that complete configuration information is generated, include

all objects; if you have many objects, it might be preferable to use several commands, each with a different selection of objects, but such that all are included.

2. You can also use the command with *RefreshType*(MQRT\_CONFIGURATION) to recover from problems such as errors on the event queue. In such cases, use appropriate selection criteria, to avoid excessive processing time and event messages generation.
3. Issue the command with *RefreshType* (MQRT\_EXPIRY) at any time when you believe that a queue could contain numbers of expired messages.
4. If *RefreshType* (MQRT\_EARLY) is specified, no other keywords are allowed and the command can be issued only from the z/OS console and only if the queue manager is not active.
5. You are unlikely to use **Refresh Queue Manager RefreshType (MQRT\_PROXYSUB)** other than in exceptional circumstances. See [Resynchronization of proxy subscriptions](#).
6. If a **Refresh Queue Manager Object Type(MQRT\_PROXYSUB)** command is issued on z/OS when the CHINIT is not running, the command is queued up and will be processed when the CHINIT starts.
7. Running the command Refresh Queue Manager RefreshType (MQRT\_CONFIGURATION) Object Type(MQOT\_ALL) includes authority records.

You cannot specify the **Refresh Interval** and **Object Name** parameters if you explicitly specify Authority Record events. If you specify **Object Type(MQOT\_ALL)** the **Refresh Interval** and **Object Name** parameters are ignored.

### **MQCMD\_REFRESH\_SECURITY (Refresh Security)**

The Refresh Security (MQCMD\_REFRESH\_SECURITY) PCF command refreshes the list of authorizations held internally by the authorization service component.

### **Optional parameters**

#### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

#### **z/OS SecurityItem (MQCFIN)**

Resource class for which the security refresh is to be performed (parameter identifier: MQIACF\_SECURITY\_ITEM). This parameter applies to z/OS only.

Use this parameter to specify the resource class for which the security refresh is to be performed. The value can be any of the following values:

##### **MQSECITEM\_ALL**

A full refresh of the type specified is performed. MQSECITEM\_ALL is the default value.

##### **MQSECITEM\_MQADMIN**

Specifies that administration type resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**MQSECITEM\_MQNLIST**

Specifies that namelist resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**MQSECITEM\_MQPROC**

Specifies that process resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**MQSECITEM\_MQQUEUE**

Specifies that queue resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**MQSECITEM\_MXADMIN**

Specifies that administration type resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**MQSECITEM\_MXNLIST**

Specifies that namelist resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**MQSECITEM\_MXPROC**

Specifies that process resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**MQSECITEM\_MXQUEUE**

Specifies that queue resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**MQSECITEM\_MXTOPIC**

Specifies that topic resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE\_CLASSES.

**SecurityType (MQCFIN)**

Security type (parameter identifier: MQIACF\_SECURITY\_TYPE).

Use this parameter to specify the type of security refresh to be performed. The value can be any of the following values:

**MQSECTYPE\_AUTHSERV**

The list of authorizations held internally by the authorization services component is refreshed. MQSECTYPE\_AUTHSERV is not valid on z/OS.

MQSECTYPE\_AUTHSERV is the default on platforms other than z/OS.


**MQSECTYPE\_CLASSES**

Permits you to select specific resource classes for which to perform the security refresh.

 MQSECTYPE\_CLASSES is valid only on z/OS where it is the default.

**MQSECTYPE\_CONNAUTH**

Refreshes the cached view of the configuration for connection authentication.

 On [Multiplatforms](#) this is also a synonym for MQSECTYPE\_AUTHSERV.

**MQSECTYPE\_SSL**

MQSECTYPE\_SSL refreshes the locations of the LDAP servers to be used for Certified Revocation Lists and the key repository. It also refreshes any cryptographic hardware parameters specified through IBM MQ and the cached view of the Secure Sockets Layer key repository. It also allows updates to become effective on successful completion of the command.

MQSECTYPE\_SSL updates all TLS channels currently running, as follows:

- Sender, server, and cluster-sender channels using TLS are allowed to complete the current batch. In general, they then run the TLS handshake again with the refreshed view of the TLS key repository. However, you must manually restart a requester-server channel on which the server definition has no CONNAME parameter.

- AMQP channels using TLS are restarted, with any currently connected clients being forcibly disconnected. The client receives an `amqp:connection:forced` AMQP error message.
- All other channel types using TLS are stopped with a `STOP CHANNEL MODE(FORCE) STATUS(INACTIVE)` command. If the partner end of the stopped message channel has retry values defined, the channel tries again and the new TLS handshake uses the refreshed view of the contents of the TLS key repository, the location of the LDAP server to be used for Certification Revocation Lists, and the location of the key repository. If there is a server-connection channel, the client application loses its connection to the queue manager and must reconnect in order to continue.

### **MQCMD\_RESET\_CF\_STRUC (Reset coupling facility structure) on z/OS**

The Reset coupling facility (CF) structure (`MQCMD_RESET_CF_STRUC`) PCF command modifies the status of a specific application structure.

#### **Required parameters**

##### **CFStructName (MQCFST)**

The name of the coupling facility application structure that you want to reset (parameter identifier: `MQCA_CF_STRUC_NAME`). The maximum length of the string is `MQ_CF_STRUC_NAME_LENGTH`.

##### **Action (MQCFIN)**

The action to perform to reset the named application structure (parameter identifier: `MQIACF_ACTION`).

##### **MQACT\_FAIL**

A structure failure is simulated and the status of the application structure is set to `FAILED`.

**Note:** Failing a structure deletes all nonpersistent messages stored in the structure, and makes the structure unavailable until recovery is complete. Structure recovery can take a long time to complete. Therefore, this action should be used only in a situation where you can resolve a problem with the structure by forcing the structure to be reallocated and recovered.

### **MQCMD\_RESET\_CHANNEL (Reset Channel)**

The Reset Channel (`MQCMD_RESET_CHANNEL`) PCF command resets the message sequence number for an IBM MQ channel with, optionally, a specified sequence number to be used the next time that the channel is started.

This command can be issued to a channel of any type (except `MQCHT_SVRCONN` and `MQCHT_CLNTCONN`). However, if it is issued to a sender (`MQCHT_SENDER`), server (`MQCHT_SERVER`), or cluster-sender (`MQCHT_CLUSSDR`) channel, the value at both ends (issuing end and receiver or requester end), is reset when the channel is next initiated or resynchronized. The value at both ends is reset to be equal.

If the command is issued to a receiver (`MQCHT_RECEIVER`), requester (`MQCHT_REQUESTER`), or cluster-receiver (`MQCHT_CLUSRCVR`) channel, the value at the other end is not reset as well; this step must be done separately if necessary.

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

#### **Required parameters**

##### **ChannelName (MQCFST)**

Channel name (parameter identifier: `MQCACH_CHANNEL_NAME`).

The name of the channel to be reset. The maximum length of the string is `MQ_CHANNEL_NAME_LENGTH`.



## Optional parameters

### **z/OS** **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **z/OS** **ChannelDisposition (MQCFIN)**

Channel disposition (parameter identifier: MQIACH\_CHANNEL\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be reset.

If this parameter is omitted, then the value for the channel disposition is taken from the default channel disposition attribute of the channel object.

The value can be any of the following values:

#### **MQCHLD\_PRIVATE**

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD\_SHARED.

#### **MQCHLD\_SHARED**

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD\_SHARED.

The combination of the **ChannelDisposition** and **CommandScope** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in [Table 212 on page 1465](#)

<b>ChannelDisposition</b>	<b>CommandScope blank or local-qmgr</b>	<b>CommandScope qmgr-name</b>
MQCHLD_PRIVATE	Reset private channel on the local queue manager	Reset private channel on the named queue manager

<i>Table 212. ChannelDisposition and CommandScope for RESET CHANNEL (continued)</i>		
<b>ChannelDisposition</b>	<b>CommandScope blank or local-qmgr</b>	<b>CommandScope qmgr-name</b>
MQCHLD_SHARED	<p>Reset a shared channel on all active queue managers.</p> <p>MQCHLD_SHARED might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted

### **MsgSeqNumber (MQCFIN)**

Message sequence number (parameter identifier: MQIACH\_MSG\_SEQUENCE\_NUMBER).

Specifies the new message sequence number.

The value must be in the range 1 through 999 999 999. The default value is one.

### **Error codes**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

#### **Reason (MQLONG)**

The value can be any of the following values:

##### **MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

### **MQCMD\_RESET\_CLUSTER (Reset Cluster)**

The Reset Cluster (MQCMD\_RESET\_CLUSTER) PCF command forces a queue manager to leave a cluster.

### **Required parameters**

#### **ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

The name of the cluster to be reset.

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

#### **QMgrIdentifier (MQCFST)**

Queue manager identifier (parameter identifier: MQCA\_Q\_MGR\_IDENTIFIER).

This parameter is the unique identifier of the queue manager to be forcibly removed from the cluster. Only one of *QMgrIdentifier* and *QMgrName* can be specified. Use *QMgrIdentifier* in preference to *QMgrName*, because *QMgrName* might not be unique.

#### **QMgrName (MQCFST)**

Queue manager name (parameter identifier: MQCA\_Q\_MGR\_NAME).

This parameter is the name of the queue manager to be forcibly removed from the cluster. Only one of `QMgrIdentifier` and `QMgrName` can be specified. Use `QMgrIdentifier` in preference to `QMgrName`, because `QMgrName` might not be unique.

### Action (MQCFIN)

Action (parameter identifier: `MQIACF_ACTION`).

Specifies the action to take place. This parameter can be requested only by a repository queue manager.

The value can be any of the following values:

#### **MQACT\_FORCE\_REMOVE**

Requests that a queue manager is forcibly removed from a cluster.

## Optional parameters

### **CommandScope (MQCFST)**

Command scope (parameter identifier: `MQCACF_COMMAND_SCOPE`). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is `MQ_QSG_NAME_LENGTH`.

### **RemoveQueues (MQCFIN)**

Whether cluster queues are removed from the cluster (parameter identifier: `MQIACF_REMOVE_QUEUES`).

This parameter indicates whether the cluster queues that belong to the queue manager being removed from the cluster are to be removed from the cluster. This parameter can be specified even if the queue manager identified by the `QMgrName` parameter is not currently in the cluster.

The value can be any of the following values:

#### **MQCFO\_REMOVE\_QUEUES\_YES**

Remove queues belonging to the queue manager being removed from the cluster.

#### **MQCFO\_REMOVE\_QUEUES\_NO**

Do not remove queues belonging to the queue manager being removed.  
`MQCFO_REMOVE_QUEUES_NO` is the default.

## Error codes

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRCCF\_ACTION\_VALUE\_ERROR**

Value not valid.

### **MQCMD\_RESET\_Q\_MGR (Reset Queue Manager)**

Use the Reset Queue Manager (`MQCMD_RESET_Q_MGR`) PCF command as part of your backup and recovery procedures. The **Archive** option enables you to notify the queue manager that all log extents,

up to the specified one, have been archived. If the log management type is not **ArchivedLog** the command fails. The **ReduceLog** option enables you to request that the queue manager reduces the number of log extents, provided they are no longer required.

You can use this command to request that the queue manager starts writing to a new log extent, making the previous log extent available for archiving.

Use the Reset Queue Manager (MQCMD\_RESET\_Q\_MGR) command to forcibly remove a publish/subscribe hierarchical connection for which this queue manager is nominated as either the parent or the child in a hierarchical connection. Valid on all supported platforms.

## Archive option

This option requires change authority on the queue manager object.

The command fails if the log extent is not recognized, or is being written.

If, for some reason, the programmatic way that your enterprise notifies your log extents are archived is not working, and the disk is filling up with log extents, your administrator can use this command.

You need to determine yourself, the name to pass in from your archiving process, as to what has already been archived.

This option is not valid on IBM i.

## ReduceLog option

This option requires change authority on the queue manager object.

You should not need this command in normal circumstances. In general, when using automatic management of log files, you should leave it up to the queue manager to reduce the number of log extents as necessary.

For circular logging, this can remove inactive secondary log extents. The increase in secondary log extents is usually noticed by an increase in disk usage, often due to some specific issue in the past.

**Note:** For circular logging the command might not be able reduce the log extents by the required number immediately. In that case, the command returns, and the reduction takes place asynchronously at some later point.

For linear logging this can remove log extents that are not required for recovery (and have been archived) as noticed by a high value for [ReusableLogSize](#) on the Inquire Queue Manager Status command.

You should run this command only after some specific event that has caused the number of log extents to be extraordinarily large.

The command blocks until the chosen number of extents have been deleted. Note that the command does not return the number of extents that have been removed, but a queue manager error log message is written, indicating what has taken place.

This option is not valid on IBM i.

## Required parameters

### Action (MQCFIN)

Action (parameter identifier: MQIACF\_ACTION).

Specifies the action to take place.

The value can be any of the following values, but you can specify one only:

### MQACT\_ADVANCE\_LOG

Requests that the queue manager starts writing to a new log extent, making the previous log extent available for archiving. This command is accepted only if the queue manager is configured to use linear logging.

**Note:** Not valid on z/OS.

### **MQACT\_COLLECT\_STATISTICS**

Requests that the queue manager ends the current statistics collection period, and writes the statistics collected.

**Note:** Not valid on z/OS.

### **MQACT\_PUBSUB**

Requests a publish/subscribe reset. This value requires that one of the optional parameters, ChildName or ParentName, is specified.

### **MQACT\_ARCHIVE\_LOG (11)**

Requests that log extents are archived.

The command fails if the log extent is not recognized, or is the current log.

If, for some reason, the programmatic way that your enterprise notifies your log extents are archived is not working, and the disk is filling up with log extents, your administrator can use this command.

### **MQACT\_REDUCE\_LOG (10)**

You should not need this command in normal circumstances. In general, when using automatic management of log files, you should leave it up to the queue manager to reduce the number of log extents as necessary.

For circular logging, you can use this option to remove inactive secondary log extents. A growth in secondary log extents is usually noticed by an increase in disk usage, often due to some specific issue in the past.

You should run this command only after some specific event that has caused the number of log extents to be extraordinarily large.

The command blocks until the chosen number of extents have been deleted. Note that the command does not return the number of extents that have been removed, but a queue manager error log message is written, indicating what has taken place.

## **Optional parameters**

### **ArchivedLog (MQCFST)**

Specifies the name of the log extent to be archived (parameter identifier: MQCACF\_ARCHIVE\_LOG\_EXTENT\_NAME).

The maximum length of the string is MQ\_LOG\_EXTENT\_NAME\_LENGTH.

### **ChildName (MQCFST)**

The name of the child queue manager for which the hierarchical connection is to be forcibly canceled (parameter identifier: MQCA\_CHILD).

This attribute is valid only when the Action parameter has the value MQACT\_PUBSUB.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

### **ParentName (MQCFST)**

The name of the parent queue manager for which the hierarchical connection is to be forcibly canceled (parameter identifier: MQCA\_PARENT).

This attribute is valid only when the Action parameter has the value MQACT\_PUBSUB.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

### **LogReduction (MQCFIN)**

Specifies that the type of log reduction (parameter identifier: MQIACF\_LOG\_REDUCTION).

The value can be one of:

**MQLR\_AUTO**

-1. The default value. Reduce the log extents by an amount chosen by the queue manager.

**MQLR\_ONE**

1. Reduce the log extents by one extent, if possible.

**MQLR\_MAX**

-2. Reduce the log extents by the maximum number possible.

**Error codes**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CURRENT\_LOG\_EXTENT**

The specified log extent is the current log extent, and cannot have been validly archived yet.

**MQRCCF\_LOG\_EXTENT\_NOT\_FOUND**

The specified log extent was not found or is not valid.

**MQRCCF\_LOG\_NOT\_REDUCED**

No log events could be removed.

**MQRC\_RESOURCE\_PROBLEM**

Insufficient system resources available.

***MQCMD\_RESET\_Q\_STATS (Reset Queue Statistics)***

The Reset Queue Statistics (MQCMD\_RESET\_Q\_STATS) PCF command reports the performance data for a queue and then resets the performance data. Performance data is maintained for each local queue (including transmission queues).

Performance data is reset at the following times:

- When a Reset Queue Statistics command is issued
- When the queue manager is restarted
- When a performance event is generated for a queue

**Required parameters****QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

The name of the local queue to be tested and reset.

Generic queue names are supported. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**Optional parameters**** CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.

- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## Error codes

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRCCF\_Q\_WRONG\_TYPE**

Action not valid for the queue of specified type.

#### **MQRCCF\_EVENTS\_DISABLED**

The queue manager performance events are disabled (PERFMEV). On z/OS, you must enable queue manager performance events to use this command. For more details, see the **PerformanceEvent** property in the [“MQCMD\\_CHANGE\\_Q\\_MGR \(Change Queue Manager\)”](#) on page 1105 command.

## **MQCMD\_RESET\_Q\_STATS (Reset Queue Statistics) Response**

The response to the Reset Queue Statistics (MQCMD\_RESET\_Q\_STATS) PCF command consists of the response header followed by the *QName* structure and the attribute parameter structures shown in the following sections.

If a generic queue name was specified, one such message is generated for each queue found.

### Always returned:

*HighQDepth, MsgDeqCount, MsgEnqCount, QName, z/OS QSGDisposition, TimeSinceReset*

## Response data

### HighQDepth (MQCFIN)

Maximum number of messages on a queue (parameter identifier: MQIA\_HIGH\_Q\_DEPTH).

This count is the peak value of the *CurrentQDepth* local queue attribute since the last reset. The *CurrentQDepth* is incremented during an MQPUT call, and during backout of an MQGET call, and is decremented during a (nonbrowse) MQGET call, and during backout of an MQPUT call.

### MsgDeqCount (MQCFIN)

Number of messages dequeued (parameter identifier: MQIA\_MSG\_DEQ\_COUNT).

This count includes messages that have been successfully retrieved (with a nonbrowse MQGET) from the queue, even though the MQGET has not yet been committed. The count is not decremented if the MQGET is later backed out.

**z/OS** On z/OS, if the value exceeds 999 999 999, it is returned as 999 999 999

### MsgEnqCount (MQCFIN)

Number of messages enqueued (parameter identifier: MQIA\_MSG\_ENQ\_COUNT).

This count includes messages that have been put to the queue, but have not yet been committed. The count is not decremented if the put is later backed out.

**z/OS** On z/OS, if the value exceeds 999 999 999, it is returned as 999 999 999

**QName (MQCFST)**

Queue name (parameter identifier: MQCA\_Q\_NAME).

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**z/OS QSGDisposition (MQCFIN)**

QSG disposition (parameter identifier: MQIA\_QSG\_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be any of the following values:

**MQQSGD\_COPY**

The object is defined as MQQSGD\_COPY.

**MQQSGD\_SHARED**

The object is defined as MQQSGD\_SHARED.

**MQQSGD\_Q\_MGR**

The object is defined as MQQSGD\_Q\_MGR.

**TimeSinceReset (MQCFIN)**

Time since statistics reset in seconds (parameter identifier: MQIA\_TIME\_SINCE\_RESET).

**z/OS MQCMD\_RESET\_SMDS (Reset shared message data sets) on z/OS**

The Reset SMDS (MQCMD\_RESET\_SMDS) PCF command modifies the availability or status information relating to one or more shared message data sets associated with a specific application structure

**Required parameters****SMDS (MQCFST)**

Specifies the queue manager for which the shared message data set availability or status information is to be modified or an asterisk to modify the information for all data sets associated with the specified CFSTRUCT. (parameter identifier: MQCACF\_CF\_SMDS).

The maximum length of the string is 4 characters.

**CFStrucName (MQCFST)**

The name of the CF application structure with SMDS connections properties that you want to reset (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

**Optional parameters****Access (MQCFIN)**

Availability of the share message data set (parameter identifier: MQIACF\_CF\_STRUC\_ACCESS).

**MQCFACCESS\_ENABLED**

The shared message data set is available for use.

**MQCFACCESS\_DISABLED**

The shared message data set is disabled.

**Status (MQCFIN)**

Status information indicates the state of a resource (parameter identifier: MQIACF\_CF\_STRUC\_STATUS).

**MQCFSTATUS\_FAILED**

The shared message data set is in an unusable state.

**MQCFSTATUS\_RECOVERED**

The data set is set to recovered, and is ready for use again, but requires some restart processing the next time it is opened. This restart processing ensures that obsolete references to any deleted messages have been removed from the coupling facility structure before the data set is made available again. The restart processing also rebuilds the data set space map.



## ***MQCMD\_RESOLVE\_CHANNEL (Resolve Channel)***

The Resolve Channel (MQCMD\_RESOLVE\_CHANNEL) PCF command requests a channel to commit or back out in-doubt messages. This command is used when the other end of a link fails during the confirmation stage, and for some reason it is not possible to reestablish the connection. In this situation the sending end remains in an in-doubt state, whether the messages were received. Any outstanding units of work must be resolved using Resolve Channel with either backout or commit.

Care must be exercised in the use of this command. If the resolution specified is not the same as the resolution at the receiving end, messages can be lost or duplicated.

This command can only be used for channels with a *ChannelType* value of MQCHT\_SENDER, MQCHT\_SERVER, or MQCHT\_CLUSSDR.

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

### **Required parameters**

#### **ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The name of the channel to be resolved. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

#### **InDoubt (MQCFIN)**

Indoubt resolution (parameter identifier: MQIACH\_IN\_DOUBT).

Specifies whether to commit or back out the in-doubt messages.

The value can be:

#### **MQIDO\_COMMIT**

Commit.

#### **MQIDO\_BACKOUT**

Backout.

### **Optional parameters**

#### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

#### **z/OS ChannelDisposition (MQCFIN)**

Channel disposition (parameter identifier: MQIACH\_CHANNEL\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be resolved.

If this parameter is omitted, then the value for the channel disposition is taken from the default channel disposition attribute of the channel object.

The value can be any of the following values:

**MQCHLD\_PRIVATE**

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD\_SHARED.

**MQCHLD\_SHARED**

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD\_SHARED.

The combination of the **ChannelDisposition** and **CommandScope** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in [Table 213 on page 1474](#)

<i>Table 213. ChannelDisposition and CommandScope for RESOLVE CHANNEL</i>		
<b>ChannelDisposition</b>	<b>CommandScope blank or local-qmgr</b>	<b>CommandScope qmgr-name</b>
MQCHLD_PRIVATE	Resolve private channel on the local queue manager	Resolve private channel on the named queue manager
MQCHLD_SHARED	Resolve a shared channel on all active queue managers.  MQCHLD_SHARED might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.  The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.	Not permitted

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

## **MQRCCF\_INDOUBT\_VALUE\_ERROR**

In-doubt value not valid.

## **z/OS MQCMD\_RESUME\_Q\_MGR (Resume Queue Manager) on z/OS**

The Resume Queue Manager (MQCMD\_RESUME\_Q\_MGR) PCF command renders the queue manager available again for the processing of IMS or Db2 messages. It reverses the action of the Suspend Queue Manager (MQCMD\_SUSPEND\_Q\_MGR) command.

### **Required parameters**

#### **Facility (MQCFIN)**

Facility (parameter identifier: MQIACF\_Q\_MGR\_FACILITY).

The type of facility for which activity is to be resumed. The value can be:

#### **MQQMFACT\_DB2**

Resumes normal activity with Db2.

#### **MQQMFACT\_IMS\_BRIDGE**

Resumes normal IMS bridge activity.

### **Optional parameters**

#### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## **MQCMD\_RESUME\_Q\_MGR\_CLUSTER (Resume Queue Manager Cluster)**

The Resume Queue Manager Cluster (MQCMD\_RESUME\_Q\_MGR\_CLUSTER) PCF command informs other queue managers in a cluster that the local queue manager is again available for processing, and can be sent messages. It reverses the action of the Suspend Queue Manager Cluster (MQCMD\_SUSPEND\_Q\_MGR\_CLUSTER) command.

### **Required parameters**

#### **ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

The name of the cluster for which availability is to be resumed.

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

#### **ClusterNamelist (MQCFST)**

Cluster Namelist (parameter identifier: MQCA\_CLUSTER\_NAMELIST).

The name of the namelist specifying a list of clusters for which availability is to be resumed.

## Optional parameters

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## Error codes

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRCCF\_CLUSTER\_NAME\_CONFLICT**

Cluster name conflict.

### **MQCMD\_REVERIFY\_SECURITY (Reverify Security) on z/OS**

The Reverify Security (MQCMD\_REVERIFY\_SECURITY) PCF command sets a reverification flag for all specified users. The user is reverified the next time that security is checked for that user.

## Required parameters

### **UserId (MQCFST)**

User ID (parameter identifier: MQCACF\_USER\_IDENTIFIER).

Use this parameter to specify one or more user IDs. Each user ID specified is signed off and signed back on again the next time that a request requiring a security check is issued on behalf of that user.

The maximum length of the string is MQ\_USER\_ID\_LENGTH.

## Optional parameters

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## **MQCMD\_SET\_ARCHIVE (Set Archive) on z/OS**

The Set Archive (MQCMD\_SET\_ARCHIVE) PCF command dynamically changes certain archive system parameter values initially set by your system parameter module at queue manager startup.

### **Required parameters**

#### **ParameterType (MQCFIN)**

Parameter type (parameter identifier: MQIACF\_SYSP\_TYPE).

Specifies how the parameters are to be reset:

#### **MQSYSP\_TYPE\_INITIAL**

The initial settings of the archive system parameters. MQSYSP\_TYPE\_INITIAL resets all the archive system parameters to the values set at queue manager startup.

#### **MQSYSP\_TYPE\_SET**

MQSYSP\_TYPE\_SET indicates that you intend to change one, or more, of the archive system parameter settings.

### **Optional parameters**

#### **AllocPrimary (MQCFIN)**

Primary space allocation for DASD data sets (parameter identifier: MQIACF\_SYSP\_ALLOC\_PRIMARY).

Specifies the primary space allocation for DASD data sets in the units specified in the **AllocUnits** parameter.

Specify a value greater than zero. This value must be sufficient for a copy of either the log data set or its corresponding BSDS, whichever is the larger.

#### **AllocSecondary (MQCFIN)**

Secondary space allocation for DASD data sets (parameter identifier: MQIACF\_SYSP\_ALLOC\_SECONDARY).

Specifies the secondary space allocation for DASD data sets in the units specified in the **AllocUnits** parameter.

Specify a value greater than zero.

#### **AllocUnits (MQCFIN)**

Allocation unit (parameter identifier: MQIACF\_SYSP\_ALLOC\_UNIT).

Specifies the unit in which primary and secondary space allocations are made. The value can be any of the following values:

#### **MQSYSP\_ALLOC\_BLK**

Blocks.

#### **MQSYSP\_ALLOC\_TRK**

Tracks.

#### **MQSYSP\_ALLOC\_CYL**

Cylinders.

#### **ArchivePrefix1 (MQCFST)**

Specifies the prefix for the first archive log data set name (parameter identifier: MQCACF\_SYSP\_ARCHIVE\_PFX1).

The maximum length of the string is MQ\_ARCHIVE\_PFX\_LENGTH.

#### **ArchivePrefix2 (MQCFST)**

Specifies the prefix for the second archive log data set name (parameter identifier: MQCACF\_SYSP\_ARCHIVE\_PFX2).

The maximum length of the string is MQ\_ARCHIVE\_PFX\_LENGTH.

**ArchiveRetention (MQCFIN)**

Archive retention period (parameter identifier: MQIACF\_SYSP\_ARCHIVE\_RETAIN).

Specifies the retention period, in days, to be used when the archive log data set is created. Specify a value in the range zero through 9999.

For more information, see [Discarding archive log data sets](#).

**ArchiveUnit1 (MQCFST)**

Specifies the device type or unit name of the device that is used to store the first copy of the archive log data set (parameter identifier: MQCACF\_SYSP\_ARCHIVE\_UNIT1).

Specify a device type or unit name of 1-8 characters.

If you archive to DASD, you can specify a generic device type with a limited volume range.

The maximum length of the string is MQ\_ARCHIVE\_UNIT\_LENGTH.

**ArchiveUnit2 (MQCFST)**

Specifies the device type or unit name of the device that is used to store the second copy of the archive log data set (parameter identifier: MQCACF\_SYSP\_ARCHIVE\_UNIT2).

Specify a device type or unit name of 1-8 characters.

If this parameter is blank, the value set for the **ArchiveUnit1** parameter is used.

The maximum length of the string is MQ\_ARCHIVE\_UNIT\_LENGTH.

**ArchiveWTOR (MQCFIN)**

Specifies whether a message is to be sent to the operator and a reply is received before attempting to mount an archive log data set (parameter identifier: MQIACF\_SYSP\_ARCHIVE\_WTOR).

Other IBM MQ users might be forced to wait until the data set is mounted, but they are not affected while IBM MQ is waiting for the reply to the message.

The value can be any of the following values:

**MQSYSP\_YES**

A message is to be sent and a reply received before an attempt to mount an archive log data set.

**MQSYSP\_NO**

A message is not to be sent and a reply received before an attempt to mount an archive log data set.

**BlockSize (MQCFIN)**

Block size of the archive log data set (parameter identifier: MQIACF\_SYSP\_BLOCK\_SIZE).

The block size you specify must be compatible with the device type you specify in the **ArchiveUnit1** and **ArchiveUnit2** parameters.

Specify a value in the range 4 097 through 28 672. The value you specify is rounded up to a multiple of 4 096.

This parameter is ignored for data sets that are managed by the storage management system (SMS).

**Catalog (MQCFIN)**

Specifies whether archive log data sets are cataloged in the primary integrated catalog facility (parameter identifier: MQIACF\_SYSP\_CATALOG).

The value can be:

**MQSYSP\_YES**

Archive log data sets are cataloged.

**MQSYSP\_NO**

Archive log data sets are not cataloged.

**CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **Compact (MQCFIN)**

Specifies whether data written to archive logs is to be compacted (parameter identifier: MQIACF\_SYSP\_COMPACT).

This parameter applies to a 3480 or 3490 device that has the improved data recording capability (IDRC) feature. When this feature is turned on, hardware in the tape control unit writes data at a much higher density than normal, allowing for more data on each volume. Specify MQSYSP\_NO if you do not use a 3480 device with the IDRC feature or a 3490 base model, except for the 3490E. Specify MQSYSP\_YES if you want the data to be compacted.

The value can be:

#### **MQSYSP\_YES**

Data is to be compacted.

#### **MQSYSP\_NO**

Data is not to be compacted.

### **Protect (MQCFIN)**

Protection by external security manager (ESM) (parameter identifier: MQIACF\_SYSP\_PROTECT).

Specifies whether archive log data sets are protected by ESM profiles when the data sets are created.

If you specify MQSYSP\_YES, ensure that:

- ESM protection is active for IBM MQ.
- The user ID associated with the IBM MQ address space has authority to create these profiles.
- The TAPEVOL class is active if you are archiving to tape.

otherwise, offload processing fails.

The value can be any of the following values:

#### **MQSYSP\_YES**

Data set profiles are created when logs are offloaded.

#### **MQSYSP\_NO**

Profiles are not created.

### **QuiesceInterval (MQCFIN)**

Maximum time allowed for the quiesce (parameter identifier: MQIACF\_SYSP\_QUIESCE\_INTERVAL).

Specifies the maximum time, in seconds, allowed for the quiesce.

Specify a value in the range 1 through 999.

### **RoutingCode (MQCFIL)**

z/OS routing code list (parameter identifier: MQIACF\_SYSP\_ROUTING\_CODE).

Specifies the list of z/OS routing codes for messages about the archive log data sets to the operator.

Specify up to 14 routing codes, each with a value in the range zero through 16. You must specify at least one code.

### **TimeStampFormat (MQCFIN)**

Time stamp included (parameter identifier: MQIACF\_SYSP\_TIMESTAMP).

Specifies whether the archive log data set name has a time stamp in it.

The value can be:

#### **MQSYSP\_YES**

Names include a time stamp. The archive log data sets are named:

```
arcpfxi.cyyddd.T hhmsst.A nnnnnnn
```

where *c* is 'D' for the years up to and including 1999 or 'E' for the year 2000 and later, and *arcpfxi* is the data set name prefix specified by *ArchivePrefix1* or *ArchivePrefix2*. *arcpfxi* can have up to 19 characters.

#### **MQSYSP\_NO**

Names do not include a time stamp. The archive log data sets are named:

```
arcpfxi.A nnnnnnn
```

Where *arcpfxi* is the data set name prefix specified by *ArchivePrefix1* or *ArchivePrefix2*. *arcpfxi* can have up to 35 characters.

#### **MQSYSP\_EXTENDED**

Names include a time stamp. The archive log data sets are named:

```
arcpfxi.D yyyyddd.T hhmsst.A nnnnnnn
```

Where *arcpfxi* is the data set name prefix specified by *ArchivePrefix1* or *ArchivePrefix2*. *arcpfxi* can have up to 17 characters.

### **Multi MQCMD\_SET\_AUTH\_REC (Set Authority Record) on Multiplatforms**

The Set Authority Record (MQCMD\_SET\_AUTH\_REC) PCF command sets the authorizations of a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

### **Required parameters**

#### **ProfileName (MQCFST)**

Profile name (parameter identifier: MQCACF\_AUTH\_PROFILE\_NAME).

The authorizations apply to all IBM MQ objects with names that match the profile name specified. You can define a generic profile. If you specify an explicit profile name, the object must exist.

The maximum length of the string is MQ\_AUTH\_PROFILE\_NAME\_LENGTH.

#### **ObjectType (MQCFIN)**

The type of object for which to set authorizations (parameter identifier: MQIACF\_OBJECT\_TYPE).

The value can be any of the following values:

##### **MQOT\_AUTH\_INFO**

Authentication information.

##### **MQOT\_CHANNEL**

Channel object.

##### **MQOT\_CLNTCONN\_CHANNEL**

Client-connection channel object.

##### **MQOT\_COMM\_INFO**

Communication information object



**MQOT\_LISTENER**

Listener object.

**MQOT\_NAMELIST**

Namelist.

**MQOT\_PROCESS**

Process.

**MQOT\_Q**

Queue, or queues, that match the object name parameter.

**MQOT\_Q\_MGR**

Queue manager.

**MQOT\_REMOTE\_Q\_MGR\_NAME**

Remote queue manager.

**MQOT\_SERVICE**

Service object.

**MQOT\_TOPIC**

Topic object.

**Note:** The required parameters must be in the order **ProfileName** followed by **ObjectType**.

**Optional parameters****AuthorityAdd (MQCFIL)**

Authority values to set (parameter identifier: MQIACF\_AUTH\_ADD\_AUTHS).

This parameter is a list of authority values to set for the named profile. The values can be:

**MQAUTH\_NONE**

The entity has authority set to 'none'.

**MQAUTH\_ALT\_USER\_AUTHORITY**

Specify an alternate user ID on an MQI call.

**MQAUTH\_BROWSE**

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

**MQAUTH\_CHANGE**

Change the attributes of the specified object, using the appropriate command set.

**MQAUTH\_CLEAR**

Clear a queue.

**MQAUTH\_CONNECT**

Connect the application to the specified queue manager by issuing an MQCONN call.

**MQAUTH\_CREATE**

Create objects of the specified type using the appropriate command set.

**MQAUTH\_DELETE**

Delete the specified object using the appropriate command set.

**MQAUTH\_DISPLAY**

Display the attributes of the specified object using the appropriate command set.

**MQAUTH\_INPUT**

Retrieve a message from a queue by issuing an MQGET call.

**MQAUTH\_INQUIRE**

Make an inquiry on a specific queue by issuing an MQINQ call.

**MQAUTH\_OUTPUT**

Put a message on a specific queue by issuing an MQPUT call.

**MQAUTH\_PASS\_ALL\_CONTEXT**

Pass all context.

**MQAUTH\_PASS\_IDENTITY\_CONTEXT**

Pass the identity context.

**MQAUTH\_SET**

Set attributes on a queue from the MQI by issuing an MQSET call.

**MQAUTH\_SET\_ALL\_CONTEXT**

Set all context on a queue.

**MQAUTH\_SET\_IDENTITY\_CONTEXT**

Set the identity context on a queue.

**MQAUTH\_CONTROL**

For listeners and services, start and stop the specified channel, listener, or service.

For channels, start, stop, and ping the specified channel.

For topics, define, alter, or delete subscriptions.

**MQAUTH\_CONTROL\_EXTENDED**

Reset or resolve the specified channel.

**MQAUTH\_PUBLISH**

Publish to the specified topic.

**MQAUTH\_SUBSCRIBE**

Subscribe to the specified topic.

**MQAUTH\_RESUME**

Resume a subscription to the specified topic.

**MQAUTH\_SYSTEM**

Use queue manager for internal system operations.

**MQAUTH\_ALL**

Use all operations applicable to the object.

**MQAUTH\_ALL\_ADMIN**

Use all administration operations applicable to the object.

**MQAUTH\_ALL\_MQI**

Use all MQI calls applicable to the object.

The contents of the *AuthorityAdd* and *AuthorityRemove* lists must be mutually exclusive. You must specify a value for either *AuthorityAdd* or *AuthorityRemove*. An error occurs if you do not specify either.

**AuthorityRemove (MQCFIL)**

Authority values to remove (parameter identifier: MQIACF\_AUTH\_REMOVE\_AUTHS).

This parameter is a list of authority values to remove from the named profile. The values can be:

**MQAUTH\_NONE**

The entity has authority set to 'none'.

**MQAUTH\_ALT\_USER\_AUTHORITY**

Specify an alternate user ID on an MQI call.

**MQAUTH\_BROWSE**

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

**MQAUTH\_CHANGE**

Change the attributes of the specified object, using the appropriate command set.

**MQAUTH\_CLEAR**

Clear a queue.

**MQAUTH\_CONNECT**

Connect the application to the specified queue manager by issuing an MQCONN call.

**MQAUTH\_CREATE**

Create objects of the specified type using the appropriate command set.

**MQAUTH\_DELETE**

Delete the specified object using the appropriate command set.

**MQAUTH\_DISPLAY**

Display the attributes of the specified object using the appropriate command set.

**MQAUTH\_INPUT**

Retrieve a message from a queue by issuing an MQGET call.

**MQAUTH\_INQUIRE**

Make an inquiry on a specific queue by issuing an MQINQ call.

**MQAUTH\_OUTPUT**

Put a message on a specific queue by issuing an MQPUT call.

**MQAUTH\_PASS\_ALL\_CONTEXT**

Pass all context.

**MQAUTH\_PASS\_IDENTITY\_CONTEXT**

Pass the identity context.

**MQAUTH\_SET**

Set attributes on a queue from the MQI by issuing an MQSET call.

**MQAUTH\_SET\_ALL\_CONTEXT**

Set all context on a queue.

**MQAUTH\_SET\_IDENTITY\_CONTEXT**

Set the identity context on a queue.

**MQAUTH\_CONTROL**

For listeners and services, start and stop the specified channel, listener, or service.

For channels, start, stop, and ping the specified channel.

For topics, define, alter, or delete subscriptions.

**MQAUTH\_CONTROL\_EXTENDED**

Reset or resolve the specified channel.

**MQAUTH\_PUBLISH**

Publish to the specified topic.

**MQAUTH\_SUBSCRIBE**

Subscribe to the specified topic.

**MQAUTH\_RESUME**

Resume a subscription to the specified topic.

**MQAUTH\_SYSTEM**

Use queue manager for internal system operations.

**MQAUTH\_ALL**

Use all operations applicable to the object.

**MQAUTH\_ALL\_ADMIN**

Use all administration operations applicable to the object.

**MQAUTH\_ALL\_MQI**

Use all MQI calls applicable to the object.

The contents of the *AuthorityAdd* and *AuthorityRemove* lists must be mutually exclusive. You must specify a value for either *AuthorityAdd* or *AuthorityRemove*. An error occurs if you do not specify either.

**GroupNames (MQCFSL)**

Group names (parameter identifier: MQCACF\_GROUP\_ENTITY\_NAMES).

The names of groups having their authorizations set. At least one group name or principal name must be specified. An error occurs if neither are specified.

Each member in this list can be a maximum length of MQ\_ENTITY\_NAME\_LENGTH.

### **PrincipalNames (MQCFSL)**

Principal names (parameter identifier: MQCACF\_PRINCIPAL\_ENTITY\_NAMES).

The names of principals having their authorizations set. At least one group name or principal name must be specified. An error occurs if neither are specified.

Each member in this list can be a maximum length of MQ\_ENTITY\_NAME\_LENGTH.

### **ServiceComponent (MQCFST)**

Service component (parameter identifier: MQCACF\_SERVICE\_COMPONENT).

If installable authorization services are supported, this parameter specifies the name of the authorization service to which the authorizations apply.

If you omit this parameter, the authorization inquiry is made to the first installable component for the service.

The maximum length of the string is MQ\_SERVICE\_COMPONENT\_LENGTH.

### **Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

#### **Reason (MQLONG)**

The value can be any of the following values:

##### **MQRC\_UNKNOWN\_ENTITY**

Userid not authorized, or unknown.

##### **MQRCCF\_AUTH\_VALUE\_ERROR**

Invalid authorization.

##### **MQRCCF\_AUTH\_VALUE\_MISSING**

Authorization missing.

##### **MQRCCF\_ENTITY\_NAME\_MISSING**

Entity name missing.

##### **MQRCCF\_OBJECT\_TYPE\_MISSING**

Object type missing.

##### **MQRCCF\_PROFILE\_NAME\_ERROR**

Invalid profile name.

### ***MQCMD\_SET\_CHLAUTH\_REC (Set Channel Authentication Record)***

The Set Channel Authentication Record (MQCMD\_SET\_CHLAUTH\_REC) PCF command sets the allowed partner details and mappings to MCAUSER for a channel or set of channels.

### **Syntax diagram**

See the syntax diagram in the MQSC [“SET CHLAUTH \(create or modify a channel authentication record\)” on page 926](#) command for combinations of parameters and values that are allowed.

### **Required parameters**

The required parameters are valid for the **Action** values of:

- MQACT\_ADD or MQACT\_REPLACE
- MQACT\_REMOVE
- MQACT\_REMOVEALL

#### **ProfileName (MQCFST)**

The name of the channel or set of channels for which you are setting channel authentication configuration (parameter identifier: MQCACH\_CHANNEL\_NAME). You can use one or more asterisks

(\*), in any position, as wildcards to specify a set of channels. If you set Type to MQCAUT\_BLOCKADDR, you must set the generic channel name to a single asterisk, which matches all channel names.

The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

### Type (MQCFIN)

The **Type** parameter must follow the **ProfileName** parameter.

The type of channel authentication record for which to set allowed partner details or mappings to MCAUSER (parameter identifier: MQIACF\_CHLAUTH\_TYPE). The following values are valid:

#### MQCAUT\_BLOCKUSER

This channel authentication record prevents a specified user or users from connecting. The MQCAUT\_BLOCKUSER parameter must be accompanied by a **UserList**.

#### MQCAUT\_BLOCKADDR

This channel authentication record prevents connections from a specified IP address or addresses. The MQCAUT\_BLOCKADDR parameter must be accompanied by an **AddrList**.

#### MQCAUT\_SSLPEERMAP

This channel authentication record maps TLS Distinguished Names (DNs) to MCAUSER values. The MQCAUT\_SSLPEERMAP parameter must be accompanied by an **SSLPeer**.

#### MQCAUT\_ADDRESSMAP

This channel authentication record maps IP addresses to MCAUSER values. The MQCAUT\_ADDRESSMAP parameter must be accompanied by an **Address**.

#### MQCAUT\_USERMAP

This channel authentication record maps asserted user IDs to MCAUSER values. The MQCAUT\_USERMAP parameter must be accompanied by a **ClntUser**.

#### MQCAUT\_QMGRMAP

This channel authentication record maps remote queue manager names to MCAUSER values. The MQCAUT\_QMGRMAP parameter must be accompanied by a **QMName**.

## Optional parameters

The following table shows which parameters are valid for each value of **Action**:

Parameter	MQACT_ADD or MQACT_REPLACE	MQACT_REMOVE	MQACT_REMOVEALL
z/OS	✓	✓	✓
z/OS	✓	✓	✓
CommandScope			
Action	✓	✓	✓
Address	✓	✓	
Addrlist	✓	✓	
CheckClient	✓	✓	
ClntUser	✓	✓	
MCAUser	✓		
QMName	✓	✓	
SSLCertIssuer	✓	✓	

Table 214. Optional parameters for ChannelAttrs (continued)

Parameter	MQACT_ADD or MQACT_REPLACE	MQACT_REMOVE	MQACT_REMOVEALL
SSLPeer	✓	✓	
UserList	✓	✓	
UserSrc	✓		
Warn	✓		
Description	✓		

### Action (MQCFIN)

The action to perform on the channel authentication record (parameter identifier: MQIACF\_ACTION). The following values are valid:

#### MQACT\_ADD

Add the specified configuration to a channel authentication record. This is the default value.

For types MQCAUT\_SSLPEERMAP, MQCAUT\_ADDRESSMAP, MQCAUT\_USERMAP and MQCAUT\_QMGRMAP, if the specified configuration exists, the command fails.

For types MQCAUT\_BLOCKUSER and MQCAUT\_BLOCKADDR, the configuration is added to the list.

#### MQACT\_REPLACE

Replace the current configuration of a channel authentication record.

For types MQCAUT\_SSLPEERMAP, MQCAUT\_ADDRESSMAP, MQCAUT\_USERMAP and MQCAUT\_QMGRMAP, if the specified configuration exists, it is replaced with the new configuration. If it does not exist it is added.

For types MQCAUT\_BLOCKUSER and MQCAUT\_BLOCKADDR, the configuration specified replaces the current list, even if the current list is empty. If you replace the current list with an empty list, this acts like MQACT\_REMOVEALL.

#### MQACT\_REMOVE

Remove the specified configuration from the channel authentication records. If the configuration does not exist the command fails. If you remove the last entry from a list, this acts like MQACT\_REMOVEALL.

#### MQACT\_REMOVEALL

Remove all members of the list and thus the whole record (for MQCAUT\_BLOCKADDR and MQCAUT\_BLOCKUSER) or all previously defined mappings (for MQCAUT\_ADDRESSMAP, MQCAUT\_SSLPEERMAP, MQCAUT\_QMGRMAP and MQCAUT\_USERMAP) from the channel authentication records. This option cannot be combined with specific values supplied in **AddrList**, **UserList**, **Address**, **SSLPeer**, **QMName** or **CIntUser**. If the specified type has no current configuration the command still succeeds.

### Address (MQCFST)

The filter to be used to compare with the IP address, or host name, of the partner queue manager or client at the other end of the channel (parameter identifier: MQCACH\_CONNECTION\_NAME).

This parameter is mandatory when **Type** is MQCAUT\_ADDRESSMAP and is also valid when **Type** is MQCAUT\_SSLPEERMAP, MQCAUT\_USERMAP, or MQCAUT\_QMGRMAP and **Action** is MQACT\_ADD, MQACT\_REPLACE, or MQACT\_REMOVE. You can define more than one channel authentication object with the same main identity, for example the same TLS peer name, with different addresses. See [“Generic IP addresses for channel authentication records”](#) on page 933 for more information about filtering IP addresses.

The maximum length of the string is MQ\_CONN\_NAME\_LENGTH.

### **AddrList (MQCFSL)**

A list of up to 100 generic IP addresses which are banned from accessing this queue manager on any channel (parameter identifier: MQCACH\_CONNECTION\_NAME\_LIST).

This parameter is only valid when **Type** is MQCAUT\_BLOCKADDR.

The maximum length of each address is MQ\_CONN\_NAME\_LENGTH.

### **CheckClient (MQCFIN)**

The user ID and password requirements for the client connection to be successful. The following values are valid:

#### **MQCHK\_REQUIRED\_ADMIN**

A valid user ID and password are required for the connection to be allowed if you are using a privileged user ID. The password cannot contain single quotation marks ( ' ).

Any connections using a non-privileged user ID are not required to provide a user ID and password.

The user ID and password are checked against the user repository details provided in an authentication information object, and supplied on ALTER QMGR in the CONNAUTH field.

If no user repository details are provided, so that user ID and password checking are not enabled on the queue manager, the connection is not successful.

A privileged user is one that has full administrative authorities for IBM MQ. See [Privileged users](#) for more information.

This option is not valid on z/OS platforms.

#### **MQCHK\_REQUIRED**

A valid user ID and password are required for the connection to be allowed. The password cannot contain single quotation marks ( ' ).

The user ID and password are checked against the user repository details provided in an authentication information object and supplied on ALTER QMGR in the CONNAUTH field.

If no user repository details are provided, so that user ID and password checking are not enabled on the queue manager, the connection is not successful.

#### **MQCHK\_AS\_Q\_MGR**

In order for the connection to be allowed, it must meet the connection authentication requirements defined on the queue manager.

If the CONNAUTH field provides an authentication information object, and the value of CHCKCLNT is REQUIRED, the connection fails unless a valid user ID and password are supplied.

If the CONNAUTH field does not provide an authentication information object, or the value of CHCKCLNT is not REQUIRED, the user ID and password are not required.

### **ClntUser (MQCFST)**

The client asserted user ID to be mapped to a new user ID, allowed through unchanged, or blocked (parameter identifier: MQCACH\_CLIENT\_USER\_ID).

This can be the user ID flowed from the client indicating the user ID the client side process is running under, or the user ID presented by the client on an MQCONN call using MQCSP.

This parameter is valid only with TYPE(USERMAP) and when **Match** is MQMATCH\_RUNCHECK.

The maximum length of the string is MQ\_CLIENT\_USER\_ID\_LENGTH.

### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is run when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is run on the queue manager on which it was entered.
- a queue manager name. The command is run on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which the command was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is run on the local queue manager and is also passed to every active queue manager in the queue sharing group.

#### **Custom (MQCFST)**

Reserved for future use.

#### **Description (MQCFST)**

Provides descriptive information about the channel authentication record, which is displayed when you issue the Inquire Channel Authentication Records command (parameter identifier: MQCA\_CHLAUTH\_DESC).

This parameter must contain only displayable characters. In a DBCS installation, it can contain DBCS characters. The maximum length of the string is MQ\_CHLAUTH\_DESC\_LENGTH.

**Note:** Use characters from the coded character set identifier (CCSID) for this queue manager. Other characters might be translated incorrectly if the information is sent to another queue manager.

#### **MCAUser (MQCFST)**

The user identifier to be used when the inbound connection matches the TLS DN, IP address, client asserted user ID or remote queue manager name supplied (parameter identifier: MQCACH\_MCA\_USER\_ID).

This parameter is mandatory when **UserSrc** is MQUSRC\_MAP and is valid when **Type** is MQCAUT\_SSLPEERMAP, MQCAUT\_ADDRESSMAP, MQCAUT\_USERMAP, or MQCAUT\_QMGRMAP.

This parameter is valid only when **Action** is MQACT\_ADD or MQACT\_REPLACE.

The maximum length of the string is MQ\_MCA\_USER\_ID\_LENGTH.

#### **QMName (MQCFST)**

The name of the remote partner queue manager, or pattern that matches a set of queue manager names, to be mapped to a user ID or blocked (parameter identifier: MQCA\_REMOTE\_Q\_MGR\_NAME).

This parameter is valid only when **Type** is MQCAUT\_QMGRMAP.

The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

#### **SSLCertIssuer (MQCFST)**

This parameter is additional to the **SSLPeer** parameter.

**SSLCertIssuer** restricts matches to being within certificates issued by a particular Certificate Authority.

#### **SSLPeer (MQCFST)**

The filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel (parameter identifier: MQCACH\_SSL\_PEER\_NAME).

The **SSLPeer** value is specified in the standard form used to specify a Distinguished Name. See [Distinguished Names](#) and [IBM MQ rules for SSLPEER values](#).

The maximum length of the string is MQ\_SSL\_PEER\_NAME\_LENGTH.

#### **UserList (MQCFSL)**


A list of up to 100 user IDs which are banned from using this channel or set of channels (parameter identifier: MQCACH\_MCA\_USER\_ID\_LIST).

The following special value can be used:



### **\*MQADMIN**

The exact meaning of this value is determined at runtime. If you are using the OAM supplied with IBM MQ, the meaning depends on platform, as follows:

- On Windows, all members of the mqm group, the Administrators group and SYSTEM
- On AIX and Linux, all members of the mqm group
- On IBM i, the profiles (users) qmqm and qmqmadm and all members of the qmqmadm group, and any user defined with the \*ALLOBJ special setting
-  On z/OS, the user ID that the CHINIT and the user ID that the MSTR address spaces are running under

This parameter is only valid when **TYPE** is MQCAUT\_BLOCKUSER.

The maximum length of each user ID is MQ\_MCA\_USER\_ID\_LENGTH .

### **UserSrc (MQCFIN)**

The source of the user ID to be used for MCAUSER at run time (parameter identifier: MQIACH\_USER\_SOURCE).

The following values are valid:

#### **MQUSRC\_MAP**

Inbound connections that match this mapping use the user ID specified in the **MCAUser** attribute. This is the default value.

#### **MQUSRC\_NOACCESS**

Inbound connections that match this mapping have no access to the queue manager and the channel ends immediately.

#### **MQUSRC\_CHANNEL**

Inbound connections that match this mapping use the flowed user ID or any user defined on the channel object in the MCAUSER field.

Note that *Warn* and MQUSRC\_CHANNEL, or MQUSRC\_MAP are incompatible. This is because channel access is never blocked in these cases, so there is never a reason to generate a warning.

### **Warn (MQCFIN)**

Indicates whether this record operates in warning mode (parameter identifier: MQIACH\_WARNING).

#### **MQWARN\_NO**

This record does not operate in warning mode. Any inbound connection that matches this record is blocked. This is the default value.

#### **MQWARN\_YES**

This record operates in warning mode. Any inbound connection that matches this record and would therefore be blocked is allowed access. An error message is written and, if events are configured, an event message is created showing the details of what would have been blocked. The connection is allowed to continue. An attempt is made to find another record that is set to WARN(NO) to set the credentials for the inbound channel.

### **Error codes**

This command might return the following error codes in the response format header, in addition to the values shown at [“Error codes applicable to all commands” on page 990](#).

### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRCCF\_CHLAUTH\_TYPE\_ERROR**

Channel authentication record type not valid.

#### **MQRCCF\_CHLAUTH\_ACTION\_ERROR**

Channel authentication record action not valid.

**MQRCCF\_CHLAUTH\_USERSRC\_ERROR**

Channel authentication record user source not valid.

**MQRCCF\_WRONG\_CHLAUTH\_TYPE**

Parameter not allowed for this channel authentication record type.

**MQRCCF\_CHLAUTH\_ALREADY\_EXISTS**

Channel authentication record already exists

**Related concepts**

[Channel authentication records](#)

**ALW MQCMD\_SET\_LOG (notify completion of log archiving) on AIX, Linux, and Windows**

The Set Log (MQCMD\_SET\_LOG) PCF command on AIX, Linux, and Windows enables you to notify the queue manager that archiving of a log is complete. If the log management type is not **Archive** the command fails. This command requires change authority on the queue manager object.

**Required parameters:**

*ParameterType*

**Optional parameters:**

*Archive*

**Required parameters****ParameterType (MQCFIN)**

Specifies the type of the log (parameter identifier: MQIACF\_SYSP\_TYPE).

The value must be MQSYSP\_TYPE\_SET

**Optional parameters****Archive (MQCFST)**

Specifies the log extent that is being marked as archived (parameter identifier: MQCACF\_ARCHIVE\_LOG\_EXTENT\_NAME).

The command fails if the log extent is not recognized, or is the current log. The command does not fail if the extent has already been marked as having been archived.

A message is written to the error log if the queue manager is notified about an extent more than once.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown at [“Error codes applicable to all commands”](#) on page 990.

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_LOG\_EXTENT\_NOT\_FOUND**

The specified log extent was not found or is not valid.

**MQRCCF\_CURRENT\_LOG\_EXTENT**

The specified log extent is the current log extent, and cannot have been validly archived yet.

**MQRCCF\_LOG\_TYPE\_ERROR**

The command has been run on a log that is not an archive log.

**MQRCCF\_LOG\_EXTENT\_ERROR**

The specified log extent is corrupt.

## **MQCMD\_SET\_LOG (Set Log) on z/OS**

The Set Log (MQCMD\_SET\_LOG) PCF command dynamically changes certain log system parameter values initially set by your system parameter module at queue manager startup.

### **Required parameters:**

*ParameterType*

### **Optional parameters (if the value of *ParameterType* is MQSYSP\_TYPE\_SET:**

*CommandScope, DeallocateInterval, LogCompression, MaxArchiveLog, MaxConcurrentOffloads, MaxReadTapeUnits, OutputBufferCount, zHyperWrite*

### **Optional parameters if *ParameterType* type is MQSYSP\_TYPE\_INITIAL:**

*CommandScope*

## **Required parameters**

### **ParameterType (MQCFIN)**

Parameter type (parameter identifier: MQIACF\_SYSP\_TYPE).

Specifies how the parameters are to be set:

#### **MQSYSP\_TYPE\_INITIAL**

The initial settings of the log system parameters. This MQSYSP\_TYPE\_INITIAL resets all the log system parameters to the values at queue manager startup.

#### **MQSYSP\_TYPE\_SET**

This MQSYSP\_TYPE\_SET indicates that you intend to change one, or more, of the archive log system parameter settings.

## **Optional parameters**

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is run when the queue manager is a member of a queue sharing group. You can specify one of the following:

- Blank (or omit the parameter altogether). The command is run on the queue manager on which it was entered.
- A queue manager name. The command is run on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- An asterisk (\*). The command is run on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **DeallocateInterval (MQCFIN)**

Deallocation interval (parameter identifier: MQIACF\_SYSP\_DEALLOC\_INTERVAL).

Specifies the length of time, in minutes, that an allocated archive read tape unit is allowed to remain unused before it is deallocated. This parameter, together with the **MaxReadTapeUnits** parameter, allows IBM MQ to optimize archive log reading from tape devices. You are recommended to specify the maximum values, within system constraints, for both parameters, in order to achieve the optimum performance for reading archive tapes.

Specify a value in the range zero and 1440. Zero means that a tape unit is deallocated immediately. If you specify a value of 1440, the tape unit is never deallocated.

### **LogCompression (MQCFIN)**

Log compression parameter (parameter identifier: MQIACF\_LOG\_COMPRESSION).

Specifies the log compression algorithm to enable.

The possible values are:

**MQCOMPRESS\_NONE**


Log compression is disabled.

**MQCOMPRESS\_RLE**

Enable run-length encoding log compression.

**MQCOMPRESS\_ANY**

Enable the queue manager to select the compression algorithm that gives the greatest degree of log record compression.

 For more details see [The log files](#).

**MaxArchiveLog (MQCFIN)**

Specifies the maximum number of archive log volumes that can be recorded in the BSDS (parameter identifier: MQIACF\_SYSP\_MAX\_ARCHIVE).

When this value is exceeded, recording recommences at the start of the BSDS.

Specify a value in the range 10 through 100.

**MaxConcurrentOffloads (MQCFIN)**

Specifies the maximum number of concurrent log offload tasks (parameter identifier: MQIACF\_SYSP\_MAX\_CONC\_OFFLOADS).

Specify a decimal number between 1 and 31. If no value is specified the default of 31 applies.

Configure a number lower than the default if your archive logs are allocated on a tape device, and there are constraints on the number of such devices that can be concurrently allocated to the queue manager.

**MaxReadTapeUnits (MQCFIN)**

Specifies the maximum number of dedicated tape units that can be allocated to read archive log tape volumes (parameter identifier: MQIACF\_SYSP\_MAX\_READ\_TAPES).

This parameter, together with the *DeallocateInterval* parameter, allows IBM MQ to optimize archive log reading from tape devices.

Specify a value in the range 1 through 99.

If you specify a value that is greater than the current specification, the maximum number of tape units allowable for reading archive logs increases. If you specify a value that is less than the current specification, tape units that are not being used are immediately deallocated to adjust to the new value. Active, or premounted, tapes remain allocated.

**OutputBufferCount (MQCFIN)**

Specifies the number of 4 KB output buffers to be filled before they are written to the active log data sets (parameter identifier: MQIACF\_SYSP\_OUT\_BUFFER\_COUNT).

Specify the number of buffers in the range 1 through 256.

The larger the number of buffers and the less often the write takes place improves the performance of IBM MQ. The buffers might be written before this number is reached if significant events, such as a commit point, occur.

**zHyperWrite (MQCFIN)**

Specifies whether writes to the active logs are made with zHyperWrite enabled (parameter identifier: MQIACF\_SYSP\_ZHYPERWRITE).

The active log datasets need to be on zHyperWrite capable volumes for zHyperWrite to be enabled.

For more information on enabling active logs with zHyperWrite, see [Using zHyperWrite with IBM MQ active logs](#).

The possible values are:

**MQSYSP\_NO**

zHyperWrite is not enabled.

## **MQSYSP\_YES**

zHyperWrite is enabled.

## **z/OS MQCMD\_SET\_SYSTEM (Set System) on z/OS**

The Set System (MQCMD\_SET\_SYSTEM) PCF command dynamically changes certain general system parameter values initially set from your system parameter module at queue manager startup.

### **Required parameters:**

*ParameterType*

### **Optional parameters (if the value of *ParameterType* is MQSYSP\_TYPE\_SET:**

**V9.3.0** From IBM MQ for z/OS 9.3.0, *CheckpointCount*, *CommandScope*, *Exclmsg*, *MaxConnects*, *MaxConnectsBackground*, *MaxConnectsForeground*, *Service*, *SMFActIntervalMins*, *SMFActIntervalSecs*, *SMFStatsIntervalMins*, *SMFStatsIntervalSecs*, *TraceSize*

### **Optional parameters if *ParameterType* type is MQSYSP\_INITIAL:**

*CommandScope*

## **Required parameters**

### **ParameterType (MQCFIN)**

Parameter type (parameter identifier: MQIACF\_SYSP\_TYPE).

Specifies how the parameters are to be set:

### **MQSYSP\_TYPE\_INITIAL**

The initial settings of the system parameters. MQSYSP\_TYPE\_INITIAL resets the parameters to the values specified in the system parameters at queue manager startup.

### **MQSYSP\_TYPE\_SET**

MQSYSP\_TYPE\_SET indicates that you intend to change one, or more, of the system parameter settings.

## **Optional parameters**

### **CheckpointCount (MQCFIN)**

The number of log records written by IBM MQ between the start of one checkpoint and the next (parameter identifier: MQIACF\_SYSP\_CHKPOINT\_COUNT).

IBM MQ starts a new checkpoint after the number of records that you specify has been written.

Specify a value in the range 200 through 16 000 000.

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### Exclmsg (MQCFSL)

A list of message identifiers to be excluded from being written to any log (parameter identifier: MQCACF\_EXCL\_OPERATOR\_MESSAGES).

Specify a list of error message identifiers to be excluded from being written to any log. For example, to exclude message CSQX500I, add X500 to this list. Messages in this list are not sent to the z/OS console and hardcopy log. As a result using the EXCLMSG parameter to exclude messages is more efficient from a CPU perspective than using z/OS mechanisms such as the message processing facility list and should be used instead where possible.

The maximum length of each message identifier is MQ\_OPERATOR\_MESSAGE\_LENGTH.

The list can contain a maximum of 16 message identifiers.

### Service (MQCFST)

Service parameter setting (parameter identifier: MQCACF\_SYSP\_SERVICE).

This parameter is reserved for use by IBM.

#### **V 9.3.0** SMFAcctIntervalMins (MQCFIN)

From IBM MQ for z/OS 9.3.0, the minutes value of the default time between each gathering of accounting data (parameter identifier: MQIACF\_SYSP\_SMF\_ACCT\_TIME\_MINS).

Specify a value in the range -1 through 1440.

**Note:** You should also set *SMFAcctIntervalSecs*, otherwise it defaults to 0.

If you specify a value of 0, without specifying a non-zero value for *SMFAcctIntervalSecs*, accounting data is collected at the SMF data collection broadcast.

If you specify a value of -1, accounting data is collected using the statistics interval values.

#### **V 9.3.0** SMFAcctIntervalSecs(MQCFIN)

From IBM MQ for z/OS 9.3.0, the seconds value of the default time between each gathering of accounting data (parameter identifier: MQIACF\_SYSP\_SMF\_ACCT\_TIME\_SECS).

Specify a value in the range 0 through 59.

**Note:** You should also set *SMFAcctIntervalMins*, otherwise it defaults to 0.

If you specify a value of 0, without specifying a non-zero value for *SMFAcctIntervalMins*, accounting data is collected at the SMF data collection broadcast.

#### **V 9.3.0** SMFStatsIntervalMins (MQCFIN)

From IBM MQ for z/OS 9.3.0, the minutes value of the default time between each gathering of statistics data (parameter identifier: MQIACF\_SYSP\_SMF\_STAT\_TIME\_MINS or parameter identifier: MQIACF\_SYSP\_SMF\_INTERVAL).

Specify a value in the range 0 through 1440.

**Note:** You should also set *SMFStatsIntervalSecs*, otherwise it defaults to 0.

If you specify a value of 0, without specifying a non-zero value for *SMFStatsIntervalSecs*, accounting data is collected at the SMF data collection broadcast.

#### **V 9.3.0** SMFStatsIntervalSecs (MQCFIN)

From IBM MQ for z/OS 9.3.0 onwards, the seconds value of the default time between each gathering of accounting data (parameter identifier: MQIACF\_SYSP\_SMF\_ACCT\_TIME\_SECS).

Specify a value in the range 0 through 59.

**Note:** You should also set *SMFStatsIntervalMins*, otherwise it defaults to zero.

If you specify a value of 0, without specifying a non-zero value for *SMFStatsIntervalMins*, statistics data is collected at the SMF data collection broadcast.

### **TraceSize (MQCFIN)**

The size of the trace table, in 4 KB blocks, to be used by the global trace facility (parameter identifier: MQIACF\_SYSP\_TRACE\_SIZE).

Specify a value in the range zero through 999.

### **MQCMD\_START\_CHANNEL (Start Channel)**

The Start Channel (MQCMD\_START\_CHANNEL) PCF command starts an IBM MQ channel. This command can be issued to a channel of any type (except MQCHT\_CLNTCONN). If, however, it is issued to a channel with a *ChannelType* value of MQCHT\_RECEIVER, MQCHT\_SVRCONN, or MQCHT\_CLUSRCVR, the only action is to enable the channel, not start it.

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

None of the following attributes are applicable to MQTT channels unless specifically mentioned in the parameter description.

## **Required parameters**

### **ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The name of the channel to be started. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

This parameter is required for all channel types including MQTT channels.

## **Optional parameters for z/OS**



### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **ChannelDisposition (MQCFIN)**

Channel disposition (parameter identifier: MQIACH\_CHANNEL\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be started.

If this parameter is omitted, then the value for the channel disposition is taken from the default channel disposition attribute of the channel object.

The value can be:

**MQCHLD\_PRIVATE**

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD\_SHARED.

**MQCHLD\_SHARED**

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD\_SHARED.

**MQCHLD\_FIXSHARED**

Shared channels tied to a specific queue manager.

The combination of the **ChannelDisposition** and **CommandScope** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On every active queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in [Table 215 on page 1496](#)

<i>Table 215. ChannelDisposition and CommandScope for START CHANNEL</i>			
<b>ChannelDisposition</b>	<b>CommandScope blank or local-qmgr</b>	<b>CommandScope qmgr-name</b>	<b>CommandScope (*)</b>
MQCHLD_PRIVATE	Start as a private channel on the local queue manager	Start as a private channel on the named queue manager	Start as a private channel on all active queue managers



Table 215. ChannelDisposition and CommandScope for START CHANNEL (continued)

<b>ChannelDisposition</b>	<b>CommandScope blank or local-qmgr</b>	<b>CommandScope qmgr-name</b>	<b>CommandScope (*)</b>
MQCHLD_SHARED	<p>For channels of <i>ChannelType</i> MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, start as a shared channel on the most suitable queue manager in the group.</p> <p>For a shared channel of <i>ChannelType</i> MQCHT_RECEIVER and MQCHT_SVRCONN, start the channel on all active queue managers.</p> <p>For a shared channel of <i>ChannelType</i> MQCHT_CLUSSDR and MQCHT_CLUSRCVR, this option is not permitted.</p> <p>MQCHLD_SHARED might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted	Not permitted
MQCHLD_FIXSHARED	For a shared channel of <i>ChannelType</i> MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, with a nonblank <i>ConnectionName</i> , start as a shared channel on the local queue manager.	For a shared channel of <i>ChannelType</i> MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, with a nonblank <i>ConnectionName</i> , start as a shared channel on the named queue manager.	Not permitted

## Optional parameters for Multiplatforms



### MQIACF\_IGNORE\_STATE

Specifies whether the command fails if the channel is already running. The possible values are:

**MQIS\_NO**

The command fails if the channel is already running. This is the default value.

**MQIS\_YES**

The command succeeds regardless of the current state of the channel.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_CHANNEL\_INDOUBT**

Channel in-doubt.

**MQRCCF\_CHANNEL\_IN\_USE**

Channel in use.

**MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

**MQRCCF\_CHANNEL\_TYPE\_ERROR**

Channel type not valid.

**MQRCCF\_MQCONN\_FAILED**

MQCONN call failed.

**MQRCCF\_MQINQ\_FAILED**

MQINQ call failed.

**MQRCCF\_MQOPEN\_FAILED**

MQOPEN call failed.

**MQRCCF\_NOT\_XMIT\_Q**

Queue is not a transmission queue.

## **MQCMD\_START\_CHANNEL (Start Channel) MQTT on AIX, Linux, and Windows**

The Start Channel (MQCMD\_START\_CHANNEL) PCF command starts an IBM MQ channel. This command can be issued to a channel of type MQCHT\_MQTT.

**Required parameters****ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The name of the channel to be started. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

This parameter is required for all channel types including MQTT channels.

**ChannelType (MQCFIN)**

The type of channel (parameter identifier: MQIACH\_CHANNEL\_TYPE). This parameter is currently only used with MQTT Telemetry channels, and is required when starting a Telemetry channel. The only value that can currently be given to the parameter is MQCHT\_MQTT.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_PARM\_SYNTAX\_ERROR**

The parameter specified contained a syntax error.

**MQRCCF\_PARM\_MISSING**

Parameters are missing.

**MQRCCF\_CHANNEL\_NOT\_FOUND**

The channel specified does not exist.

**MQRCCF\_CHANNEL\_IN\_USE**

The command did not specify a parameter or parameter value that was required.

**MQRCCF\_NO\_STORAGE**

Insufficient storage is available.

**MQRCCF\_COMMAND\_FAILED**

The command has failed.

**MQRCCF\_PORT\_IN\_USE**

The port is in use.

**MQRCCF\_BIND\_FAILED**

The bind to a remote system during session negotiation has failed.

**MQRCCF\_SOCKET\_ERROR**

Socket error has occurred.

**MQRCCF\_HOST\_NOT\_AVAILABLE**

An attempt to allocate a conversation to a remote system was unsuccessful. The error might be transitory, and the allocate might succeed later. This reason can occur if the listening program at the remote system is not running.

***MQCMD\_START\_CHANNEL\_INIT (Start Channel Initiator)***

The Start Channel Initiator (MQCMD\_START\_CHANNEL\_INIT) PCF command starts an IBM MQ channel initiator.

**Required parameters****InitiationQName (MQCFST)**

Initiation queue name (parameter identifier: MQCA\_INITIATION\_Q\_NAME).

The name of the initiation queue for the channel initiation process. That is, the initiation queue that is specified in the definition of the transmission queue.

This parameter is not valid on z/OS.

The maximum length of the string is MQ\_Q\_NAME\_LENGTH.

**Optional parameters****CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

## EnvironmentInfo (MQCFST)

Environment information (parameter identifier: MQCACF\_ENV\_INFO).

The parameters and values to be substituted in the JCL procedure (xxxxCHIN, where xxxx is the queue manager name) that is used to start the channel initiator address space. This parameter applies to z/OS only.

The maximum length of the string is MQ\_ENV\_INFO\_LENGTH.

## Error codes

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRCCF\_MQCONN\_FAILED**

MQCONN call failed.

#### **MQRCCF\_MQGET\_FAILED**

MQGET call failed.

#### **MQRCCF\_MQOPEN\_FAILED**

MQOPEN call failed.

## ***MQCMD\_START\_CHANNEL\_LISTENER (Start Channel Listener)***

The Start Channel Listener (MQCMD\_START\_CHANNEL\_LISTENER) PCF command starts an IBM MQ listener. On z/OS, this command is valid for any transmission protocol; on other platforms, it is valid only for TCP transmission protocols.

## Optional parameters

### **z/OS** CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_Q\_MGR\_NAME\_LENGTH.

### **z/OS** InboundDisposition (MQCFIN)

Inbound transmission disposition (parameter identifier: MQIACH\_INBOUND\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the inbound transmissions that are to be handled. The value can be any of the following values:

#### **MQINBD\_Q\_MGR**

Listen for transmissions directed to the queue manager. MQINBD\_Q\_MGR is the default.

#### **MQINBD\_GROUP**

Listen for transmissions directed to the queue sharing group. MQINBD\_GROUP is permitted only if there is a shared queue manager environment.

**z/OS** **IPAddress (MQCFST)**

IP address (parameter identifier: MQCACH\_IP\_ADDRESS). This parameter applies to z/OS only.

The IP address for TCP/IP specified in IPv4 dotted decimal, IPv6 hexadecimal, or alphanumeric form. This parameter is valid only for channels that have a *TransportType* of MQXPT\_TCP.

The maximum length of the string is MQ\_IP\_ADDRESS\_LENGTH.

**ListenerName (MQCFST)**

Listener name (parameter identifier: MQCACH\_LISTENER\_NAME). This parameter does not apply to z/OS.

The name of the listener definition to be started. On those platforms on which this parameter is valid, if this parameter is not specified, the default listener SYSTEM.DEFAULT.LISTENER is assumed. If this parameter is specified, no other parameters can be specified.

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

**z/OS** **LUName (MQCFST)**

LU name (parameter identifier: MQCACH\_LU\_NAME). This parameter applies to z/OS only.

The symbolic destination name for the logical unit (LU) as specified in the APPC side information data set. The LU must be the same LU that is specified in the channel initiator parameters to be used for outbound transmissions. This parameter is valid only for channels with a *TransportType* of MQXPT\_LU62.

The maximum length of the string is MQ\_LU\_NAME\_LENGTH.

**z/OS** **Port (MQCFIN)**

Port number for TCP (parameter identifier: MQIACH\_PORT\_NUMBER). This parameter applies to z/OS only.

The port number for TCP. This parameter is valid only for channels with a *TransportType* of MQXPT\_TCP.

**z/OS** **TransportType (MQCFIN)**

Transmission protocol type (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

The value can be:

**MQXPT\_LU62**

LU 6.2.

**MQXPT\_TCP**

TCP.

**MQXPT\_NETBIOS**

NetBIOS.

**MQXPT\_SPX**

SPX.

**Multi** **MQIACF\_IGNORE\_STATE**

Specifies whether the command fails if the listener is already running. The possible values are:

**MQIS\_NO**

The command fails if the listener is already running. This is the default value.

**MQIS\_YES**

The command succeeds regardless of the current state of the listener.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRCCF\_COMMS\_LIBRARY\_ERROR**

Communications protocol library error.

#### **MQRCCF\_LISTENER\_NOT\_STARTED**

Listener not started.

#### **MQRCCF\_LISTENER\_RUNNING**

Listener already running.

#### **MQRCCF\_NETBIOS\_NAME\_ERROR**

NetBIOS listener name error.

### **MQCMD\_START\_SERVICE (Start Service) on Multiplatforms**

The Start Service (MQCMD\_START\_SERVICE) PCF command starts an existing IBM MQ service definition.

### Required parameters

#### **ServiceName (MQCFST)**

Service name (parameter identifier: MQCA\_SERVICE\_NAME).

This parameter is the name of the service definition to be started. The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

### Optional parameters

#### **MQIACF\_IGNORE\_STATE**

Specifies whether the command fails if the service is already running. The possible values are:

#### **MQIS\_NO**

The command fails if the service is already running. This is the default value.

#### **MQIS\_YES**

The command succeeds regardless of the current state of the service.

### Error codes

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

#### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRCCF\_NO\_START\_CMD**

The **StartCommand** parameter of the service is blank.

#### **MQRCCF\_SERVICE\_RUNNING**

Service is already running.

### **MQCMD\_START\_SMDSCONN (Start SMDS Connection) on z/OS**

Use the Start SMDS Connection (MQCMD\_START\_SMDSCONN) PCF command after connections have been put into the AVAIL(STOPPED) state by a previous MQCMD\_STOP\_SMDSCONN command. It can also be used to signal to the queue manager to retry a connection which is in the AVAIL(ERROR) state after a previous error.

### Required parameters

#### **SMDSConn (MQCFST)**

Specifies the queue manager name relating to the connection between the shared message data set and the queue manager (parameter identifier: MQCACF\_CF\_SMDSCONN).

An asterisk value can be used to denote all shared message data sets associated with a specific CFSTRUCT name.

The maximum length of the string is 4 characters.

### **CFStrucName (MQCFST)**

The name of the CF application structure with SMDS connections properties that you want to start (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

### **CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### ***MQCMD\_STOP\_CHANNEL (Stop Channel)***

The Stop Channel (MQCMD\_STOP\_CHANNEL) PCF command stops an IBM MQ channel.

This command can be issued to a channel of any type (except MQCHT\_CLNTCONN).

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

None of the following attributes are applicable to MQTT channels unless specifically mentioned in the parameter description.

## **Required parameters**

### **ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

The name of the channel to be stopped. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

This parameter is required for all channel types..

## **Optional parameters**

### **ChannelDisposition (MQCFIN)**

Channel disposition (parameter identifier: MQIACH\_CHANNEL\_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be stopped.

If this parameter is omitted, then the value for the channel disposition is taken from the default channel disposition attribute of the channel object.

The value can be any of the following values:

**MQCHLD\_PRIVATE**

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD\_SHARED.

**MQCHLD\_SHARED**

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD\_SHARED.

The combination of the **ChannelDisposition** and **CommandScope** parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On every active queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in [Table 216 on page 1504](#)

<i>Table 216. ChannelDisposition and CommandScope for STOP CHANNEL</i>			
<b>ChannelDisposition</b>	<b>CommandScope blank or local-qmgr</b>	<b>CommandScope qmgr-name</b>	<b>CommandScope (*)</b>
MQCHLD_PRIVATE	Stop as a private channel on the local queue manager	Stop as a private channel on the named queue manager	Stop as a private channel on all active queue managers



Table 216. ChannelDisposition and CommandScope for STOP CHANNEL (continued)

<b>ChannelDisposition</b>	<b>CommandScope blank or local-qmgr</b>	<b>CommandScope qmgr-name</b>	<b>CommandScope (*)</b>
MQCHLD_SHARED	<p>For channels of <i>ChannelType</i> MQCHT_RECEIVER or MQCHT_SVRCONN, stop as shared channel on all active queue managers.</p> <p>For channels of <i>ChannelType</i> MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, stop as a shared channel on the queue manager where it is running. If the channel is in an inactive state (not running), or if it is in RETRY state because the channel initiator on which it was running has stopped, a STOP request for the channel is issued on the local queue manager.</p> <p>MQCHLD_SHARED might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted	Not permitted

**ChannelStatus (MQCFIN)**

The new state of the channel after the command is executed (parameter identifier: MQIACH\_CHANNEL\_STATUS).

The value can be any of the following values:

**MQCHS\_INACTIVE**

Channel is inactive.

**MQCHS\_STOPPED**

Channel is stopped. MQCHS\_STOPPED is the default if nothing is specified.

 **CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### ConnectionName (MQCFST)

Connection name of channel to be stopped (parameter identifier: MQCACH\_CONNECTION\_NAME).

This parameter is the connection name of the channel to be stopped. If this parameter is omitted, all channels with the specified channel name and remote queue manager name are stopped.

On Multiplatforms, the maximum length of the string is MQ\_CONN\_NAME\_LENGTH. On z/OS, the maximum length of the string is MQ\_LOCAL\_ADDRESS\_LENGTH.

If this parameter is specified, ChannelStatus must be MQCHS\_INACTIVE.

### Mode (MQCFIN)

How the channel must be stopped (parameter identifier: MQIACF\_MODE).

The value can be:

#### MQMODE\_QUIESCE

Quiesce the channel. MQMODE\_QUIESCE is the default.

If you issue a `Stop Channel channelname Mode(MQMODE_QUIESCE)` command on a server-connection channel with the sharing conversations feature enabled, the IBM MQ client infrastructure becomes aware of the stop request in a timely manner; this time is dependent upon the speed of the network. The client application becomes aware of the stop request as a result of issuing a subsequent call to IBM MQ.

#### MQMODE\_FORCE

Stop the channel immediately; the thread or process of the channel is not terminated. Stops transmission of any current batch.

For server-connection channels, breaks the current connection, returning MQRC\_CONNECTION\_BROKEN.

For other types of channels, this situation is likely to result in in-doubt situations.

**z/OS** On z/OS, this option interrupts any message reallocation in progress, which can leave BIND\_NOT\_FIXED messages partially reallocated or out of order.

#### MQMODE\_TERMINATE

**Multi** On Multiplatforms, stop the channel immediately; the thread or process of the channel is terminated.

**z/OS** On z/OS, MQMODE\_TERMINATE is synonymous with FORCE.

**z/OS** On z/OS, this option interrupts any message reallocation in progress, which can leave BIND\_NOT\_FIXED messages partially reallocated or out of order.

**Note:** This parameter was previously called *Quiesce* (MQIACF\_QUIESCE), with values MQQO\_YES and MQQO\_NO. The old names can still be used.

### QMgrName (MQCFST)

Name of remote queue manager (parameter identifier: MQCA\_Q\_MGR\_NAME).

This parameter is the name of the remote queue manager to which the channel is connected. If this parameter is omitted, all channels with the specified channel name and connection name are stopped. The maximum length of the string is MQ\_Q\_MGR\_NAME\_LENGTH.

If this parameter is specified, ChannelStatus must be MQCHS\_INACTIVE.

#### **Multi** MQIACF\_IGNORE\_STATE

Specifies whether the command fails if the channel is already stopped. The possible values are:

##### **MQIS\_NO**

The command fails if the channel is already stopped. This is the default value.

##### **MQIS\_YES**

The command succeeds regardless of the current state of the channel.

## **Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

### **Reason (MQLONG)**

The value can be any of the following values:

#### **MQRCCF\_CHANNEL\_DISABLED**

Channel disabled.

#### **MQRCCF\_CHANNEL\_NOT\_ACTIVE**

Channel not active.

#### **MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

#### **MQRCCF\_MODE\_VALUE\_ERROR**

Mode value not valid.

#### **MQRCCF\_MQCONN\_FAILED**

MQCONN call failed.

#### **MQRCCF\_MQOPEN\_FAILED**

MQOPEN call failed.

#### **MQRCCF\_MQSET\_FAILED**

MQSET call failed.

## **ALW** MQCMD\_STOP\_CHANNEL (Stop Channel) MQTT on AIX, Linux, and Windows

The Stop Channel (MQCMD\_STOP\_CHANNEL) PCF command stops an MQ Telemetry channel.

## **Required parameters**

### **ChannelName (MQCFST)**

Channel name (parameter identifier: MQCACH\_CHANNEL\_NAME).

This parameter is required.

The name of the channel to be stopped. The maximum length of the string is MQ\_CHANNEL\_NAME\_LENGTH.

### **ChannelType (MQCFIN)**

The type of channel (parameter identifier: MQIACH\_CHANNEL\_TYPE). This parameter is currently only used with MQTT Telemetry channels, and is required when stopping a Telemetry channel. The only value that can currently be given to the parameter is **MQCHT\_MQTT**.

## Optional parameters

### ClientIdentifier (MQCFST)

Client identifier. The client identifier is a 23-byte string that identifies an MQ Telemetry Transport client. When the Stop Channel command specifies a *ClientIdentifier*, only the connection for the specified client identifier is stopped. If the CLIENTID is not specified, all the connections on the channel are stopped.

## Error codes

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

### Reason (MQLONG)

The value can be any of the following values:

#### **MQRCCF\_CHANNEL\_DISABLED**

Channel disabled.

#### **MQRCCF\_CHANNEL\_NOT\_ACTIVE**

Channel not active.

#### **MQRCCF\_CHANNEL\_NOT\_FOUND**

Channel not found.

#### **MQRCCF\_MODE\_VALUE\_ERROR**

Mode value not valid.

#### **MQRCCF\_MQCONN\_FAILED**

MQCONN call failed.

#### **MQRCCF\_MQOPEN\_FAILED**

MQOPEN call failed.

#### **MQRCCF\_MQSET\_FAILED**

MQSET call failed.

## **MQCMD\_STOP\_CHANNEL\_INIT (Stop Channel Initiator) on z/OS**

The Stop Channel Initiator (MQCMD\_STOP\_CHANNEL\_INIT) PCF command stops an IBM MQ channel initiator.

## Optional parameters

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### SharedChannelRestart (MQCFIN)

Shared channel restart (parameter identifier: MQIACH\_SHARED\_CHL\_RESTART).

Specifies whether the channel initiator attempts to restart any active sending channels, started with the **ChannelDisposition** parameter set to MQCHLD\_SHARED, that it owns on another queue manager. The value can be:

**MQCHSH\_RESTART\_YES**

Shared sending channels are to be restarted. MQCHSH\_RESTART\_YES is the default.

**MQCHSH\_RESTART\_NO**

Shared sending channels are not to be restarted, so become inactive.

Active channels started with the **ChannelDisposition** parameter set to MQCHLD\_FIXSHARED are not restarted, and always become inactive.

**MQCMD\_STOP\_CHANNEL\_LISTENER (Stop Channel Listener)**

The Stop Channel Listener (MQCMD\_STOP\_CHANNEL\_LISTENER) PCF command stops an IBM MQ listener.

**Required parameters**

**ListenerName (MQCFST)**

Listener name (parameter identifier: MQCACH\_LISTENER\_NAME). This parameter does not apply to z/OS.

The name of the listener definition to be stopped. If this parameter is specified, no other parameters can be specified.

The maximum length of the string is MQ\_LISTENER\_NAME\_LENGTH.

**Optional parameters for z/OS**



**CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

This parameter is valid only on z/OS.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

**InboundDisposition (MQCFIN)**

Inbound transmission disposition (parameter identifier: MQIACH\_INBOUND\_DISP).

Specifies the disposition of the inbound transmissions that the listener handles. The value can be any of the following values:

**MQINBD\_Q\_MGR**

Handling for transmissions directed to the queue manager. MQINBD\_Q\_MGR is the default.

**MQINBD\_GROUP**

Handling for transmissions directed to the queue sharing group. MQINBD\_GROUP is permitted only if there is a shared queue manager environment.

This parameter is valid only on z/OS.

**IPAddress (MQCFST)**

IP address (parameter identifier: MQCACH\_IP\_ADDRESS).

The IP address for TCP/IP specified in dotted decimal or alphanumeric form. This parameter is valid on z/OS only where channels have a *TransportType* of MQXPT\_TCP.

The maximum length of the string is MQ\_IP\_ADDRESS\_LENGTH.

#### **Port (MQCFIN)**

Port number for TCP (parameter identifier: MQIACH\_PORT\_NUMBER).

The port number for TCP. This parameter is valid only on z/OS where channels have a *TransportType* of MQXPT\_TCP.

#### **TransportType (MQCFIN)**

Transmission protocol type (parameter identifier: MQIACH\_XMIT\_PROTOCOL\_TYPE).

The value can be:

##### **MQXPT\_LU62**

LU 6.2.

##### **MQXPT\_TCP**

TCP.

This parameter is valid only on z/OS.

### **Optional parameters for Multiplatforms**

Multi

#### **MQIACF\_IGNORE\_STATE**

Specifies whether the command fails if the listener is already stopped. The possible values are:

##### **MQIS\_NO**

The command fails if the listener is already stopped. This is the default value.

##### **MQIS\_YES**

The command succeeds regardless of the current state of the listener.

### **Error codes**

This command might return the following error code in the response format header, in addition to the values shown in [“Error codes applicable to all commands” on page 990](#).

#### **Reason (MQLONG)**

The value can be any of the following values:

##### **MQRCCF\_LISTENER\_STOPPED**

Listener not running.

Multi

#### **MQCMD\_STOP\_CONNECTION (Stop Connection) on Multiplatforms**

The Stop Connection (MQCMD\_STOP\_CONNECTION) PCF command attempts to break a connection between an application and the queue manager. There might be circumstances in which the queue manager cannot implement this command.

### **Required parameters**

#### **ConnectionId (MQCFBS)**

Connection identifier (parameter identifier: MQBACF\_CONNECTION\_ID).

This parameter is the unique connection identifier associated with an application that is connected to the queue manager.

The length of the byte string is MQ\_CONNECTION\_ID\_LENGTH.

**MQCMD\_STOP\_SERVICE (Stop Service) on Multiplatforms**

The Stop Service (MQCMD\_STOP\_SERVICE) PCF command stops an existing IBM MQ service definition that is running.

**Required parameters****ServiceName (MQCFST)**

Service name (parameter identifier: MQCA\_SERVICE\_NAME).

This parameter is the name of the service definition to be stopped. The maximum length of the string is MQ\_OBJECT\_NAME\_LENGTH.

**Optional parameters****MQIACF\_IGNORE\_STATE**

Specifies whether the command fails if the service is already stopped. The possible values are:

**MQIS\_NO**

The command fails if the service is already stopped. This is the default value.

**MQIS\_YES**

The command succeeds regardless of the current state of the service.

**Error codes**

This command might return the following error codes in the response format header, in addition to the values shown on page [“Error codes applicable to all commands”](#) on page 990.

**Reason (MQLONG)**

The value can be any of the following values:

**MQRCCF\_NO\_STOP\_CMD**

The **StopCommand** parameter of the service is blank.

**MQRCCF\_SERVICE\_STOPPED**

Service is not running.

**MQCMD\_STOP\_SMDSCONN (stop shared message data sets connection) on****z/OS**

The Stop SMDS Connection (MQCMD\_STOP\_SMDSCONN) PCF command terminates the connection from this queue manager to one or more specified shared message data sets. This causes the data sets to be closed and deallocated, and marks the connection as STOPPED.

**Required parameters****SMDSConn (MQCFST)**

Specifies the queue manager name relating to the connection between the shared message data set and the queue manager (parameter identifier: MQCACF\_CF\_SMDSCONN).

An asterisk value can be used to denote all shared message data sets associated with a specific CFSTRUCT name.

The maximum length of the string is 4 characters.

**CFStrucName (MQCFST)**

The name of the CF application structure with SMDS connections properties that you want to stop (parameter identifier: MQCA\_CF\_STRUC\_NAME).

The maximum length of the string is MQ\_CF\_STRUC\_NAME\_LENGTH.

**CommandScope (MQCFST)**

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.
- an asterisk (\*). The command is processed on the local queue manager and is also passed to every active queue manager in the queue sharing group.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

### **MQCMD\_SUSPEND\_Q\_MGR (Suspend Queue Manager) on z/OS**

The Suspend Queue Manager (MQCMD\_SUSPEND\_Q\_MGR) PCF command renders the local queue manager unavailable for the processing of IMS or Db2 messages. Its action can be reversed by the Resume Queue Manager command (MQCMD\_RESUME\_Q\_MGR) command.

## Required parameters

### Facility (MQCFIN)

Facility (parameter identifier: MQIACF\_Q\_MGR\_FACILITY).

The type of facility for which activity is to be suspended. The value can be:

#### **MQQMFACT\_DB2**

The existing connection to Db2 is terminated.

Any in-flight or subsequent MQGET or MQPUT requests are suspended and applications wait until the Db2 connection is re-established by the Resume Queue Manager command, or if the queue manager is stopped.

#### **MQQMFACT\_IMS\_BRIDGE**

Resumes normal IMS bridge activity.

Stops the sending of messages from IMS bridge queues to OTMA. No further messages are sent to IMS until one of these events occurs:

- OTMA is stopped and restarted
- IMS or IBM MQ is stopped or restarted
- A Resume Queue Manager command is processed

Messages returning from IMS OTMA to the queue manager are unaffected.

## Optional parameters

### CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF\_COMMAND\_SCOPE).

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.



## ***MQCMD\_SUSPEND\_Q\_MGR\_CLUSTER (Suspend Queue Manager Cluster)***

The Suspend Queue Manager Cluster (MQCMD\_SUSPEND\_Q\_MGR\_CLUSTER) PCF command informs other queue managers in a cluster that the local queue manager is not available for processing, and cannot be sent messages. Its action can be reversed by the Resume Queue Manager Cluster (MQCMD\_RESUME\_Q\_MGR\_CLUSTER) command.

### **Required parameters**

#### **ClusterName (MQCFST)**

Cluster name (parameter identifier: MQCA\_CLUSTER\_NAME).

The name of the cluster for which availability is to be suspended.

The maximum length of the string is MQ\_CLUSTER\_NAME\_LENGTH.

#### **ClusterNamelist (MQCFST)**

Cluster Namelist (parameter identifier: MQCA\_CLUSTER\_NAMELIST).

The name of the namelist specifying a list of clusters for which availability is to be suspended.

### **Optional parameters**

#### **z/OS CommandScope (MQCFST)**

Command scope (parameter identifier: MQACF\_COMMAND\_SCOPE). This parameter applies to z/OS only.

Specifies how the command is processed when the queue manager is a member of a queue sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is processed on the queue manager on which it was entered.
- a queue manager name. The command is processed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue sharing group environment, and the command server must be enabled.

The maximum length is MQ\_QSG\_NAME\_LENGTH.

#### **Mode (MQCFIN)**

How the local queue manager is suspended from the cluster (parameter identifier: MQIACF\_MODE).

The value can be:

##### **MQMODE QUIESCE**

Other queue managers in the cluster are told not to send further messages to the local queue manager.

##### **MQMODE FORCE**

All inbound and outbound channels to other queue managers in the cluster are stopped forcibly.

**Note:** This parameter was previously called *Quiesce* (MQIACF QUIESCE), with values MQQO\_YES and MQQO\_NO. The old names can still be used.

### **Error codes**

This command might return the following error codes in the response format header, in addition to the values shown in [“Error codes applicable to all commands”](#) on page 990.

#### **Reason (MQLONG)**

The value can be any of the following values:

##### **MQRCCF\_CLUSTER\_NAME\_CONFLICT**

Cluster name conflict.

## **MQRCCF\_MODE\_VALUE\_ERROR**

Mode value not valid.

## **Structures for PCF commands and responses**

PCF commands and responses have a consistent structure including of a header and any number of parameter structures of defined types.

Commands and responses have the following form:

- PCF header (MQCFH) structure (described in topic [“MQCFH - PCF header” on page 1515](#)), followed by
- Zero or more parameter structures. Each of these is one of the following:
  - PCF byte string filter parameter (MQCFBF, see topic [“MQCFBF - PCF byte string filter parameter” on page 1518](#))
  - PCF byte string parameter (MQCFBS, see topic [“MQCFBS - PCF byte string parameter” on page 1521](#))
  - PCF integer filter parameter (MQCFIF, see topic [“MQCFIF - PCF integer filter parameter” on page 1523](#))
  - PCF integer list parameter (MQCFIL, see topic [“MQCFIL - PCF integer list parameter” on page 1526](#))
  - PCF integer parameter (MQCFIN, see topic [“MQCFIN - PCF integer parameter” on page 1528](#))
  - PCF string filter parameter (MQCFSF, see topic [“MQCFSF - PCF string filter parameter” on page 1530](#))
  - PCF string list parameter (MQCFSL, see topic [“MQCFSL - PCF string list parameter” on page 1534](#))
  - PCF string parameter (MQCFST, see topic [“MQCFST - PCF string parameter” on page 1538](#))

### ***How the PCF command structures are shown***

The PCF command structures are described in a language-independent form.

The declarations are shown in the following programming languages:

- C
- COBOL
- PL/I
- S/390 assembler
- Visual Basic

### **Data types**

For each field of the structure, the data type is given in brackets after the field name. These data types are the elementary data types described in [Data types used in the MQI](#).

### **Initial values and default structures**

See [IBM MQ COPY](#), header, include, and module files for details of the supplied header files that contain the structures, constants, initial values, and default structures.

### ***PCF structures: Usage notes***

The format of the strings in the PCF message determines the settings of the character set fields in the message descriptor to enable conversion of strings within the message.

If all of the strings in a PCF message have the same coded character-set identifier, the *CodedCharSetId* field in the message descriptor MQMD should be set to that identifier when the message is put, and the *CodedCharSetId* fields in the MQCFST, MQCFSL, and MQCFSF structures within the message should be set to MQCCSI\_DEFAULT.

If the format of the PCF message is MQFMT\_ADMIN, MQFMT\_EVENT, or MQFMT\_PCF and some of the strings in the message have different character-set identifiers, the *CodedCharSetId* field in MQMD should be set to MQCCSI\_EMBEDDED when the message is put, and the *CodedCharSetId* fields in the MQCFST, MQCFSL, and MQCFSF structures within the message should all be set to the identifiers that apply.

This enables conversions of the strings within the message, to the *CodedCharSetId* value in the MQMD specified on the MQGET call, if the MQGMO\_CONVERT option is also specified.

For more information about the MQEPH structure, see [MQEPH - Embedded PCF header](#).

**Note:** If you request conversion of the internal strings in the message, the conversion will occur only if the value of the *CodedCharSetId* field in the MQMD of the message is different from the *CodedCharSetId* field of the MQMD specified on the MQGET call.

Do not specify MQCCSI\_EMBEDDED in MQMD when the message is put, with MQCCSI\_DEFAULT in the MQCFST, MQCFSL, or MQCFSF structures within the message, as this will prevent conversion of the message.

### **MQCFH - PCF header**

The MQCFH PCF structure describes the information that is present at the start of the message data of a command message, or a response to a command message. In either case, the message descriptor *Format* field is MQFMT\_ADMIN.

The PCF structures are also used for event messages. In this case the message descriptor *Format* field is MQFMT\_EVENT.

The PCF structures can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT\_PCF (see [Message descriptor for a PCF command](#)). Also in this case, not all the fields in the structure are meaningful. The supplied initial values can be used for most fields, but the application must set the *StrucLength* and *ParameterCount* fields to the values appropriate to the data.

## **Fields for MQCFH**

### **Type (MQLONG)**

Structure type.

This field indicates the content of the message. The following values are valid for commands:

#### **MQCFT\_COMMAND**

Message is a command.

#### **MQCFT\_COMMAND\_XR**

Message is a command to which standard or extended responses might be sent.

This value is required on z/OS.

#### **MQCFT\_RESPONSE**

Message is a response to a command.

#### **MQCFT\_XR\_MSG**

Message is an extended response to a command. It contains informational or error details.

#### **MQCFT\_XR\_ITEM**

Message is an extended response to an Inquire command. It contains item data.

#### **MQCFT\_XR\_SUMMARY**

Message is an extended response to a command. It contains summary information.

#### **MQCFT\_USER**

User-defined PCF message.

### **StrucLength (MQLONG)**

Structure length.

This field is the length in bytes of the MQCFH structure. The value must be:

**MQCFH\_STRUC\_LENGTH**

Length of command format header structure.

**Version (MQLONG)**

Structure version number.

For z/OS, the value must be:

**MQCFH\_VERSION\_3**

Version number for command format header structure.

The following constant specifies the version number of the current version:

**MQCFH\_CURRENT\_VERSION**

Current version of command format header structure.

**Command (MQLONG)**

Command identifier.

For a command message, this field identifies the function to be performed. For a response message, it identifies the command to which this field is the reply. See the description of each command for the value of this field.

**MsgSeqNumber (MQLONG)**

Message sequence number.

This field is the sequence number of the message within a set of related messages. For a command, this field must have the value one (because a command is always contained within a single message). For a response, the field has the value one for the first (or only) response to a command, and increases by one for each successive response to that command.

The last (or only) message in a set has the MQCFC\_LAST flag set in the *Control* field.

**Control (MQLONG)**

Control options.

The following values are valid:

**MQCFC\_LAST**

Last message in the set.

For a command, this value must always be set.

**MQCFC\_NOT\_LAST**

Not the last message in the set.

**CompCode (MQLONG)**

Completion code.

This field is meaningful only for a response; its value is not significant for a command. The following values are possible:

**MQCC\_OK**

Command completed successfully.

**MQCC\_WARNING**

Command completed with warning.

**MQCC\_FAILED**

Command failed.

**MQCC\_UNKNOWN**

Whether command succeeded is not known.

**Reason (MQLONG)**

Reason code qualifying completion code.

This field is meaningful only for a response; its value is not significant for a command.

The possible reason codes that can be returned in response to a command are listed in, [“Definitions of the Programmable Command Formats”](#) on page 983 and in the description of each command.

### ParameterCount (MQLONG)

Count of parameter structures.

This field is the number of parameter structures (MQCFBF, MQCFBS, MQCFIF, MQCFIL, MQCFIN, MQCFSL, MQCFSF, and MQCFST) that follow the MQCFH structure. The value of this field is zero or greater.

### C language declaration

```
typedef struct tagMQCFH {
    MQLONG Type; /* Structure type */
    MQLONG StrucLength; /* Structure length */
    MQLONG Version; /* Structure version number */
    MQLONG Command; /* Command identifier */
    MQLONG MsgSeqNumber; /* Message sequence number */
    MQLONG Control; /* Control options */
    MQLONG CompCode; /* Completion code */
    MQLONG Reason; /* Reason code qualifying completion code */
    MQLONG ParameterCount; /* Count of parameter structures */
} MQCFH;
```

### COBOL language declaration

```
** MQCFH structure
10 MQCFH.
** Structure type
15 MQCFH-TYPE PIC S9(9) BINARY.
** Structure length
15 MQCFH-STRULENGTH PIC S9(9) BINARY.
** Structure version number
15 MQCFH-VERSION PIC S9(9) BINARY.
** Command identifier
15 MQCFH-COMMAND PIC S9(9) BINARY.
** Message sequence number
15 MQCFH-MSGSEQNUMBER PIC S9(9) BINARY.
** Control options
15 MQCFH-CONTROL PIC S9(9) BINARY.
** Completion code
15 MQCFH-COMPCODE PIC S9(9) BINARY.
** Reason code qualifying completion code
15 MQCFH-REASON PIC S9(9) BINARY.
** Count of parameter structures
15 MQCFH-PARAMETERCOUNT PIC S9(9) BINARY.
```

### PL/I language declaration ( z/OS only)

```
dcl
1 MQCFH based,
3 Type fixed bin(31), /* Structure type */
3 StrucLength fixed bin(31), /* Structure length */
3 Version fixed bin(31), /* Structure version number */
3 Command fixed bin(31), /* Command identifier */
3 MsgSeqNumber fixed bin(31), /* Message sequence number */
3 Control fixed bin(31), /* Control options */
3 CompCode fixed bin(31), /* Completion code */
3 Reason fixed bin(31), /* Reason code qualifying completion
code */
3 ParameterCount fixed bin(31); /* Count of parameter structures */
```

### System/390 assembler-language declaration ( z/OS only)

MQCFH	DSECT	
MQCFH_TYPE	DS F	Structure type
MQCFH_STRULENGTH	DS F	Structure length

MQCFH_VERSION	DS	F	Structure version number
MQCFH_COMMAND	DS	F	Command identifier
MQCFH_MSGSEQNUMBER	DS	F	Message sequence number
MQCFH_CONTROL	DS	F	Control options
MQCFH_COMPCODE	DS	F	Completion code
MQCFH_REASON	DS	F	Reason code qualifying completion code
* MQCFH_PARAMETERCOUNT	DS	F	Count of parameter structures
* MQCFH_LENGTH	EQU	*-MQCFH	Length of structure
	ORG	MQCFH	
MQCFH_AREA	DS	CL(MQCFH_LENGTH)	

### Visual Basic language declaration ( Windows only)

```

Type MQCFH
  Type As Long           'Structure type
  StruLength As Long    'Structure length
  Version As Long       'Structure version number
  Command As Long       'Command identifier
  MsgSeqNumber As Long  'Message sequence number
  Control As Long       'Control options
  CompCode As Long     'Completion code
  Reason As Long        'Reason code qualifying completion code
  ParameterCount As Long 'Count of parameter structures
End Type

Global MQCFH_DEFAULT As MQCFH

```

### RPG language declaration ( IBM i only)

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQCFH Structure
D*
D* Structure type
D FHTYP           1      4I 0 INZ(1)
D* Structure length
D FHLEN          5      8I 0 INZ(36)
D* Structure version number
D FHVER          9      12I 0 INZ(1)
D* Command identifier
D FHCMD         13      16I 0 INZ(0)
D* Message sequence number
D FHSEQ         17      20I 0 INZ(1)
D* Control options
D FHCTL         21      24I 0 INZ(1)
D* Completion code
D FHCMP         25      28I 0 INZ(0)
D* Reason code qualifying completion code
D FHREA         29      32I 0 INZ(0)
D* Count of parameter structures
D FHCNT         33      36I 0 INZ(0)
D*


```

### MQCFBF - PCF byte string filter parameter

The MQCFBF PCF structure describes a byte string filter parameter. The format name in the message descriptor is MQFMT\_ADMIN.

The MQCFBF structure is used in Inquire commands to provide a filter description. This filter description is used to filter the results of the Inquire command and return to the user only those objects that satisfy the filter description.

When an MQCFBF structure is present, the Version field in the MQCFH structure at the start of the PCF must be MQCFH\_VERSION\_3 or higher.

 On z/OS, a single filter parameter only is allowed. If multiple MQCFIF, MQCFSF and MQCFBF, or MQCFBF, parameters are specified, the PCF command fails with error MQRCCF\_TOO\_MANY\_FILTERS (MQRCCF 3248).

## Fields for MQCFBF

### Type (MQLONG)

Structure type.

This indicates that the structure is a MQCFBF structure describing a byte string filter parameter. The value must be:

#### **MQCFT\_BYTE\_STRING\_FILTER**

Structure defining a byte string filter.

### StrucLength (MQLONG)

Structure length.

This is the length, in bytes, of the MQCFBF structure, including the string at the end of the structure (the *FilterValue* field). The length must be a multiple of 4, and must be sufficient to contain the string. Bytes between the end of the string and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *FilterValue* field:

#### **MQCFBF\_STRUC\_LENGTH\_FIXED**

Length of fixed part of command format filter string-parameter structure.

### Parameter (MQLONG)

Parameter identifier.

This identifies the parameter that is to be filtered on. The value of this identifier depends on the parameter to be filtered on.

The parameter is one of the following:

- MQBACF\_EXTERNAL\_UOW\_ID
- MQBACF\_Q\_MGR\_UOW\_ID
- MQBACF\_ORIGIN\_UOW\_ID (on z/OS only)

### Operator (MQLONG)

Operator identifier.

This identifies the operator that is being used to evaluate whether the parameter satisfies the filter-value.

Possible values are:

#### **MQCFOP\_GREATER**

Greater than

#### **MQCFOP\_LESS**

Less than

#### **MQCFOP\_EQUAL**

Equal to

#### **MQCFOP\_NOT\_EQUAL**

Not equal to

#### **MQCFOP\_NOT\_LESS**

Greater than or equal to

#### **MQCFOP\_NOT\_GREATER**

Less than or equal to

### FilterValueLength (MQLONG)

Length of filter-value string.

This is the length, in bytes, of the data in the *FilterValue* field. This must be zero or greater, and does not need to be a multiple of 4.

## FilterValue (MQBYTE x FilterValueLength)

Filter value.

This specifies the filter-value that must be satisfied. Use this parameter where the response type of the filtered parameter is a byte string.

**Note:** If the specified byte string is shorter than the standard length of the parameter in MQFMT\_ADMIN command messages, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, it is an error.

## C language declaration

```
typedef struct tagMQCFBF {
    MQLONG  Type;           /* Structure type */
    MQLONG  StructLength;  /* Structure length */
    MQLONG  Parameter;     /* Parameter identifier */
    MQLONG  Operator;      /* Operator identifier */
    MQLONG  FilterValueLength; /* Filter value length */
    MQBYTE  FilterValue[1]; /* Filter value -- first byte */
} MQCFBF;
```

## COBOL language declaration

```
** MQCFBF structure
   10 MQCFBF.
** Structure type
   15 MQCFBF-TYPE PIC S9(9) BINARY.
** Structure length
   15 MQCFBF-STRUCLength PIC S9(9) BINARY.
** Parameter identifier
   15 MQCFBF-PARAMETER PIC S9(9) BINARY.
** Operator identifier
   15 MQCFBF-OPERATOR PIC S9(9) BINARY.
** Filter value length
   15 MQCFBF-FILTERVALUELENGTH PIC S9(9) BINARY.
```

## PL/I language declaration ( z/OS only)

```
dcl
  1 MQCFBF based,
  3 Type fixed bin(31)
    init(MQCFBT_BYTE_STRING_FILTER), /* Structure type */
  3 StructLength fixed bin(31)
    init(MQCFBF_STRUC_LENGTH_FIXED), /* Structure length */
  3 Parameter fixed bin(31)
    init(0), /* Parameter identifier */
  3 Operator fixed bin(31)
    init(0), /* Operator identifier */
  3 FilterValueLength fixed bin(31)
    init(0); /* Filter value length */
```

## System/390 assembler-language declaration (z/OS only)

```
MQCFBF          DSECT
MQCFBF_TYPE     DS F   Structure type
MQCFBF_STRUCLength DS F   Structure length
MQCFBF_PARAMETER DS F   Parameter identifier
MQCFBF_OPERATOR DS F   Operator identifier
MQCFBF_FILTERVALUELENGTH DS F   Filter value length
MQCFBF_LENGTH  EQU *-MQCFIF Length of structure
MQCFBF         ORG MQCFBF
MQCFBF_AREA    DS    CL(MQCFBF_LENGTH)
```



## Visual Basic language declaration ( Windows only)

```
Type MQCFBF
  Type As Long 'Structure type'
  StrucLength As Long 'Structure length'
  Parameter As Long 'Parameter identifier'
  Operator As Long 'Operator identifier'
  FilterValueLength As Long 'Filter value length'
  FilterValue As 1 'Filter value -- first byte'
End Type
Global MQCFBF_DEFAULT As MQCFBF
```

## RPG language declaration ( IBM i only)

```
D* MQCFBF Structure
D*
D* Structure type
D FBFTYP 1 4I 0 INZ(15)
D* Structure length
D FBFLen 5 8I 0 INZ(20)
D* Parameter identifier
D FBFRM 9 12I 0 INZ(0)
D* Operator identifier
D FBFOp 13 16I 0 INZ(0)
D* Filter value length
D FBFFVL 17 20I 0 INZ(0)
D* Filter value -- first byte
D FBFFV 21 21 INZ
```

### **MQCFBS - PCF byte string parameter**

The MQCFBS PCF structure describes a byte-string parameter in a PCF message. The format name in the message descriptor is MQFMT\_ADMIN.

When an MQCFBS structure is present, the *Version* field in the MQCFH structure at the start of the PCF must be MQCFH\_VERSION\_2 or greater.

In a user PCF message, the *Parameter* field has no significance, and can be used by the application for its own purposes.

The structure ends with a variable-length byte string; see the *String* field in the following section for further details.

### **Fields for MQCFBS**

#### **Type (MQLONG)**

Structure type.

This indicates that the structure is an MQCFBS structure describing byte string parameter. The value must be:

#### **MQCFT\_BYTE\_STRING**

Structure defining a byte string.

#### **StrucLength (MQLONG)**

Structure length.

This is the length in bytes of the MQCFBS structure, including the variable-length string at the end of the structure (the *String* field). The length must be a multiple of four, and must be sufficient to contain the string; any bytes between the end of the string and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *String* field:

#### **MQCFBS\_STRUC\_LENGTH\_FIXED**

Length of fixed part of MQCFBS structure.

### Parameter (MQLONG)

Parameter identifier.

This identifies the parameter with a value that is contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see “MQCFH - PCF header” on page 1515 for details. In user PCF messages (MQCFT\_USER), this field has no significance.

The parameter is from the MQBACF\_\* group of parameters.

### StringLength (MQLONG)

Length of string.

This is the length in bytes of the data in the *string* field; it must be zero or greater. This length does not need to be a multiple of four.

### String (MQBYTE x StringLength)

String value.

This is the value of the parameter identified by the *parameter* field. The string is a byte string, and so is not subject to character-set conversion when sent between different systems.

**Note:** A null character in the string is treated as normal data, and does not act as a delimiter for the string. For MQFMT\_ADMIN messages, if the specified string is shorter than the standard length of the *parameter*, the omitted characters are assumed to be nulls. If the specified string is longer than the standard length, it is an error.

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure must be allocated dynamically, and pointers used to address the fields within it.
- For other programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, you must include MQCFBS in a larger structure, and declare additional fields following MQCFBS, to represent the *String* field as required.

### C language declaration

```
typedef struct tagMQCFBS {
    MQLONG  Type;          /* Structure type */
    MQLONG  StrucLength;   /* Structure length */
    MQLONG  Parameter;     /* Parameter identifier */
    MQLONG  StringLength; /* Length of string */
    MQBYTE  String[1];    /* String value - first byte */

} MQCFBS;
```

### COBOL language declaration

```
**      MQCFBS structure
10  MQCFBS.
**      Structure type
15  MQCFBS-TYPE          PIC S9(9) BINARY.
**      Structure length
15  MQCFBS-STRUCLNGTH  PIC S9(9) BINARY.
**      Parameter identifier
15  MQCFBS-PARAMETER   PIC S9(9) BINARY.
**      Length of string
15  MQCFBS-STRINGLENGTH PIC S9(9) BINARY.
```

### PL/I language declaration ( z/OS only)

```
dcl
```

```

1 MQCFBS based,
3 Type      fixed bin(31), /* Structure type */
3 StructLength fixed bin(31), /* Structure length */
3 Parameter  fixed bin(31), /* Parameter identifier */
3 StringLength fixed bin(31) /* Length of string */

```

### System/390 assembler-language declaration (z/OS only)

```

MQCFBS          DSECT
MQCFBS_TYPE     DS  F          Structure type
MQCFBS_STRUCLNGTH DS  F          Structure length
MQCFBS_PARAMETER DS  F          Parameter identifier
MQCFBS_STRINGLENGTH DS  F          Length of string
MQCFBS_AREA     DS  CL(MQCFBS_LENGTH)

```

### Visual Basic language declaration ( Windows only)

```

Type MQCFBS
  Type As Long      ' Structure type
  StructLength As Long ' Structure length
  Parameter As Long ' Parameter identifier
  StringLength As Long ' Operator identifier
  String as 1      ' String value - first byte
End Type

Global MQCFBS_DEFAULT As MQCFBS

```

### RPG language declaration ( IBM i only)

```

D* MQCFBS Structure
D*
D* Structure type
D  BSTYP          1      4I 0 INZ(3)
D* Structure length
D  BSLEN          5      8I 0 INZ(16)
D* Parameter identifier
D  BSPRM          9      12I 0 INZ(0)
D* Length of string
D  BSSTL         13      16I 0 INZ(0)
D* String value - first byte
D  BSSRA         17      16
D*


```

### ***MQCFIF - PCF integer filter parameter***

The MQCFIF PCF structure describes an integer filter parameter. The format name in the message descriptor is MQFMT\_ADMIN.

The MQCFIF structure is used in Inquire commands to provide a filter condition. This filter condition is used to filter the results of the Inquire command and return to the user only those objects that satisfy the filter condition.

When an MQCFIF structure is present, the Version field in the MQCFH structure at the start of the PCF must be MQCFH\_VERSION\_3 or higher.

 On z/OS, a single filter parameter only is allowed. If multiple MQCFIF, MQCFSF and MQCFBF, or MQCFBF, parameters are specified, the PCF command fails with error MQRCCF\_TOO\_MANY\_FILTERS (MQRCCF 3248).

### Fields for MQCFIF

#### Type (MQLONG)

Structure type.

This indicates that the structure is an MQCFIF structure describing an integer filter parameter. The value must be:

**MQCFT\_INTEGER\_FILTER**

Structure defining an integer filter.

**StrucLength (MQLONG)**

Structure length.

This is the length in bytes of the MQCFIF structure. The value must be:

**MQCFIF\_STRUC\_LENGTH**

Length of command format integer-parameter structure.

**Parameter (MQLONG)**

Parameter identifier.

This identifies the parameter that is to be filtered on. The value of this identifier depends on the parameter to be filtered on. Any of the parameters which can be used in the Inquire command can be used in this field.

The parameter is from the following groups of parameters:

- MQIA\_\*
- MQIACF\_\*
- MQIAMO\_\*
- MQIACH\_\*

**Operator (MQLONG)**

Operator identifier.

This identifies the operator that is being used to evaluate whether the parameter satisfies the filter-value.

Possible values are:

**MQCFOP\_GREATER**

Greater than

**MQCFOP\_LESS**

Less than

**MQCFOP\_EQUAL**

Equal to

**MQCFOP\_NOT\_EQUAL**

Not equal to

**MQCFOP\_NOT\_LESS**

Greater than or equal to

**MQCFOP\_NOT\_GREATER**

Less than or equal to

**MQCFOP\_CONTAINS**

Contains a specified value. Use MQCFOP\_CONTAINS when filtering on lists of values or integers.

**MQCFOP\_EXCLUDES**

Does not contain a specified value. Use MQCFOP\_EXCLUDES when filtering on lists of values or integers.

See the *FilterValue* description for details telling you which operators can be used in which circumstances.

**FilterValue (MQLONG)**

Filter value identifier.

This specifies the filter-value that must be satisfied.

Depending on the parameter, the value and the permitted operators can be:

- An explicit integer value, if the parameter takes a single integer value.

You can only use the following operators:

- MQCFOP\_GREATER
  - MQCFOP\_LESS
  - MQCFOP\_EQUAL
  - MQCFOP\_NOT\_EQUAL
  - MQCFOP\_NOT\_GREATER
  - MQCFOP\_NOT\_LESS
- An MQ constant, if the parameter takes a single value from a possible set of values (for example, the value MQCHT\_SENDER on the **ChannelType** parameter). You can only use MQCFOP\_EQUAL or MQCFOP\_NOT\_EQUAL.
  - An explicit value or an MQ constant, as the case might be, if the parameter takes a list of values. You can use either MQCFOP\_CONTAINS or MQCFOP\_EXCLUDES. For example, if the value 6 is specified with the operator MQCFOP\_CONTAINS, all items where one of the parameter values is 6 are listed.

For example, if you need to filter on queues that are enabled for put operations in your Inquire Queue command, the parameter would be MQIA\_INHIBIT\_PUT and the filter-value would be MQQA\_PUT\_ALLOWED.

The filter value must be a valid value for the parameter being tested.

## C language declaration

```
typedef struct tagMQCFIF {
    MQLONG Type; /* Structure type */
    MQLONG StrucLength; /* Structure length */
    MQLONG Parameter; /* Parameter identifier */
    MQLONG Operator; /* Operator identifier */
    MQLONG FilterValue; /* Filter value */
} MQCFIF;
```

## COBOL language declaration

```
** MQCFIF structure
10 MQCFIF.
** Structure type
15 MQCFIF-TYPE PIC S9(9) BINARY.
** Structure length
15 MQCFIF-STRUCLength PIC S9(9) BINARY.
** Parameter identifier
15 MQCFIF-PARAMETER PIC S9(9) BINARY.
** Operator identifier
15 MQCFIF-OPERATOR PIC S9(9) BINARY.
** Filter value
15 MQCFIF-FILTERVALUE PIC S9(9) BINARY.
```

## PL/I language declaration ( z/OS only)

```
dcl
1 MQCFIF based,
3 Type fixed bin(31), /* Structure type */
3 StrucLength fixed bin(31), /* Structure length */
3 Parameter fixed bin(31), /* Parameter identifier */
3 Operator fixed bin(31) /* Operator identifier */
3 FilterValue fixed bin(31); /* Filter value */
```

## System/390 assembler-language declaration ( z/OS only)

```
MQCFIF                DSECT
MQCFIF_TYPE            DS  F      Structure type
MQCFIF_STRUCLength    DS  F      Structure length
MQCFIF_PARAMETER      DS  F      Parameter identifier
MQCFIF_OPERATOR       DS  F      Operator identifier
MQCFIF_FILTERVALUE    DS  F      Filter value
MQCFIF_LENGTH         EQU *-MQCFIF Length of structure
MQCFIF_LENGTH         ORG  MQCFIF
MQCFIF_AREA           DS  CL(MQCFIF_LENGTH)
```

## Visual Basic language declaration ( Windows only)

```
Type MQCFIF
  Type As Long          ' Structure type
  StruLength As Long   ' Structure length
  Parameter As Long    ' Parameter identifier
  Operator As Long     ' Operator identifier
  FilterValue As Long ' Filter value
End Type

Global MQCFIF_DEFAULT As MQCFIF
```

## RPG language declaration ( IBM i only)

```
D* MQCFIF Structure
D*
D* Structure type
D FIFTYP                1      4I 0 INZ(3)
D* Structure length
D FIFLEN                5      8I 0 INZ(16)
D* Parameter identifier
D FIFPRM                9      12I 0 INZ(0)
D* Operator identifier
D FIFOP                 13     16I 0 INZ(0)
D* Condition identifier
D FIFFV                 17     20I 0 INZ(0)
D*
```

### **MQCFIL - PCF integer list parameter**

The MQCFIL PCF structure describes an integer-list parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT\_ADMIN.

The MQCFIL structure can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT\_PCF (see [Message descriptor for a PCF command](#)). Also in this case, not all the fields in the structure are meaningful. The supplied initial values can be used for most fields, but the application must set the *StrucLength*, *Count*, and *Values* fields to the values appropriate to the data.

The structure ends with a variable-length array of integers; see the *Values* field in the following section for further details.

### **Fields for MQCFIL**

#### **Type (MQLONG)**

Structure type.

This indicates that the structure is an MQCFIL structure describing an integer-list parameter. The value must be:

#### **MQCFT\_INTEGER\_LIST**

Structure defining an integer list.

### StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFIL structure, including the array of integers at the end of the structure (the *Values* field). The length must be a multiple of four, and must be sufficient to contain the array; any bytes between the end of the array and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *Values* field:

#### MQCFIL\_STRUC\_LENGTH\_FIXED

Length of fixed part of command format integer-list parameter structure.

### Parameter (MQLONG)

Parameter identifier.

This identifies the parameter with values that are contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see [“MQCFH - PCF header” on page 1515](#) for details.

The parameter is from the following groups of parameters:

- MQIA\_\*
- MQIACF\_\*
- MQIAMO\_\*
- MQIACH\_\*

### Count (MQLONG)

Count of parameter values.

This is the number of elements in the *Values* array; it must be zero or greater.

### Values (MQLONG x Count)

Parameter values.

This is an array of values for the parameter identified by the *Parameter* field. For example, for MQIACF\_Q\_ATTRS, this field is a list of attribute selectors (MQCA\_\* and MQIA\_\* values).

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure must be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL, PL/I, RPG, and System/390 assembler programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, you must include MQCFIL in a larger structure, and declare additional fields following MQCFIL, to represent the *Values* field as required.

### C language declaration

```
typedef struct tagMQCFIL {
    MQLONG  Type;          /* Structure type */
    MQLONG  StrucLength;   /* Structure length */
    MQLONG  Parameter;     /* Parameter identifier */
    MQLONG  Count;        /* Count of parameter values */
    MQLONG  Values[1];     /* Parameter values - first element */
} MQCFIL;
```

### COBOL language declaration

```
**  MQCFIL structure
** 10 MQCFIL.
**   Structure type
** 15 MQCFIL-TYPE          PIC S9(9) BINARY.
```

```

**      Structure length
**      15 MQCFIL-STRUCLength PIC S9(9) BINARY.
**      Parameter identifier
**      15 MQCFIL-PARAMETER   PIC S9(9) BINARY.
**      Count of parameter values
**      15 MQCFIL-COUNT      PIC S9(9) BINARY.

```

### PL/I language declaration ( z/OS only)

```

dcl
  1 MQCFIL based,
  3 Type      fixed bin(31), /* Structure type */
  3 StrucLength fixed bin(31), /* Structure length */
  3 Parameter  fixed bin(31), /* Parameter identifier */
  3 Count     fixed bin(31); /* Count of parameter values */

```

### System/390 assembler-language declaration ( z/OS only)

```

MQCFIL          DSECT
MQCFIL_TYPE     DS   F           Structure type
MQCFIL_STRUCLength DS   F           Structure length
MQCFIL_PARAMETER DS   F           Parameter identifier
MQCFIL_COUNT    DS   F           Count of parameter values
MQCFIL_LENGTH   EQU  *-MQCFIL Length of structure
MQCFIL_AREA     DS   CL(MQCFIL_LENGTH)

```

### Visual Basic language declaration ( Windows only)

```

Type MQCFIL
  Type As Long      ' Structure type
  StrucLength As Long ' Structure length
  Parameter As Long ' Parameter identifier
  Count As Long     ' Count of parameter values
End Type

Global MQCFIL_DEFAULT As MQCFIL

```

### RPG language declaration ( IBM i only)

```

D* MQCFIL Structure
D*
D* Structure type
D  ILTYP          1      4I 0 INZ(5)
D* Structure length
D  ILLEN          5      8I 0 INZ(16)
D* Parameter identifier
D  ILPRM          9      12I 0 INZ(0)
D* Count of parameter values
D  ILCNT          13     16I 0 INZ(0)
D*

```

### ***MQCFIN - PCF integer parameter***

The MQCFIN PCF structure describes an integer parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT\_ADMIN.

The MQCFIN structure can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT\_PCF (see Message descriptor for a PCF command). Also in this case, not all the fields in the structure are meaningful. The supplied initial values can be used for most fields, but the application must set the *Value* field to the value appropriate to the data.



## Fields for MQCFIN

### Type (MQLONG)

Structure type.

This indicates that the structure is an MQCFIN structure describing an integer parameter. The value must be:

#### MQCFT\_INTEGER

Structure defining an integer.

### StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFIN structure. The value must be:

#### MQCFIN\_STRUC\_LENGTH

Length of command format integer-parameter structure.

### Parameter (MQLONG)

Parameter identifier.

This identifies the parameter with a value that is contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see [“MQCFH - PCF header” on page 1515](#) for details.

The parameter is from the following groups of parameters:

- MQIA\_\*
- MQIACF\_\*
- MQIAMO\_\*
- MQIACH\_\*

### Value (MQLONG)

Parameter value.

This is the value of the parameter identified by the *Parameter* field.

## C language declaration

```
typedef struct tagMQCFIN {
    MQLONG  Type;          /* Structure type */
    MQLONG  StrucLength;  /* Structure length */
    MQLONG  Parameter;    /* Parameter identifier */
    MQLONG  Value;        /* Parameter value */
} MQCFIN;
```

## COBOL language declaration

```
** MQCFIN structure
10 MQCFIN.
** Structure type
15 MQCFIN-TYPE          PIC S9(9) BINARY.
** Structure length
15 MQCFIN-STRUCLNGTH  PIC S9(9) BINARY.
** Parameter identifier
15 MQCFIN-PARAMETER   PIC S9(9) BINARY.
** Parameter value
15 MQCFIN-VALUE       PIC S9(9) BINARY.
```

## PL/I language declaration ( z/OS only)

```
dcl
  1 MQCFIN based,
  3 Type          fixed bin(31), /* Structure type */
```

```

3 StrucLength fixed bin(31), /* Structure length */
3 Parameter   fixed bin(31), /* Parameter identifier */
3 Value      fixed bin(31); /* Parameter value */

```

### System/390 assembler-language declaration ( z/OS only)

```

MQCFIN                DSECT
MQCFIN_TYPE           DS    F           Structure type
MQCFIN_STRULENGTH     DS    F           Structure length
MQCFIN_PARAMETER      DS    F           Parameter identifier
MQCFIN_VALUE          DS    F           Parameter value
MQCFIN_LENGTH         EQU    *-MQCFIN Length of structure
MQCFIN_AREA           ORG    MQCFIN
                     DS    CL(MQCFIN_LENGTH)

```

### Visual Basic language declaration ( Windows only)

```

Type MQCFIN
  Type As Long          ' Structure type
  StrucLength As Long   ' Structure length
  Parameter As Long     ' Parameter identifier
  Value As Long         ' Parameter value
End Type

Global MQCFIN_DEFAULT As MQCFIN

```

### RPG language declaration ( IBM i only)

```


D* MQCFIN Structure
D*
D* Structure type
D  INTYP                1      4I 0 INZ(3)
D* Structure length
D  INLEN                5      8I 0 INZ(16)
D* Parameter identifier
D  INPRM               9      12I 0 INZ(0)
D* Parameter value
D  INVAL              13     16I 0 INZ(0)
D*

```

### MQCFSF - PCF string filter parameter

The MQCFSF PCF structure describes a string filter parameter. The format name in the message descriptor is MQFMT\_ADMIN.

The MQCFSF structure is used in Inquire commands to provide a filter condition. This filter condition is used to filter the results of the Inquire command and return to the user only those objects that satisfy the filter condition.

 On z/OS, a single filter parameter only is allowed. If multiple MQCFIF, MQCFSF and MQCFBF, or MQCFBF, parameters are specified, the PCF command fails with error MQRCCF\_TOO\_MANY\_FILTERS (MQRCCF 3248).

The results of filtering character strings on EBCDIC-based systems might be different from those results achieved on ASCII-based systems. This difference is because comparison of character strings is based on the collating sequence of the internal built-in values representing the characters.

When an MQCFSF structure is present, the Version field in the MQCFH structure at the start of the PCF must be MQCFH\_VERSION\_3 or higher.

### Fields for MQCFSF

#### Type (MQLONG)

Structure type.

This indicates that the structure is an MQCFSF structure describing a string filter parameter. The value must be:

**MQCFT\_STRING\_FILTER**

Structure defining a string filter.

**StrucLength (MQLONG)**

Structure length.

This is the length in bytes of the MQCFSF structure. The value must be:

**MQCFSF\_STRUC\_LENGTH**

MQCFSF\_STRUC\_LENGTH is the length, in bytes, of the MQCFSF structure, including the string at the end of the structure (the *FilterValue* field). The length must be a multiple of 4, and must be sufficient to contain the string. Bytes between the end of the string and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *FilterValue* field:

**MQCFSF\_STRUC\_LENGTH\_FIXED**

Length of fixed part of command format filter string-parameter structure.

**Parameter (MQLONG)**

Parameter identifier.

This identifies the parameter that is to be filtered on. The value of this identifier depends on the parameter to be filtered on. Any of the parameters which can be used in the Inquire command can be used in this field.

The parameter is from the following groups of parameters:

- MQCA\_\*
- MQCACF\_\*
- MQCAMO\_\*
- MQCACH\_\*

**Operator (MQLONG)**

Operator identifier.

This identifies the operator that is being used to evaluate whether the parameter satisfies the filter-value.

Possible values are:

**MQCFOP\_GREATER**

Greater than

**MQCFOP\_LESS**

Less than

**MQCFOP\_EQUAL**

Equal to

**MQCFOP\_NOT\_EQUAL**

Not equal to

**MQCFOP\_NOT\_LESS**

Greater than or equal to

**MQCFOP\_NOT\_GREATER**

Less than or equal to

**MQCFOP\_LIKE**

Matches a generic string

**MQCFOP\_NOT\_LIKE**

Does not match a generic string

**MQCFOP\_CONTAINS**

Contains a specified string. Use MQCFOP\_CONTAINS when filtering on lists of strings.

**MQCFOP\_EXCLUDES**

Does not contain a specified string. Use MQCFOP\_EXCLUDES when filtering on lists of strings.

**MQCFOP\_CONTAINS\_GEN**

Contains an item which matches a generic string. Use MQCFOP\_CONTAINS\_GEN when filtering on lists of strings.

**MQCFOP\_EXCLUDES\_GEN**

Does not contain any item which matches a generic string. Use MQCFOP\_EXCLUDES\_GEN when filtering on lists of strings.

See the *FilterValue* description for details telling you which operators can be used in which circumstances.

**CodedCharSetId (MQLONG)**

Coded character set identifier.

This specifies the coded character set identifier of the data in the *FilterValue* field. The following special value can be used:

**MQCCSI\_DEFAULT**


Default character set identifier.

The string data is in the character set defined by the *CodedCharSetId* field in the MQ header structure that *precedes* the MQCFH structure, or by the *CodedCharSetId* field in the MQMD if the MQCFH structure is at the start of the message.

**FilterValueLength (MQLONG)**

Length of filter-value string.

This is the length, in bytes, of the data in the *FilterValue* field. This parameter must be zero or greater, and does not need to be a multiple of 4.

**Note:**  On z/OS there is a 256 character limit for the filter-value of the MQSC **WHERE** clause. This limit is not in place for other platforms.

**FilterValue (MQCHAR x FilterValueLength)**

Filter value.

This specifies the filter-value that must be satisfied. Depending on the parameter, the value and the permitted operators can be:

- An explicit string value.
  - You can only use the following operators:
    - MQCFOP\_GREATER
    - MQCFOP\_LESS
    - MQCFOP\_EQUAL
    - MQCFOP\_NOT\_EQUAL
    - MQCFOP\_NOT\_GREATER
    - MQCFOP\_NOT\_LESS
  - A generic string value. This field is a character string with an asterisk at the end, for example ABC\*. The operator must be either MQCFOP\_LIKE or MQCFOP\_NOT\_LIKE. The characters must be valid for the attribute you are testing. If the operator is MQCFOP\_LIKE, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is MQCFOP\_NOT\_LIKE, all items where the attribute value does not begin with the string are listed.
  - If the parameter takes a list of string values, the operator can be:
    - MQCFOP\_CONTAINS

- MQCFOP\_EXCLUDES
- MQCFOP\_CONTAINS\_GEN
- MQCFOP\_EXCLUDES\_GEN

An item in a list of values. The value can be explicit or generic. If it is explicit, use MQCFOP\_CONTAINS or MQCFOP\_EXCLUDES as the operator. For example, if the value DEF is specified with the operator MQCFOP\_CONTAINS, all items where one of the attribute values is DEF are listed. If it is generic, use MQCFOP\_CONTAINS\_GEN or MQCFOP\_EXCLUDES\_GEN as the operator. If ABC\* is specified with the operator MQCFOP\_CONTAINS\_GEN, all items where one of the attribute values begins with ABC are listed.

**Note:**

1. If the specified string is shorter than the standard length of the parameter in MQFMT\_ADMIN command messages, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, it is an error.
2. When the queue manager reads an MQCFSF structure in an MQFMT\_ADMIN message from the command input queue, the queue manager processes the string as though it had been specified on an MQI call. This processing means that within the string, the first null and the characters following it (up to the end of the string) are treated as blanks.
3. On z/OS there is a 256 character limit for the filter-value of the MQSC **WHERE** clause. This limit is not in place for other platforms.

The filter value must be a valid value for the parameter being tested.

**C language declaration**

```
typedef struct tagMQCFSF {
    MQLONG  Type;           /* Structure type */
    MQLONG  StrucLength;    /* Structure length */
    MQLONG  Parameter;     /* Parameter identifier */
    MQLONG  Operator;      /* Operator identifier */
    MQLONG  CodedCharSetId; /* Coded character set identifier */
    MQLONG  FilterValueLength /* Filtervalue length */
    MQCHAR[1]  FilterValue; /* Filter value */
} MQCFSF;
```

**COBOL language declaration**

```
**      MQCFSF structure
10      MQCFSF.
**      Structure type
15      MQCFSF-TYPE          PIC S9(9) BINARY.
**      Structure length
15      MQCFSF-STRULENGTH PIC S9(9) BINARY.
**      Parameter identifier
15      MQCFSF-PARAMETER   PIC S9(9) BINARY.
**      Operator identifier
15      MQCFSF-OPERATOR   PIC S9(9) BINARY.
**      Coded character set identifier
15      MQCFSF-CODEDCHARSETID PIC S9(9) BINARY.
**      Filter value length
15      MQCFSF-FILTERVALUE PIC S9(9) BINARY.
```

**PL/I language declaration ( z/OS only)**

```
dcl
  1 MQCFSF based,
  3 Type          fixed bin(31), /* Structure type */
  3 StrucLength   fixed bin(31), /* Structure length */
  3 Parameter     fixed bin(31), /* Parameter identifier */
  3 Operator      fixed bin(31) /* Operator identifier */
```

```

3 CodedCharSetId    fixed bin(31) /* Coded character set identifier */
3 FilterValueLength fixed bin(31); /* Filter value length */

```

### System/390 assembler-language declaration ( z/OS only)

```

MQCFSF                DSECT
MQCFSF_TYPE            DS    F           Structure type
MQCFSF_STRUCLNGTH     DS    F           Structure length
MQCFSF_PARAMETER      DS    F           Parameter identifier
MQCFSF_OPERATOR       DS    F           Operator identifier
MQCFSF_CODEDCHARSETID DS    F           Coded character set identifier
MQCFSF_FILTERVALUELENGTH DS    F       Filter value length
MQCFSF_LENGTH         EQU    *-MQCFSF  Length of structure
                       ORG    MQCFSF
MQCFSF_AREA           DS    CL(MQCFSF_LENGTH)

```

### Visual Basic language declaration ( Windows only)

```

Type MQCFSF
  Type As Long          ' Structure type
  StructLength As Long ' Structure length
  Parameter As Long     ' Parameter identifier
  Operator As Long      ' Operator identifier
  CodedCharSetId As Long ' Coded character set identifier
  FilterValueLength As Long ' Operator identifier
  FilterValue As String*1 ' Condition value -- first character
End Type

Global MQCFSF_DEFAULT As MQCFSF

```

### RPG language declaration ( IBM i only)

```

D* MQCFSF Structure
D*
D* Structure type
D  FISTYP                1      4I 0 INZ(3)
D* Structure length
D  FSFLEN                5      8I 0 INZ(16)
D* Parameter identifier
D  FSFPRM                9      12I 0 INZ(0)
D* Reserved field
D  FSFRSV               13      16I 0 INZ(0)
D* Parameter value
D  FSFVAL               17      16
D* Structure type
D  FSFTYP               17      20I 0
D* Structure length
D  FSFLEN               21      24I 0
D* Parameter value
D  FSFPRM               25      28I 0
D* Operator identifier
D  FSFOP                29      32I 0
D* Coded character set identifier
D  FSFCSI               33      36I 0
D* Length of condition
D  FSFFVL               37      40 0
D* Condition value -- first character
D  FSFFV                41      41
D*

```

### **MQCFSL - PCF string list parameter**

The MQCFSL PCF structure describes a string-list parameter in a message which is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT\_ADMIN.

The MQCFSL structure can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT\_PCF (see [Message descriptor for a PCF command](#)). Also in this case, not all the fields in the structure are meaningful. The supplied initial values can be used for most fields,

but the application must set the *StrucLength*, *Count*, *StringLength*, and *Strings* fields to the values appropriate to the data.

The structure ends with a variable-length array of character strings; see the *Strings* field section for further details.

See [“PCF structures: Usage notes” on page 1514](#) for further information about how to use the structure.

## Fields for MQCFSL

### Type (MQLONG)

Structure type.

This indicates that the structure is an MQCFSL structure describing a string-list parameter. The value must be:

#### **MQCFT\_STRING\_LIST**

Structure defining a string list.

### StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFSL structure, including the data at the end of the structure (the *Strings* field). The length must be a multiple of four, and must be sufficient to contain all the strings; any bytes between the end of the strings and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *Strings* field:

#### **MQCFSL\_STRUC\_LENGTH\_FIXED**

Length of fixed part of command format string-list parameter structure.

### Parameter (MQLONG)

Parameter identifier.

This identifies the parameter with values that are contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see [“MQCFH - PCF header” on page 1515](#) for details.

The parameter is from the following groups of parameters:

- MQCA\_\*
- MQCACF\_\*
- MQCAMO\_\*
- MQCACH\_\*

### CodedCharSetId (MQLONG)

Coded character set identifier.

This specifies the coded character set identifier of the data in the *Strings* field. The following special value can be used:

#### **MQCCSI\_DEFAULT**

Default character set identifier.

The string data is in the character set defined by the *CodedCharSetId* field in the MQ header structure that *precedes* the MQCFH structure, or by the *CodedCharSetId* field in the MQMD if the MQCFH structure is at the start of the message.

### Count (MQLONG)

Count of parameter values.

This is the number of strings present in the *Strings* field; it must be zero or greater.

## StringLength (MQLONG)

Length of one string.

This is the length in bytes of one parameter value, that is the length of one string in the *Strings* field; all the strings are this length. The length must be zero or greater, and need not be a multiple of four.

## Strings (MQCHAR x StringLength x Count)

String values.

This is a set of string values for the parameter identified by the *Parameter* field. The number of strings is given by the *Count* field, and the length of each string is given by the *StringLength* field. The strings are concatenated together, with no bytes skipped between adjacent strings. The total length of the strings is the length of one string multiplied by the number of strings present (that is, *StringLength* x *Count*).

- In MQFMT\_ADMIN command messages, if the specified string is shorter than the standard length of the parameter, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, it is an error.
- In MQFMT\_ADMIN response messages, string parameters might be returned padded with blanks to the standard length of the parameter.
- In MQFMT\_EVENT messages, trailing blanks might be omitted from string parameters (that is, the string might be shorter than the standard length of the parameter).

In all cases, *StringLength* gives the length of the string present in the message.

The strings can contain any characters that are in the character set defined by *CodedCharSetId*, and that are valid for the parameter identified by *Parameter*.

**Note:** When the queue manager reads an MQCFSL structure in an MQFMT\_ADMIN message from the command input queue, the queue manager processes each string in the list as though it had been specified on an MQI call. This processing means that within each string, the first null, and the characters following it (up to the end of the string) are treated as blanks.

In responses and all other cases, a null character in a string is treated as normal data, and does not act as a delimiter for the string. This treatment means that when a receiving application reads a MQFMT\_PCF, MQFMT\_EVENT, or MQFMT\_ADMIN message, the receiving application receives all the data specified by the sending application.

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure must be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL, PL/I, RPG, and System/390 assembler programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, you must include MQCFSL in a larger structure, and declare additional fields following MQCFSL, to represent the *Strings* field as required.

## C language declaration

```
typedef struct tagMQCFSL {
    MQLONG  Type;           /* Structure type */
    MQLONG  StringLength;  /* Structure length */
    MQLONG  Parameter;     /* Parameter identifier */
    MQLONG  CodedCharSetId; /* Coded character set identifier */
    MQLONG  Count;         /* Count of parameter values */
    MQLONG  StringLength;  /* Length of one string */
    MQCHAR  Strings[1];    /* String values - first
                           character */
} MQCFSL;
```



## COBOL language declaration

```
** MQCFSL structure
10 MQCFSL.
** Structure type
15 MQCFSL-TYPE PIC S9(9) BINARY.
** Structure length
15 MQCFSL-STRUCLength PIC S9(9) BINARY.
** Parameter identifier
15 MQCFSL-PARAMETER PIC S9(9) BINARY.
** Coded character set identifier
15 MQCFSL-CODEDCHARSETID PIC S9(9) BINARY.
** Count of parameter values
15 MQCFSL-COUNT PIC S9(9) BINARY.
** Length of one string
15 MQCFSL-STRINGLENGTH PIC S9(9) BINARY.
```

## PL/I language declaration ( z/OS only)

```
dcl
1 MQCFSL based,
3 Type fixed bin(31), /* Structure type */
3 StrucLength fixed bin(31), /* Structure length */
3 Parameter fixed bin(31), /* Parameter identifier */
3 CodedCharSetId fixed bin(31), /* Coded character set identifier */
3 Count fixed bin(31), /* Count of parameter values */
3 StringLength fixed bin(31); /* Length of one string */
```

## System/390 assembler-language declaration ( z/OS only)

```
MQCFSL DSECT
MQCFSL_TYPE DS F Structure type
MQCFSL_STRUCLength DS F Structure length
MQCFSL_PARAMETER DS F Parameter identifier
MQCFSL_CODEDCHARSETID DS F Coded character set
* identifier
MQCFSL_COUNT DS F Count of parameter values
MQCFSL_STRINGLENGTH DS F Length of one string
MQCFSL_LENGTH EQU *-MQCFSL Length of structure
ORG MQCFSL
MQCFSL_AREA DS CL(MQCFSL_LENGTH)
```

## Visual Basic language declaration ( Windows only)

```
Type MQCFSL
Type As Long ' Structure type
StrucLength As Long ' Structure length
Parameter As Long ' Parameter identifier
CodedCharSetId As Long ' Coded character set identifier
Count As Long ' Count of parameter values
StringLength As Long ' Length of one string
End Type

Global MQCFSL_DEFAULT As MQCFSL
```

## RPG language declaration ( IBM i only)

```
D* MQCFSL Structure
D*
D* Structure type
D SLTYP 1 4I 0 INZ(6)
D* Structure length
D SLLen 5 8I 0 INZ(24)
D* Parameter identifier
D SLPRM 9 12I 0 INZ(0)
D* Coded character set identifier
D SLCSI 13 16I 0 INZ(0)
```

D*	Count of parameter values		
D	SLCNT	17	20I 0 INZ(0)
D*	Length of one string		
D	SLSTL	21	24I 0 INZ(0)

### **MQCFST - PCF string parameter**

The MQCFST PCF structure describes a string parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT\_ADMIN.

The MQCFST structure can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT\_PCF (see [Message descriptor for a PCF command](#)). Also in this case, not all the fields in the structure are meaningful. The supplied initial values can be used for most fields, but the application must set the *StrucLength*, *StringLength*, and *String* fields to the values appropriate to the data.

The structure ends with a variable-length character string; see the *String* field section for further details.

See [“PCF structures: Usage notes” on page 1514](#) for further information about how to use the structure.

## **Fields for MQCFST**

### **Type (MQLONG)**

Structure type.

This indicates that the structure is an MQCFST structure describing a string parameter. The value must be:

#### **MQCFST\_STRING**

Structure defining a string.

### **StrucLength (MQLONG)**

Structure length.

This is the length in bytes of the MQCFST structure, including the string at the end of the structure (the *String* field). The length must be a multiple of four, and must be sufficient to contain the string; any bytes between the end of the string and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *String* field:

#### **MQCFST\_STRUC\_LENGTH\_FIXED**

Length of fixed part of command format string-parameter structure.

### **Parameter (MQLONG)**

Parameter identifier.

This identifies the parameter with a value that is contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see [“MQCFH - PCF header” on page 1515](#) for details.

The parameter is from the following groups of parameters:

- MQCA\_\*
- MQCACF\_\*
- MQCAMO\_\*
- MQCACH\_\*

### **CodedCharSetId (MQLONG)**

Coded character set identifier.

This specifies the coded character set identifier of the data in the *String* field. The following special value can be used:

## **MQCCSI\_DEFAULT**

Default character set identifier.

The string data is in the character set defined by the *CodedCharSetId* field in the MQ header structure that *precedes* the MQCFH structure, or by the *CodedCharSetId* field in the MQMD if the MQCFH structure is at the start of the message.

## **StringLength (MQLONG)**

Length of string.

This is the length in bytes of the data in the *String* field; it must be zero or greater. This length does not need to be a multiple of four.

## **String (MQCHAR x StringLength)**

String value.

This is the value of the parameter identified by the *Parameter* field:

- In MQFMT\_ADMIN command messages, if the specified string is shorter than the standard length of the parameter, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, it is an error.
- In MQFMT\_ADMIN response messages, string parameters might be returned padded with blanks to the standard length of the parameter.
- In MQFMT\_EVENT messages, trailing blanks might be omitted from string parameters (that is, the string can be shorter than the standard length of the parameter).

The value of *StringLength* depends on whether, when the specified string is shorter than the standard length, padding blanks have been added to the string. If so, the value of *StringLength* is the sum of the actual length of the string plus the padded blanks.

The string can contain any characters that are in the character set defined by *CodedCharSetId*, and that are valid for the parameter identified by *Parameter*.

**Note:** When the queue manager reads an MQCFST structure in an MQFMT\_ADMIN message from the command input queue, the queue manager processes the string as though it had been specified on an MQI call. This processing means that within the string, the first null and the characters following it (up to the end of the string) are treated as blanks.

In responses and all other cases, a null character in the string is treated as normal data, and does not act as a delimiter for the string. This treatment means that when a receiving application reads a MQFMT\_PCF, MQFMT\_EVENT, or MQFMT\_ADMIN message, the receiving application receives all the data specified by the sending application.

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure must be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL, PL/I, and System/390 assembler programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, the user must include MQCFST in a larger structure, and declare an additional field or additional fields following MQCFST, to represent the *String* field as required.

## **C language declaration**

```
typedef struct tagMQCFST {
    MQLONG  Type;          /* Structure type */
    MQLONG  StrucLength;   /* Structure length */
    MQLONG  Parameter;    /* Parameter identifier */
    MQLONG  CodedCharSetId; /* Coded character set identifier */
    MQLONG  StringLength; /* Length of string */
    MQCHAR  String[1];    /* String value - first
                          character */
} MQCFST;
```

## COBOL language declaration

```
**      MQCFST structure
10 MQCFST.
**      Structure type
15 MQCFST-TYPE          PIC S9(9) BINARY.
**      Structure length
15 MQCFST-STRUCLNGTH  PIC S9(9) BINARY.
**      Parameter identifier
15 MQCFST-PARAMETER    PIC S9(9) BINARY.
**      Coded character set identifier
15 MQCFST-CODEDCHARSETID PIC S9(9) BINARY.
**      Length of string
15 MQCFST-STRINGLENGTH PIC S9(9) BINARY.
```

## PL/I language declaration ( z/OS only)

```
dcl
  1 MQCFST based,
  3 Type          fixed bin(31), /* Structure type */
  3 StructLength  fixed bin(31), /* Structure length */
  3 Parameter     fixed bin(31), /* Parameter identifier */
  3 CodedCharSetId fixed bin(31), /* Coded character set identifier */
  3 StringLength  fixed bin(31); /* Length of string */
```

## System/390 assembler-language declaration ( z/OS only)

```
MQCFST          DSECT
MQCFST_TYPE     DS  F      Structure type
MQCFST_STRUCLNGTH DS  F      Structure length
MQCFST_PARAMETER DS  F      Parameter identifier
MQCFST_CODEDCHARSETID DS  F      Coded character set
*              identifier
MQCFST_STRINGLENGTH DS  F      Length of string
MQCFST_LENGTH   EQU *-MQCFST Length of structure
MQCFST_AREA     DS  CL(MQCFST_LENGTH)
```

## Visual Basic language declaration ( Windows only)

```
Type MQCFST
  Type As Long          ' Structure type
  StructLength As Long  ' Structure length
  Parameter As Long     ' Parameter identifier
  CodedCharSetId As Long ' Coded character set identifier
  StringLength As Long  ' Length of string
End Type

Global MQCFST_DEFAULT As MQCFST
```

## RPG language declaration ( IBM i only)

```
D* MQCFST Structure
D*
D* Structure type
D STYP          1      4I 0 INZ(4)
D* Structure length
D STLEN        5      8I 0 INZ(20)
D* Parameter identifier
D STPRM        9      12I 0 INZ(0)
D* Coded character set identifier
D STCSI       13      16I 0 INZ(0)
D* Length of string
D STSTL       17      20I 0 INZ(0)
D*
```

## PCF example

In this example the compiled program, written in C language, uses IBM MQ for Windows. The program inquires of the default queue manager about a subset of the attributes for all local queues defined to it. It then produces an output file, SAVEQMGR.TST, in the directory from which it was run for use with RUNMQSC.

## Inquire local queue attributes

This following section provides an example of how Programmable Command Formats can be used in a program for administration of IBM MQ queues.

The program is given as an example of using PCFs and has been limited to a simple case. This program is of most use as an example if you are considering the use of PCFs to manage your IBM MQ environment.

## Program listing

```
/*=====*/
/*
/* This is a program to inquire of the default queue manager about the
/* local queues defined to it.
/*
/* The program takes this information and appends it to a file
/* SAVEQMGR.TST which is of a format suitable for RUNMQSC. It could,
/* therefore, be used to re-create or clone a queue manager.
/*
/* It is offered as an example of using Programmable Command Formats (PCFs)
/* as a method for administering a queue manager.
/*
/*=====*/

/* Include standard libraries */
#include <memory.h>
#include <stdio.h>

/* Include MQSeries headers */
#include <cmqc.h>
#include <cmqcfc.h>
#include <cmqxc.h>

typedef struct LocalQParms {
    MQCHAR48    QName;
    MQLONG     QType;
    MQCHAR64    QDesc;
    MQLONG     InhibitPut;
    MQLONG     DefPriority;
    MQLONG     DefPersistence;
    MQLONG     InhibitGet;
    MQCHAR48    ProcessName;
    MQLONG     MaxQDepth;
    MQLONG     MaxMsgLength;
    MQLONG     BackoutThreshold;
    MQCHAR48    BackoutReqQName;
    MQLONG     Shareability;
    MQLONG     DefInputOpenOption;
    MQLONG     HardenGetBackout;
    MQLONG     MsgDeliverySequence;
    MQLONG     RetentionInterval;
    MQLONG     DefinitionType;
    MQLONG     Usage;
    MQLONG     OpenInputCount;
    MQLONG     OpenOutputCount;
    MQLONG     CurrentQDepth;
    MQCHAR12    CreationDate;
    MQCHAR8     CreationTime;
    MQCHAR48    InitiationQName;
    MQLONG     TriggerControl;
    MQLONG     TriggerType;
    MQLONG     TriggerMsgPriority;
    MQLONG     TriggerDepth;
    MQCHAR64    TriggerData;
    MQLONG     Scope;
    MQLONG     QDepthHighLimit;
    MQLONG     QDepthLowLimit;
```

```

    MQLONG      QDepthMaxEvent;
    MQLONG      QDepthHighEvent;
    MQLONG      QDepthLowEvent;
    MQLONG      QServiceInterval;
    MQLONG      QServiceIntervalEvent;
} LocalQParms;

MQOD  ObjDesc = { MQOD_DEFAULT };
MQMD  md      = { MQMD_DEFAULT };
MQPMO pmo     = { MQPMO_DEFAULT };
MQGMO gmo     = { MQGMO_DEFAULT };

void ProcessStringParm( MQCFST *pPCFString, LocalQParms *DefnLQ );
void ProcessIntegerParm( MQCFIN *pPCFInteger, LocalQParms *DefnLQ );
void AddToFileQLOCAL( LocalQParms DefnLQ );
void MQParmCpy( char *target, char *source, int length );

void PutMsg( MQHCONN  hConn      /* Connection to queue manager */
            , MQCHAR8  MsgFormat /* Format of user data to be put in msg */
            , MQHOBJ   hQName    /* handle of queue to put the message to */
            , MQCHAR48 QName     /* name of queue to put the message to */
            , MQBYTE   *UserMsg  /* The user data to be put in the message */
            , MQLONG   UserMsgLen /* */
            );

void GetMsg( MQHCONN  hConn      /* handle of queue manager */
            , MQLONG   MQParm    /* Options to specify nature of get */
            , MQHOBJ   hQName    /* handle of queue to read from */
            , MQBYTE   *UserMsg  /* Input/Output buffer containing msg */
            , MQLONG   ReadBufferLen /* Length of supplied buffer */
            );
MQHOBJ OpenQ( MQHCONN  hConn
            , MQCHAR48 QName
            , MQLONG   OpenOpts
            );

int main( int argc, char *argv[] )
{
    MQCHAR48      QMgrName;      /* Name of connected queue mgr */
    MQHCONN       hConn;        /* handle to connected queue mgr */
    MQOD          ObjDesc;      /* */
    MQLONG        OpenOpts;     /* */
    MQLONG        CompCode;     /* MQ API completion code */
    MQLONG        Reason;       /* Reason qualifying CompCode */
    /* */
    MQHOBJ        hAdminQ;      /* handle to output queue */
    MQHOBJ        hReplyQ;     /* handle to input queue */
    /* */
    MQLONG        AdminMsgLen;  /* Length of user message buffer */
    MQBYTE        *pAdminMsg;   /* Ptr to outbound data buffer */
    MQCFH         *pPCFHeader;  /* Ptr to PCF header structure */
    MQCFST        *pPCFString;  /* Ptr to PCF string parm block */
    MQCFIN        *pPCFInteger; /* Ptr to PCF integer parm block */
    MQLONG        *pPCFType;    /* Type field of PCF message parm */
    LocalQParms   DefnLQ;      /* */
    /* */
    char          ErrorReport[40]; /* */
    MQCHAR8       MsgFormat;     /* Format of inbound message */
    short         Index;         /* Loop counter */
    /* */

    /* Connect to default queue manager */
    QMgrName[0] = '\0'; /* set to null default QM */
    if ( argc > 1 )
        strcpy(QMgrName, argv[1]);

    MQCONN( QMgrName /* use default queue manager */
          , &hConn   /* queue manager handle */
          , &CompCode /* Completion code */
          , &Reason  /* Reason qualifying CompCode */
          );

    if ( CompCode != MQCC_OK ) {
        printf( "MQCONN failed for %s, CC=%d RC=%d\n"
              , QMgrName
              , CompCode
              , Reason
              );
        exit( -1 );
    }
} /* endif */

```

```

/* Open all the required queues */
hAdminQ = OpenQ( hConn, "SYSTEM.ADMIN.COMMAND.QUEUE\0", MQOO_OUTPUT );

hReplyQ = OpenQ( hConn, "SAVEQMR.REPLY.QUEUE\0", MQOO_INPUT_EXCLUSIVE );

/* ***** */
/* Put a message to the SYSTEM.ADMIN.COMMAND.QUEUE to inquire all */
/* the local queues defined on the queue manager. */
/* */
/* The request consists of a Request Header and a parameter block */
/* used to specify the generic search. The header and the parameter */
/* block follow each other in a contiguous buffer which is pointed */
/* to by the variable pAdminMsg. This entire buffer is then put to */
/* the queue. */
/* */
/* The command server, (use STRMQCSV to start it), processes the */
/* SYSTEM.ADMIN.COMMAND.QUEUE and puts a reply on the application */
/* ReplyToQ for each defined queue. */
/* ***** */

/* Set the length for the message buffer */
AdminMsgLen = MQCFH_STRUC_LENGTH
             + MQCFST_STRUC_LENGTH_FIXED + MQ_Q_NAME_LENGTH
             + MQCFIN_STRUC_LENGTH
             ;

/* ----- */
/* Set pointers to message data buffers */
/* */
/* pAdminMsg points to the start of the message buffer */
/* */
/* pPCFHeader also points to the start of the message buffer. It is */
/* used to indicate the type of command we wish to execute and the */
/* number of parameter blocks following in the message buffer. */
/* */
/* pPCFString points into the message buffer immediately after the */
/* header and is used to map the following bytes onto a PCF string */
/* parameter block. In this case the string is used to indicate the */
/* name of the queue we want details about, * indicating all queues. */
/* */
/* pPCFInteger points into the message buffer immediately after the */
/* string block described above. It is used to map the following */
/* bytes onto a PCF integer parameter block. This block indicates */
/* the type of queue we wish to receive details about, thereby */
/* qualifying the generic search set up by passing the previous */
/* string parameter. */
/* */
/* Note that this example is a generic search for all attributes of */
/* all local queues known to the queue manager. By using different, */
/* or more, parameter blocks in the request header it is possible */
/* to narrow the search. */
/* ----- */

pAdminMsg = (MQBYTE *)malloc( AdminMsgLen );

pPCFHeader = (MQCFH *)pAdminMsg;

pPCFString = (MQCFST *) (pAdminMsg
                       + MQCFH_STRUC_LENGTH
                       );

pPCFInteger = (MQCFIN *) ( pAdminMsg
                          + MQCFH_STRUC_LENGTH
                          + MQCFST_STRUC_LENGTH_FIXED + MQ_Q_NAME_LENGTH
                          );

/* Set up request header */
pPCFHeader->Type = MQCFT_COMMAND;
pPCFHeader->StrucLength = MQCFH_STRUC_LENGTH;
pPCFHeader->Version = MQCFH_VERSION_1;
pPCFHeader->Command = MQCMD_INQUIRE_Q;
pPCFHeader->MsgSeqNumber = MQCFC_LAST;
pPCFHeader->Control = MQCFC_LAST;
pPCFHeader->ParameterCount = 2;

/* Set up parameter block */
pPCFString->Type = MQCFT_STRING;
pPCFString->StrucLength = MQCFST_STRUC_LENGTH_FIXED + MQ_Q_NAME_LENGTH;
pPCFString->Parameter = MQCA_Q_NAME;
pPCFString->CodedCharSetId = MQCCSI_DEFAULT;

```

```

pPCFString->StringLength = 1;
memcpy( pPCFString->String, "*", 1 );

/* Set up parameter block */
pPCFInteger->Type = MQCFI_INTEGER;
pPCFInteger->StrucLength = MQCFIN_STRUC_LENGTH;
pPCFInteger->Parameter = MQIA_Q_TYPE;
pPCFInteger->Value = MQQT_LOCAL;

PutMsg( hConn /* Queue manager handle */
, MQFMT_ADMIN /* Format of message */
, hAdminQ /* Handle of command queue */
, "SAVEQMGR.REPLY.QUEUE\0" /* reply to queue */
, (MQBYTE *)pAdminMsg /* Data part of message to put */
, AdminMsgLen
);

free( pAdminMsg );

/* ***** */
/* Get and process the replies received from the command server onto */
/* the applications ReplyToQ. */
/* */
/* There will be one message per defined local queue. */
/* */
/* The last message will have the Control field of the PCF header */
/* set to MQCFC_LAST. All others will be MQCFC_NOT_LAST. */
/* */
/* An individual Reply message consists of a header followed by a */
/* number a parameters, the exact number, type and order will depend */
/* upon the type of request. */
/* ----- */
/* The message is retrieved into a buffer pointed to by pAdminMsg. */
/* This buffer has been allocated enough memory to hold every */
/* parameter needed for a local queue definition. */
/* */
/* pPCFHeader is then allocated to point also to the beginning of */
/* the buffer and is used to access the PCF header structure. The */
/* header contains several fields. The one we are specifically */
/* interested in is the ParameterCount. This tells us how many */
/* parameters follow the header in the message buffer. There is */
/* one parameter for each local queue attribute known by the */
/* queue manager. */
/* */
/* At this point we do not know the order or type of each parameter */
/* block in the buffer, the first MQLONG of each block defines its */
/* type; they may be parameter blocks containing either strings or */
/* integers. */
/* */
/* pPCFType is used initially to point to the first byte beyond the */
/* known parameter block. Initially then, it points to the first byte */
/* after the PCF header. Subsequently it is incremented by the length */
/* of the identified parameter block and therefore points at the */
/* next. Looking at the value of the data pointed to by pPCFType we */
/* can decide how to process the next group of bytes, either as a */
/* string, or an integer. */
/* */
/* In this way we parse the message buffer extracting the values of */
/* each of the parameters we are interested in. */
/* */
/* ***** */

/* AdminMsgLen is to be set to the length of the expected reply */
/* message. This structure is specific to Local Queues. */
AdminMsgLen = MQCFH_STRUC_LENGTH
+ ( MQCFST_STRUC_LENGTH_FIXED * 7 )
+ ( MQCFIN_STRUC_LENGTH * 39 )
+ ( MQ_Q_NAME_LENGTH * 6 )
+ ( MQ_Q_MGR_NAME_LENGTH * 2 )
+ MQ_Q_DESC_LENGTH
+ MQ_PROCESS_NAME_LENGTH
+ MQ_CREATION_DATE_LENGTH
+ MQ_CREATION_TIME_LENGTH
+ MQ_TRIGGER_DATA_LENGTH + 100
;

/* Set pointers to message data buffers */
pAdminMsg = (MQBYTE *)malloc( AdminMsgLen );

do {

```



```

GetMsg( hConn          /* Queue manager handle          */
      , MQGMO_WAIT
      , hReplyQ        /* Get queue handle          */
      , (MQBYTE *)pAdminMsg /* pointer to message area  */
      , AdminMsgLen    /* length of get buffer     */
      );

/* Examine Header */
pPCFHeader = (MQCFH *)pAdminMsg;

/* Examine first parameter */
pPCFType = (MQLONG *) (pAdminMsg + MQCFH_STRUC_LENGTH);

Index = 1;

while ( Index <= pPCFHeader->ParameterCount ) {

    /* Establish the type of each parameter and allocate */
    /* a pointer of the correct type to reference it.    */
    switch ( *pPCFType ) {
    case MQCFT_INTEGER:
        pPCFInteger = (MQCFIN *)pPCFType;
        ProcessIntegerParm( pPCFInteger, &DefnLQ );
        Index++;
        /* Increment the pointer to the next parameter by the */
        /* length of the current parm.                          */
        pPCFType = (MQLONG *) ( (MQBYTE *)pPCFType
                                + pPCFInteger->StrucLength
                                );
        break;
    case MQCFT_STRING:
        pPCFString = (MQCFST *)pPCFType;
        ProcessStringParm( pPCFString, &DefnLQ );
        Index++;
        /* Increment the pointer to the next parameter by the */
        /* length of the current parm.                          */
        pPCFType = (MQLONG *) ( (MQBYTE *)pPCFType
                                + pPCFString->StrucLength
                                );
        break;
    } /* endswitch */

} /* endwhile */

/* ***** */
/* Message parsed, append to output file */
/* ***** */
AddToFileQLOCAL( DefnLQ );

/* ***** */
/* Finished processing the current message, do the next one. */
/* ***** */

} while ( pPCFHeader->Control == MQCFC_NOT_LAST ); /* enddo */

free( pAdminMsg );

/* ***** */
/* Processing of the local queues complete */
/* ***** */

}

void ProcessStringParm( MQCFST *pPCFString, LocalQParms *DefnLQ )
{
    switch ( pPCFString->Parameter ) {
    case MQCA_Q_NAME:
        MQParmCpy( DefnLQ->QName, pPCFString->String, 48 );
        break;
    case MQCA_Q_DESC:
        MQParmCpy( DefnLQ->QDesc, pPCFString->String, 64 );
        break;
    case MQCA_PROCESS_NAME:
        MQParmCpy( DefnLQ->ProcessName, pPCFString->String, 48 );
        break;
    case MQCA_BACKOUT_REQ_Q_NAME:
        MQParmCpy( DefnLQ->BackoutReqQName, pPCFString->String, 48 );
        break;
    case MQCA_CREATION_DATE:
        MQParmCpy( DefnLQ->CreationDate, pPCFString->String, 12 );

```

```

        break;
    case MQCA_CREATION_TIME:
        MQParmCpy( DefnLQ->CreationTime, pPCFString->String, 8 );
        break;
    case MQCA_INITIATION_Q_NAME:
        MQParmCpy( DefnLQ->InitiationQName, pPCFString->String, 48 );
        break;
    case MQCA_TRIGGER_DATA:
        MQParmCpy( DefnLQ->TriggerData, pPCFString->String, 64 );
        break;
} /* endswitch */
}

void ProcessIntegerParm( MQCFIN *pPCFInteger, LocalQParms *DefnLQ )
{
    switch ( pPCFInteger->Parameter ) {
    case MQIA_Q_TYPE:
        DefnLQ->QType = pPCFInteger->Value;
        break;
    case MQIA_INHIBIT_PUT:
        DefnLQ->InhibitPut = pPCFInteger->Value;
        break;
    case MQIA_DEF_PRIORITY:
        DefnLQ->DefPriority = pPCFInteger->Value;
        break;
    case MQIA_DEF_PERSISTENCE:
        DefnLQ->DefPersistence = pPCFInteger->Value;
        break;
    case MQIA_INHIBIT_GET:
        DefnLQ->InhibitGet = pPCFInteger->Value;
        break;
    case MQIA_SCOPE:
        DefnLQ->Scope = pPCFInteger->Value;
        break;
    case MQIA_MAX_Q_DEPTH:
        DefnLQ->MaxQDepth = pPCFInteger->Value;
        break;
    case MQIA_MAX_MSG_LENGTH:
        DefnLQ->MaxMsgLength = pPCFInteger->Value;
        break;
    case MQIA_BACKOUT_THRESHOLD:
        DefnLQ->BackoutThreshold = pPCFInteger->Value;
        break;
    case MQIA_SHAREABILITY:
        DefnLQ->Shareability = pPCFInteger->Value;
        break;
    case MQIA_DEF_INPUT_OPEN_OPTION:
        DefnLQ->DefInputOpenOption = pPCFInteger->Value;
        break;
    case MQIA_HARDEN_GET_BACKOUT:
        DefnLQ->HardenGetBackout = pPCFInteger->Value;
        break;
    case MQIA_MSG_DELIVERY_SEQUENCE:
        DefnLQ->MsgDeliverySequence = pPCFInteger->Value;
        break;
    case MQIA_RETENTION_INTERVAL:
        DefnLQ->RetentionInterval = pPCFInteger->Value;
        break;
    case MQIA_DEFINITION_TYPE:
        DefnLQ->DefinitionType = pPCFInteger->Value;
        break;
    case MQIA_USAGE:
        DefnLQ->Usage = pPCFInteger->Value;
        break;
    case MQIA_OPEN_INPUT_COUNT:
        DefnLQ->OpenInputCount = pPCFInteger->Value;
        break;
    case MQIA_OPEN_OUTPUT_COUNT:
        DefnLQ->OpenOutputCount = pPCFInteger->Value;
        break;
    case MQIA_CURRENT_Q_DEPTH:
        DefnLQ->CurrentQDepth = pPCFInteger->Value;
        break;
    case MQIA_TRIGGER_CONTROL:
        DefnLQ->TriggerControl = pPCFInteger->Value;
        break;
    case MQIA_TRIGGER_TYPE:
        DefnLQ->TriggerType = pPCFInteger->Value;
        break;
    case MQIA_TRIGGER_MSG_PRIORITY:
        DefnLQ->TriggerMsgPriority = pPCFInteger->Value;
        break;
    }
}

```

```

case MQIA_TRIGGER_DEPTH:
    DefnLQ->TriggerDepth = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_HIGH_LIMIT:
    DefnLQ->QDepthHighLimit = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_LOW_LIMIT:
    DefnLQ->QDepthLowLimit = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_MAX_EVENT:
    DefnLQ->QDepthMaxEvent = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_HIGH_EVENT:
    DefnLQ->QDepthHighEvent = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_LOW_EVENT:
    DefnLQ->QDepthLowEvent = pPCFInteger->Value;
    break;
case MQIA_Q_SERVICE_INTERVAL:
    DefnLQ->QServiceInterval = pPCFInteger->Value;
    break;
case MQIA_Q_SERVICE_INTERVAL_EVENT:
    DefnLQ->QServiceIntervalEvent = pPCFInteger->Value;
    break;
} /* endswitch */
}

/* ----- */
/* This process takes the attributes of a single local queue and adds them */
/* to the end of a file, SAVEQMGR.TST, which can be found in the current */
/* directory. */
/* The file is of a format suitable for subsequent input to RUNMQSC. */
/* ----- */
void AddToFileQLOCAL( LocalQParms DefnLQ )
{
    char    ParmBuffer[120]; /* Temporary buffer to hold for output to file */
    FILE    *fp;             /* Pointer to a file */

    /* Append these details to the end of the current SAVEQMGR.TST file */
    fp = fopen( "SAVEQMGR.TST", "a" );

    sprintf( ParmBuffer, "DEFINE QLOCAL ('%s') REPLACE +\n", DefnLQ.QName );
    fputs( ParmBuffer, fp );

    sprintf( ParmBuffer, "        DESCR('%s') +\n" , DefnLQ.QDesc );
    fputs( ParmBuffer, fp );

    if ( DefnLQ.InhibitPut == MQQA_PUT_ALLOWED ) {
        sprintf( ParmBuffer, "        PUT(ENABLED) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "        PUT(DISABLED) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    sprintf( ParmBuffer, "        DEFPRTY(%d) +\n", DefnLQ.DefPriority );
    fputs( ParmBuffer, fp );

    if ( DefnLQ.DefPersistence == MQPER_PERSISTENT ) {
        sprintf( ParmBuffer, "        DEFPSIST(YES) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "        DEFPSIST(NO) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    if ( DefnLQ.InhibitGet == MQQA_GET_ALLOWED ) {
        sprintf( ParmBuffer, "        GET(ENABLED) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "        GET(DISABLED) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    sprintf( ParmBuffer, "        MAXDEPTH(%d) +\n", DefnLQ.MaxQDepth );
    fputs( ParmBuffer, fp );

    sprintf( ParmBuffer, "        MAXMSGL(%d) +\n", DefnLQ.MaxMsgLength );
    fputs( ParmBuffer, fp );
}

```

```

if ( DefnLQ.Shareability == MQQA_SHAREABLE ) {
    sprintf( ParmBuffer, "      SHARE +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      NOSHARE +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.DefInputOpenOption == MQ00_INPUT_SHARED ) {
    sprintf( ParmBuffer, "      DEFSOPT(SHARED) +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      DEFSOPT(EXCL) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.MsgDeliverySequence == MQMDS_PRIORITY ) {
    sprintf( ParmBuffer, "      MSGDLVSQ(PRIORITY) +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      MSGDLVSQ(FIFO) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.HardenGetBackout == MQQA_BACKOUT_HARDENED ) {
    sprintf( ParmBuffer, "      HARDENBO +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      NOHARDENBO +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.Usage == MQUS_NORMAL ) {
    sprintf( ParmBuffer, "      USAGE(NORMAL) +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      USAGE(XMIT) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.TriggerControl == MQTC_OFF ) {
    sprintf( ParmBuffer, "      NOTRIGGER +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      TRIGGER +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

switch ( DefnLQ.TriggerType ) {
case MQTT_NONE:
    sprintf( ParmBuffer, "      TRIGTYPE(NONE) +\n" );
    fputs( ParmBuffer, fp );
    break;
case MQTT_FIRST:
    sprintf( ParmBuffer, "      TRIGTYPE(FIRST) +\n" );
    fputs( ParmBuffer, fp );
    break;
case MQTT EVERY:
    sprintf( ParmBuffer, "      TRIGTYPE(EVERY) +\n" );
    fputs( ParmBuffer, fp );
    break;
case MQTT_DEPTH:
    sprintf( ParmBuffer, "      TRIGTYPE(DEPTH) +\n" );
    fputs( ParmBuffer, fp );
    break;
} /* endswitch */

sprintf( ParmBuffer, "      TRIGDPATH(%d) +\n", DefnLQ.TriggerDepth );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      TRIGMPRI(%d) +\n", DefnLQ.TriggerMsgPriority);
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      TRIGDATA('%s') +\n", DefnLQ.TriggerData );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      PROCESS('%s') +\n", DefnLQ.ProcessName );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      INITQ('%s') +\n", DefnLQ.InitiationQName );
fputs( ParmBuffer, fp );

```

```

printf( ParmBuffer, "          RETINTVL(%d) +\n", DefnLQ.RetentionInterval );
fputs( ParmBuffer, fp );

printf( ParmBuffer, "          BOTHRESH(%d) +\n", DefnLQ.BackoutThreshold );
fputs( ParmBuffer, fp );

printf( ParmBuffer, "          BOQNAME('%s') +\n", DefnLQ.BackoutReqQName );
fputs( ParmBuffer, fp );

if ( DefnLQ.Scope == MQSCO_Q_MGR ) {
    printf( ParmBuffer, "          SCOPE(QMGR) +\n" );
    fputs( ParmBuffer, fp );
} else {
    printf( ParmBuffer, "          SCOPE(CELL) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

printf( ParmBuffer, "          QDEPTHHI(%d) +\n", DefnLQ.QDepthHighLimit );
fputs( ParmBuffer, fp );

printf( ParmBuffer, "          QDEPTHLO(%d) +\n", DefnLQ.QDepthLowLimit );
fputs( ParmBuffer, fp );

if ( DefnLQ.QDepthMaxEvent == MQEVR_ENABLED ) {
    printf( ParmBuffer, "          QDPMAXEV(ENABLED) +\n" );
    fputs( ParmBuffer, fp );
} else {
    printf( ParmBuffer, "          QDPMAXEV(DISABLED) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.QDepthHighEvent == MQEVR_ENABLED ) {
    printf( ParmBuffer, "          QDPHIEV(ENABLED) +\n" );
    fputs( ParmBuffer, fp );
} else {
    printf( ParmBuffer, "          QDPHIEV(DISABLED) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.QDepthLowEvent == MQEVR_ENABLED ) {
    printf( ParmBuffer, "          QDPLOEV(ENABLED) +\n" );
    fputs( ParmBuffer, fp );
} else {
    printf( ParmBuffer, "          QDPLOEV(DISABLED) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

printf( ParmBuffer, "          QSVCINT(%d) +\n", DefnLQ.QServiceInterval );
fputs( ParmBuffer, fp );

switch ( DefnLQ.QServiceIntervalEvent ) {
case MQQSIE_OK:
    printf( ParmBuffer, "          QSVCI EV(OK)\n" );
    fputs( ParmBuffer, fp );
    break;
case MQQSIE_NONE:
    printf( ParmBuffer, "          QSVCI EV(NONE)\n" );
    fputs( ParmBuffer, fp );
    break;
case MQQSIE_HIGH:
    printf( ParmBuffer, "          QSVCI EV(HIGH)\n" );
    fputs( ParmBuffer, fp );
    break;
} /* endswitch */

printf( ParmBuffer, "\n" );
fputs( ParmBuffer, fp );

fclose(fp);
}

/* ----- */
/*
/* The queue manager returns strings of the maximum length for each
/* specific parameter, padded with blanks.
/*
/* We are interested in only the non-blank characters so will extract them
/* from the message buffer, and terminate the string with a null, \0.
/*
/*

```

```

/* ----- */
void MQParmCpy( char *target, char *source, int length )
{
    int counter=0;

    while ( counter < length && source[counter] != ' ' ) {
        target[counter] = source[counter];
        counter++;
    } /* endwhile */

    if ( counter < length) {
        target[counter] = '\0';
    } /* endif */
}

MQHOBJ OpenQ( MQHCONN hConn, MQCHAR48 QName, MQLONG OpenOpts)
{
    MQHOBJ Hobj;
    MQLONG CompCode, Reason;

    ObjDesc.ObjectType = MQOT_Q;
    strncpy(ObjDesc.ObjectName, QName, MQ_Q_NAME_LENGTH);

    MQOPEN(hConn, /* connection handle */
           &ObjDesc, /* object descriptor for queue */
           OpenOpts, /* open options */
           &Hobj, /* object handle */
           &CompCode, /* MQOPEN completion code */
           &Reason); /* reason code */

    /* report reason, if any; stop if failed */
    if (Reason != MQRC_NONE)
    {
        printf("MQOPEN for %s ended with Reason Code %d and Comp Code %d\n",
               QName,
               Reason,
               CompCode);

        exit( -1 );
    }

    return Hobj;
}

void PutMsg(MQHCONN hConn,
            MQCHAR8 MsgFormat,
            MQHOBJ hQName,
            MQCHAR48 QName,
            MQBYTE *UserMsg,
            MQLONG UserMsgLen)
{
    MQLONG CompCode, Reason;

    /* set up the message descriptor prior to putting the message */
    md.Report = MQRO_NONE;
    md.MsgType = MQMT_REQUEST;
    md.Expiry = MQEI_UNLIMITED;
    md.Feedback = MQFB_NONE;
    md.Encoding = MQENC_NATIVE;
    md.Priority = MQPRI_PRIORITY_AS_Q_DEF;
    md.Persistence = MQPER_PERSISTENCE_AS_Q_DEF;
    md.MsgSeqNumber = 1;
    md.Offset = 0;
    md.MsgFlags = MQMF_NONE;
    md.OriginalLength = MQOL_UNDEFINED;

    memcpy(md.GroupId, MQGI_NONE, sizeof(md.GroupId));
    memcpy(md.Format, MsgFormat, sizeof(md.Format));
    memcpy(md.ReplyToQ, QName, sizeof(md.ReplyToQ));

    /* reset MsgId and CorrelId to get a new one */
    memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
    memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));

    MQPUT(hConn, /* connection handle */
          hQName, /* object handle */
          &md, /* message descriptor */
          &pmo, /* default options */
          UserMsgLen, /* message length */
          (MQBYTE *)UserMsg, /* message buffer */
          &CompCode, /* completion code */
          &Reason); /* reason code */
}

```

```

    if (Reason != MQRC_NONE) {
        printf("MQPUT ended with Reason Code %d and Comp Code %d\n",
            Reason, CompCode);
        exit( -1 );
    }
}

void GetMsg(MQHCONN hConn, MQLONG MQParm, MQHOBJ hQName,
           MQBYTE *UserMsg, MQLONG ReadBufferLen)
{
    MQLONG CompCode, Reason, msglen;

    gmo.Options      = MQParm;
    gmo.WaitInterval = 15000;

    /* reset MsgId and CorrelId to get a new one */
    memcpy(md.MsgId,    MQMI_NONE, sizeof(md.MsgId) );
    memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId) );

    MQGET(hConn,          /* connection handle */
          hQName,         /* object handle */
          &md,           /* message descriptor */
          &gmo,          /* get message options */
          ReadBufferLen, /* Buffer length */
          (MQBYTE *)UserMsg, /* message buffer */
          &msglen,       /* message length */
          &CompCode,     /* completion code */
          &Reason);     /* reason code */

    if (Reason != MQRC_NONE) {
        printf("MQGET ended with Reason Code %d and Comp Code %d\n",
            Reason, CompCode);
        exit( -1 );
    }
}

```

## IBM i CL commands for IBM i reference

A list of CL commands for IBM i, grouped according to command type.

- Authentication Information Commands
  - [CHGMQMAUTI](#), Change IBM MQ Authentication Information
  - [CPYMQMAUTI](#), Copy IBM MQ Authentication Information
  - [CRTMQMAUTI](#), Create IBM MQ Authentication Information
  - [DLTMQMAUTI](#), Delete IBM MQ Authentication Information
  - [DSPMQMAUTI](#), Display IBM MQ Authentication Information
  - [WRKMQMAUTI](#), Work with IBM MQ Authentication Information
- Authority Commands
  - [DSPMQMAUT](#), Display IBM MQ Object Authority
  - [GRTMQMAUT](#), Grant IBM MQ Object Authority
  - [RFRMQMAUT](#), Refresh IBM MQ Object Authority
  - [RVKMQMAUT](#), Revoke IBM MQ Object Authority
  - [WRKMQMAUT](#), Work with IBM MQ Authority
  - [WRKMQMAUTD](#), Work with IBM MQ Authority Data
- Broker Commands

The following commands do not perform any function and are only provided for compatibility with previous releases of IBM MQ.

- [CLRMQMBRK](#), Clear IBM MQ Broker
- [DLTMQMBRK](#), Delete IBM MQ Broker
- [DSPMQMBRK](#), Display IBM MQ Pub/Sub Broker
- [DSPMQMBRK](#), Display IBM MQ Broker

- [ENDMQMBRK, End IBM MQ Broker](#)
- [STRMQMBRK, Start IBM MQ Broker](#)
- Channel Commands
  - [CHGMQMCHL, Change IBM MQ Channel](#)
  - [CPYMQMCHL, Copy IBM MQ Channel](#)
  - [CRTMQMCHL, Create IBM MQ Channel](#)
  - [DLTMQMCHL, Delete IBM MQ Channel](#)
  - [DSPMQMCHL, Display IBM MQ Channel](#)
  - [ENDMQMCHL, End IBM MQ Channel](#)
  - [PNGMQMCHL, Ping IBM MQ Channel](#)
  - [RSTMQMCHL, Reset IBM MQ Channel](#)
  - [RSVMQMCHL, Resolve IBM MQ Channel](#)
  - [STRMQMCHL, Start IBM MQ Channel](#)
  - [STRMQMCHLI, Start IBM MQ Channel Initiator](#)
  - [WRKMQMCHL, Work with IBM MQ Channels](#)
  - [WRKMQMCHST, Work with IBM MQ Channel Status](#)
- Cluster Commands
  - [RFRMQMCL, Refresh IBM MQ Cluster](#)
  - [RSMMQMCLQM, Resume IBM MQ Cluster Queue Manager](#)
  - [RSTMQMCL, Reset IBM MQ Cluster](#)
  - [SPDMQMCLQM, Suspend IBM MQ Cluster Queue Manager](#)
  - [WRKMQMCL, Work with IBM MQ Clusters](#)
  - [WRKMQMCLQ, Work with IBM MQ Cluster Queues](#)
- Command Server Commands
  - [DSPMQMCSVR, Display IBM MQ Command Server](#)
  - [ENDMQMCSVR, End IBM MQ Command Server](#)
  - [STRMQMCSVR, Start IBM MQ Command Server](#)
- Connection Commands
  - [ENDMQMCONN, End IBM MQ Connection](#)
  - [WRKMQMCONN, Work with IBM MQ Connections](#)
- Data Conversion Exit Command
  - [CVTMQMDTA, Convert IBM MQ Data Type](#)
- Listener Commands
  - [CHGMQMLSR, Change IBM MQ Listener Object](#)
  - [CPYMQMLSR, Copy IBM MQ Listener Object](#)
  - [CRTMQMLSR, Create IBM MQ Listener Object](#)
  - [DLTMQMLSR, Delete IBM MQ Listener Object](#)
  - [DSPMQMLSR, Display IBM MQ Listener Object](#)
  - [ENDMQMLSR, End IBM MQ Listener](#)
  - [STRMQMLSR, Start IBM MQ Listener](#)
  - [WRKMQMLSR, Work with IBM MQ Listeners](#)
- Media Recovery Commands



- [RCDMQMIMG, Record IBM MQ Object Image](#)
- [RCRMQMOBJ, Re-create IBM MQ Object](#)
- [WRKMQMTRN, Work with IBM MQ Transactions](#)
- Name Command
  - [DSPMQMOBJN, Display IBM MQ Object Names](#)
- Namelist Commands
  - [CHGMQMNL, Change IBM MQ Namelist](#)
  - [CPYMQMNL, Copy IBM MQ Namelist](#)
  - [CRTMQMNL, Create IBM MQ Namelist](#)
  - [DLTMQMNL, Delete IBM MQ Namelist](#)
  - [DSPMQMNL, Display IBM MQ Namelist](#)
  - [WRKMQMNL, Work with IBM MQ Namelists](#)
- Process Commands
  - [CHGMQMPCR, Change IBM MQ Process](#)
  - [CPYMQMPCR, Copy IBM MQ Process](#)
  - [CRTMQMPCR, Create IBM MQ Process](#)
  - [DLTMQMPCR, Delete IBM MQ Process](#)
  - [DSPMQMPCR, Display IBM MQ Process](#)
  - [WRKMQMPCR, Work with IBM MQ Processes](#)
- Queue Commands
  - [CHGMQMQ, Change IBM MQ Queue](#)
  - [CLRMQMQ, Clear IBM MQ Queue](#)
  - [CPYMQMQ, Copy IBM MQ Queue](#)
  - [CRTMQMQ, Create IBM MQ Queue](#)
  - [DLTMQMQ, Delete IBM MQ Queue](#)
  - [DSPMQMQ, Display IBM MQ Queue](#)
  - [WRKMQMMSG, Work with IBM MQ Messages](#)
  - [WRKMQMQ, Work with IBM MQ Queues](#)
  - [WRKMQMQSTS, Work with IBM MQ Queue Status](#)
- Queue Manager Commands
  - [CCTMQM, Connect to Message Queue Manager](#)
  - [CHGMQM, Change Message Queue Manager](#)
  - [CRTMQM, Create Message Queue Manager](#)
  - [DLTMQM, Delete Message Queue Manager](#)
  - [DSCMQM, Disconnect from Message Queue Manager](#)
  - [DSPMQM, Display Message Queue Manager](#)
  - [DSPMQMSTS, Display Message Queue Manager Status](#)
  - [ENDMQM, End Message Queue Manager](#)
  - [RFRMQM, Refresh Message Queue Manager](#)
  - [STRMQM, Start Message Queue Manager](#)
  - [STRMQMTRM, Start IBM MQ Trigger Monitor](#)
  - [WRKMQM, Work with Message Queue Manager](#)
- Service Commands

- [CHGMQMSVC, Change IBM MQ Service](#)
- [CPYMQMSVC, Copy IBM MQ Service](#)
- [CRTMQMSVC, Create IBM MQ Service](#)
- [DLTMQMSVC, Delete IBM MQ Service](#)
- [DSPMQMSVC, Display IBM MQ Service](#)
- [ENDMQMSVC, End IBM MQ Service](#)
- [STRMQMSVC, Start IBM MQ Service](#)
- [WRKMQMSVC, Work with IBM MQ Services](#)
- Subscription Commands
  - [CHGMQMSUB, Change IBM MQ Subscription](#)
  - [CPYMQMSUB, Copy IBM MQ Subscription](#)
  - [CRTMQMSUB, Create IBM MQ Subscription](#)
  - [DLTMQMSUB, Delete IBM MQ Subscription](#)
  - [DSPMQMSUB, Display IBM MQ Subscription](#)
  - [WRKMQMSUB, Work with IBM MQ Subscription](#)
- Topic Commands
  - [CHGMQMTOP, Change IBM MQ Topic](#)
  - [CLRMQMTOP, Clear IBM MQ Topic](#)
  - [CPYMQMTOP, Copy IBM MQ Topic](#)
  - [CRTMQMTOP, Create IBM MQ Topic](#)
  - [DLTMQMTOP, Delete IBM MQ Topic](#)
  - [DSPMQMTOP, Display IBM MQ Topic](#)
  - [WRKMQMTOP, Work with IBM MQ Topics](#)
- Trace Command
  - [TRCMQM, Trace IBM MQ Job](#)
- IBM MQSC Commands
  - [RUNMQSC, Run IBM MQSC Commands](#)
  - [STRMQMMQSC, Start IBM MQSC Commands](#)
- IBM MQ Dead-Letter Queue Handler Command
  - [STRMQMDLQ, Start IBM MQ Dead-Letter Queue Handler](#)
- IBM MQ Route Information
  - [DSPMQMRTE, Display IBM MQ Route Information](#)
- IBM MQ Configuration Dump
  - [Dump MQ Configuration \(DMPMQMCFG\)](#)
- IBM MQ Version Details
  - [DSPMQMVER, Display IBM MQ Version](#)

### Related tasks

[Managing IBM MQ for IBM i using CL commands](#)

## **ADDMQMINF (Add Queue Manager Information)**

### Where allowed to run

All environments (\*ALL)

## Threadsafe

Yes

The Add Message Queue Manager Information (ADDMQMINF) command adds configuration information for a queue manager. This command may be used, for example, to create a secondary queue manager instance by adding a reference to shared queue manager data.

## Parameters

Table 217. Command parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1
<u>PREFIX</u>	Queue Manager Prefix	Character value	Required, Positional 2
<u>MQMDIR</u>	Queue Manager Directory	Character value	Required, Positional 3
<u>MQMLIB</u>	Queue Manager Library	Name	Required, Positional 4
<u>DATAPATH</u>	Queue Manager Data Path	Character value	Optional, Positional 5

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager to add information for.

#### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### Queue Manager Prefix (PREFIX)

Specifies the prefix for the queue manager filesystem, for example, '/QIBM/UserData/mqm'

The possible values are:

#### queue-manager-directory-prefix

The prefix for the queue manager filesystem.

### Queue Manager Directory (MQMDIR)

Specifies the directory name for the queue manager filesystem. In most cases this will be the same as the queue manager name, unless the directory name has been modified to cater for characters that are not allowed in directory names, or to avoid a clash with an existing directory name.

The possible values are:

#### queue-manager-directory-name

The prefix for the queue manager filesystem. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### Queue Manager Library (MQMLIB)

Specifies the library to be used by the queue manager.

The possible values are:

#### library name

Specify the library to be used by the queue manager.

## Queue Manager Data Path (DATAPATH)

Specifies the fully qualified directory path for the queue manager data. This parameter is optional and if specified, overrides the prefix and directory name for the queue managers data files. Typically this parameter could be used to reference queue data stored on a networked filesystem, such as NFSv4.

The possible values are:

### queue-manager-data-path

Specify the data path to be used by the queue manager.

## ADDMQMJRN (Add Queue Manager Journal)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Add Queue Manager Journals command (ADDMQMJRN) adds a journal to a queue manager. This command can be used, for example, to configure remote journal replication for a backup or multi-instance queue manager.

## Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 1
<u>JRN</u>	Queue Manager Journal	Character value, *DFT	Optional, Positional 2
<u>RMTJRNRDB</u>	Remote Relational Database	Character value	Optional, Positional 3
<u>RMTJRNSTS</u>	Remote Journal Status	*ACTIVE, *INACTIVE	Optional, Positional 4
<u>RMTJRNDLV</u>	Remote Journal Delivery	*SYNC, *ASYNC	Optional, Positional 5
<u>RMTJRNTIMO</u>	Remote Journal Sync. Timeout	1-3600, *DFT	Optional, Positional 6

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager associated with the journal.

### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

## Queue Manager Journal (JRN)

Specifies the journal name to create.

The possible values are:

### \*DFT

The journal name is chosen by the system. If a local journal already exists for the queue manager on this system - the existing local journal name is used, otherwise a unique name is generated of the format AMQxJRN where x is a character in the range 'A - Z'.

**journal-name**

Specify the name of the journal. The name can contain up to 10 characters. Journal receiver names will be derived from this journal name by truncating at the 4th character (or at the last character if the journal name is shorter than 4 characters) and appending zeroes. If the local queue manager library already contains a local journal, its name must match that supplied. Only one local journal can exist in a queue manager library. DLTMQM will not remove journal artifacts from a queue manager library unless they are prefixed with "AMQ".

**Remote Relational Database (RMTJRNRDB)**

Specifies the name of the relational database directory entry that contains the remote location name of the target system. Use the WRKRDBDIRE command to locate an existing entry or configure a new relational database directory entry for the target system.

**relational-database-directory-entry**

Specify the name of the relational database directory entry. The name can contain up to 18 characters.

**Remote Journal Status (RMTJRNSTS)**

Specifies whether the remote journal is ready to receive journal entries from the queue manager's local journal.

The possible values are:

**\*ACTIVE**

The remote journal is ready to receive journal entries from the local queue manager journal. Replication of journal entries starts with the oldest local journal receiver required to perform a full media recovery and queue manager restart. If these recovery points do not exist, replication starts with the currently attached local journal receiver.

**\*INACTIVE**

The remote journal is not ready to receive journal entries from the local queue manager journal.

**Remote Journal Delivery (RMTJRNDLV)**

Specifies whether the journal entries are replicated synchronously or asynchronously when the remote journal is activated. Note that this parameter is ignored when RMTJRNSTS(\*INACTIVE) is specified.

The possible values are:

**\*SYNC**

The remote journal is replicated synchronously with the local queue manager journal.

**\*ASYNC**

The remote journal is replicated asynchronously with the local queue manager journal.

**Remote Journal Sync. Timeout (RMTJRNTIMO)**

Specifies the maximum amount of time in seconds to wait for a response from the remote system when using synchronous replication with remote journaling. If a response is not received from the remote system within the timeout period, the remote journal environment will automatically be deactivated. Note that this parameter is ignored when RMTJRNDLV(\*ASYNC) or RMTJRNSTS(\*INACTIVE) are specified.

The possible values are:

**\*DFT**

The system uses the default value of 60 seconds to wait for a response from the remote system.

**1-3600**

Specify the maximum number of seconds to wait for a response from the remote system. Note that this option is only available on IBM i V6R1M0 and later operating systems.

## IBM i CCTMQM (Connect MQ)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Connect Message Queue Manager (CCTMQM) command does not perform any function and is only provided for compatibility with previous releases of IBM MQ and MQSeries®.

### Parameters

None

## IBM i CHGMQM (Change Message Queue Manager)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Change Message Queue Manager (CHGMQM) command changes the specified attributes of the local queue manager.

### Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Key, Positional 1
<u>FORCE</u>	Force	<b>*NO</b> , *YES	Optional, Positional 2
<u>TEXT</u>	Text 'description'	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 3
<u>TRGITV</u>	Trigger interval	0-999999999, <b>*SAME</b>	Optional, Positional 4
<u>UDLMSGQ</u>	Undelivered message queue	Character value, *NONE, <b>*SAME</b>	Optional, Positional 5
<u>DFTTMQ</u>	Default transmission queue	Character value, *NONE, <b>*SAME</b>	Optional, Positional 6
<u>MAXHDL</u>	Maximum handle limit	0-999999999, <b>*SAME</b>	Optional, Positional 7
<u>MAXUMSG</u>	Maximum uncommitted messages	1-999999999, <b>*SAME</b>	Optional, Positional 8
<u>AUTEVT</u>	Authorization events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 9
<u>INHEVT</u>	Inhibit events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 10
<u>LCLERREVT</u>	Local error events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 11
<u>RMTERREVT</u>	Remote error events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 12
<u>PFREVT</u>	Performance events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 13

Table 219. Queue manager attributes (continued)

Keyword	Description	Choices	Notes
<a href="#">STRSTPEVT</a>	Start and stop events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 14
<a href="#">CHAD</a>	Automatic Channel Definition	<b>*SAME</b> , *YES, *NO	Optional, Positional 15
<a href="#">CHADEV</a>	Automatic Channel Definition events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 16
<a href="#">CHADEXIT</a>	Automatic Channel Definition exit program	Single values: <b>*SAME</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 17
	Qualifier 1: Automatic Channel Definition exit program	Name	
	Qualifier 2: Library	Name	
<a href="#">MAXMSGL</a>	Max message length	32768-104857600, <b>*SAME</b>	Optional, Positional 18
<a href="#">CCSID</a>	Coded Character Set	<i>Integer</i> , <b>*SAME</b>	Optional, Positional 19
<a href="#">CLWLDATA</a>	Cluster workload exit data	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 20
<a href="#">CLWLEXIT</a>	Cluster workload exit	Single values: <b>*SAME</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 21
	Qualifier 1: Cluster workload exit	Name	
	Qualifier 2: Library	Name	
<a href="#">CLWLLEN</a>	Cluster workload exit length	0-999999999, <b>*SAME</b>	Optional, Positional 22
<a href="#">REPOS</a>	Repository name	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 23
<a href="#">REPOSNL</a>	Repository name list	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 24
<a href="#">SSLCRLNL</a>	TLS CRL Namelist	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 25
<a href="#">SSLKEYR</a>	TLS Key Repository	<i>Character value</i> , *NONE, <b>*SAME</b> , *SYSTEM	Optional, Positional 26
<a href="#">SSLKEYRPWD</a>	TLS Repository Password	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 27
<a href="#">SSLRSTCNT</a>	TLS key reset count	0-999999999, <b>*SAME</b> , *NONE	Optional, Positional 28
<a href="#">IPADDRV</a>	IP protocol	<b>*SAME</b> , *IPv4, *IPv6	Optional, Positional 29
<a href="#">CLWLMRUC</a>	Cluster workload channels	0-999999999, <b>*SAME</b>	Optional, Positional 30
<a href="#">CLWLUSEQ</a>	Cluster workload queue use	<b>*SAME</b> , *LOCAL, *ANY	Optional, Positional 31

Table 219. Queue manager attributes (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>LOGGEREVT</u>	Log recovery events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 32
<u>CHLEVT</u>	Channel events enabled	<b>*SAME</b> , *YES, *NO, *EXCEPTION	Optional, Positional 33
<u>SSLEVT</u>	TLS events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 34
<u>SCHINIT</u>	Channel initiator control	<b>*SAME</b> , *QMGR, *MANUAL	Optional, Positional 35
<u>SCMDSERV</u>	Command server control	<b>*SAME</b> , *QMGR, *MANUAL	Optional, Positional 36
<u>MONQ</u>	Queue Monitoring	<b>*SAME</b> , *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 37
<u>MONCHL</u>	Channel Monitoring	<b>*SAME</b> , *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 38
<u>MONACLS</u>	Cluster Sender Monitoring	<b>*SAME</b> , *QMGR, *NONE, *LOW, *MEDIUM, *HIGH	Optional, Positional 39
<u>STATMQI</u>	Queue Manager Statistics	<b>*SAME</b> , *OFF, *ON	Optional, Positional 40
<u>STATQ</u>	Queue Statistics	<b>*SAME</b> , *NONE, *OFF, *ON	Optional, Positional 41
<u>STATCHL</u>	Channel Statistics	<b>*SAME</b> , *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 42
<u>STATACLS</u>	Cluster Sender Statistics	<b>*SAME</b> , *QMGR, *NONE, *LOW, *MEDIUM, *HIGH	Optional, Positional 43
<u>STATINT</u>	Statistics Interval	1-604800, <b>*SAME</b>	Optional, Positional 44
<u>ACCTMQI</u>	MQI Accounting	<b>*SAME</b> , *OFF, *ON	Optional, Positional 45
<u>ACCTQ</u>	Queue Accounting	<b>*SAME</b> , *NONE, *OFF, *ON	Optional, Positional 46
<u>ACCTINT</u>	Accounting Interval	1-604800, <b>*SAME</b>	Optional, Positional 47
<u>ACCTCONO</u>	Accounting Override	<b>*SAME</b> , *ENABLED, *DISABLED	Optional, Positional 48
<u>ROUTEREC</u>	Trace Route Recording	<b>*SAME</b> , *MSG, *QUEUE, *DISABLED	Optional, Positional 49
<u>ACTIVREC</u>	Activity Recording	<b>*SAME</b> , *MSG, *QUEUE, *DISABLED	Optional, Positional 50
<u>MAXPROPLEN</u>	Maximum Property Data Length	0-104857600, <b>*SAME</b> , *ANY	Optional, Positional 51
<u>MARKINT</u>	Message mark-browse interval	0-999999999, <b>*SAME</b> , *ANY	Optional, Positional 52
<u>PSRTYCNT</u>	PubSub max msg retry count	0-999999999, <b>*SAME</b>	Optional, Positional 53
<u>PSNPMSG</u>	PubSub NPM msg	<b>*SAME</b> , *DISCARD, *KEEP	Optional, Positional 54



Table 219. Queue manager attributes (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<a href="#"><u>PSNPMRES</u></a>	PubSub NPM msg response	<b>*SAME</b> , *NORMAL, *SAFE, *DISCARD, *KEEP	Optional, Positional 55
<a href="#"><u>PSSYNCPT</u></a>	PubSub syncpoint	<b>*SAME</b> , *YES, *IFPER	Optional, Positional 56
<a href="#"><u>PSMODE</u></a>	Pubsub Engine Control	<b>*SAME</b> , *ENABLED, *DISABLED, *COMPATIBLE	Optional, Positional 57
<a href="#"><u>TREELIFE</u></a>	Topic Tree Life Time	0-604000, <b>*SAME</b>	Optional, Positional 58
<a href="#"><u>CFGEVT</u></a>	Configuration events enabled	<b>*SAME</b> , *YES, *NO	Optional, Positional 59
<a href="#"><u>CMDEVT</u></a>	Command events enabled	<b>*SAME</b> , *YES, *NO, *NODSP	Optional, Positional 60
<a href="#"><u>ACTVTRC</u></a>	Activity tracing	Character value, *ON, <b>*SAME</b> , *OFF	Optional, Positional 61
<a href="#"><u>ACTVCONO</u></a>	Override activity tracing	Character value, *DISABLED, <b>*SAME</b> , *ENABLED	Optional, Positional 62
<a href="#"><u>CHLAUTH</u></a>	Channel authentication	Character value, *DISABLED, <b>*SAME</b> , *ENABLED	Optional, Positional 63
<a href="#"><u>CUSTOM</u></a>	Custom attribute	Character value, *NONE, <b>*SAME</b> , 128 character string	Optional, Positional 64
<a href="#"><u>DFTCLXQ</u></a>	Default cluster transmission queue type	<b>*SAME</b> , *SCTQ, *CHANNEL	Optional, Positional 65
<a href="#"><u>CERTLABL</u></a>	Certificate label	<b>*SAME</b> , *DFT	Optional, Positional 66
<a href="#"><u>REVDNS</u></a>	Reverse lookup of host name	<b>*SAME</b> , *DISABLED, *ENABLED	Optional, Positional 67
<a href="#"><u>CONNAUTH</u></a>	Connection authentication object	<b>*SAME</b> , *NONE, 48 character string	Optional, Positional 68
<a href="#"><u>IMGSCHEd</u></a>	Media image scheduling	<b>*SAME</b> , *MANUAL, *AUTO	Optional, Positional 69
<a href="#"><u>IMGINTVL</u></a>	Media image write interval	<b>*SAME</b> , *OFF, 1 - 999999999	Optional, Positional 70
<a href="#"><u>IMGLOGLN</u></a>	Recovery log target size	<b>*SAME</b> , *OFF, 1 - 999999999	Optional, Positional 71
<a href="#"><u>IMGRCOVO</u></a>	Whether objects recoverable	<b>*SAME</b> , *NO, *YES	Optional, Positional 72
<a href="#"><u>IMGRCOVQ</u></a>	Queue object attribute	<b>*SAME</b> , *NO, *YES	Optional, Positional 73

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Force (FORCE)**

Specifies whether the command should be forced to complete if both of the following statements are true:

- DFTTMQ is specified.
- An application has a remote queue open, the resolution of which will be affected by this change.

The possible values are:

**\*NO**

The command fails if an open remote queue will be affected.

**\*YES**

The command is forced to complete.

**Text 'description' (TEXT)**

Specifies the text that briefly describes the queue manager definition.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Trigger interval (TRGITV)**

Specifies the trigger time interval, expressed in milliseconds, to be used with queues that have TRGTYPE(\*FIRST) specified.

When TRGTYPE(\*FIRST) is specified the arrival of a message on a previously empty queue causes a trigger message to be generated. Any further messages that arrive on the queue within the specified interval will not cause a further trigger message to be generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**interval-value**

Specify a value in the range 0 through 999999999.

**Undelivered message queue (UDLMSGQ)**

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

There is no undelivered-message queue. The attribute is set to a blank string.

**undelivered-message-queue-name**

Specify the name of a local queue that is to be used as the undelivered-message queue.

**Default transmission queue (DFTTMQ)**

Specifies the name of the local transmission queue that is to be used as the default transmission queue. Messages transmitted to a remote queue manager are put on the default transmission queue if there is no transmission queue defined for their destination.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

There is no default transmission queue. The attribute is set to a blank string.

**default-transmission-queue-name**

Specify the name of a local transmission queue that is to be used as the default transmission queue.

**Maximum handle limit (MAXHDL)**

Specifies the maximum number of handles that any one job can have open at the same time.

The possible values are:

**\*SAME**

The attribute is unchanged.

**maximum-handle-limit**

Specify a value in the range 0 through 999999999.

**Maximum uncommitted messages (MAXUMSG)**

Specifies the maximum number of uncommitted messages. That is:

- The number of messages that can be retrieved, plus
- The number of messages that can be put, plus
- Any trigger and report messages generated within this unit of work, under any one syncpoint.

This limit does not apply to messages that are retrieved or put outside syncpoint.

The possible values are:

**\*SAME**

The attribute is unchanged.

**maximum-uncommitted-messages**

Specify a value in the range 1 through 999999999.

**Authorization events enabled (AUTEVT)**

Specifies whether authorization (Not Authorized) events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Authorization events are not generated.

**\*YES**

Authorization events are generated.

**Inhibit events enabled (INHEVT)**

Specifies whether inhibit events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Inhibit events are not generated.

**\*YES**

Inhibit events are generated.

**Local error events enabled (LCLERREVT)**

Specifies whether local error events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Local error events are not generated.

**\*YES**

Local error events are generated.

**Remote error events enabled (RMTERREVT)**

Specifies whether remote error events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Remote error events are not generated.

**\*YES**

Remote error events are generated.

**Performance events enabled (PFREVT)**

Specifies whether performance events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Performance events are not generated.

**\*YES**

Performance events are generated.

**Start and stop events enabled (STRSTPEVT)**

Specifies whether start and stop events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Start and stop events are not generated.

**\*YES**

Start and stop events are generated.

**Automatic Channel Definition (CHAD)**

Specifies whether receiver and server-connection channels are automatically defined.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Receiver and server-connection channels are not automatically defined.

**\*YES**

Receiver and server-connection channels are automatically defined.

**Automatic Channel Definition events enabled (CHADEV)**

Specifies whether automatic channel definition events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Automatic channel definition events are not generated.

**\*YES**

Automatic channel definition events are generated.

**Automatic Channel Definition exit program (CHADEXIT)**

Specifies the entry point of the program to be called as the automatic channel-definition exit.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No automatic channel definition exit is invoked.

**channel-definition-exit-name**

Specify the name of the channel definition exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified and the values \*LIBL and \*CURLIB are not permitted.

**Maximum Message Length (MAXMSGL)**

Specifies the maximum message length of messages (in bytes) allowed on queues for this queue manager.

The possible values are:

**\*SAME**

The attribute is unchanged.

**maximum-message-length**

Specify a value in bytes, in the range 32 KB through 100 MB.

**Coded Character Set (CCSID)**

The coded character set identifier for the queue manager.

The CCSID is the identifier used with all character string fields defined by the API. It does not apply to application data carried in the text of messages unless the CCSID in the message descriptor is set to the value MQCCSI\_Q\_MGR when the message is put to a queue.

If you use this keyword to change the CCSID, applications that are running when the change is applied continue to use the original CCSID. You must stop and restart all running applications before you continue. This includes the command server and channel programs. You are recommended to stop and restart the queue manager after making the change to achieve this.

The possible values are:

**\*SAME**

The attribute is unchanged.

**number**

Specify a value in the range 1 through 65535. The value must represent a coded character set identifier (CCSID) that is recognised by the system.

**Cluster Workload Exit Data (CLWLDATA)**

Specifies the cluster workload exit data (maximum length 32 characters).

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The cluster workload exit data is not specified.

**cluster-workload-exit-data**

This is passed to the cluster-workload exit when it is called.

**Cluster Workload Exit (CLWLEXIT)**

Specifies the entry point of the program to be called as the cluster-workload exit.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No cluster-workload exit is invoked.

**cluster-workload-exit**

You must specify a fully-qualified name, when you specify a cluster-workload exit. In this instance, the libraries defined as \*LIBL and \*CURLIB are not permitted.

**Cluster Workload Exit Data Length (CLWLLEN)**

The maximum number of bytes of message data that is passed to the cluster workload exit.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-exit-data-length**

Specify a value in bytes, in the range 0 through 999999999.

## Repository name (REPOS)

The name of a cluster for which this queue manager is to provide a repository manager service.

If the parameter REPOSNL is non-blank this parameter must be blank.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

A cluster is not specified.

### **clustername**

The maximum length is 48 characters conforming to the rules for naming IBM MQ objects.

## Repository name list (REPOSNL)

The name of a namelist of clusters for which this queue manager is to provide a repository manager service.

If the parameter REPOS is non-blank this parameter must be blank.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

A namelist of clusters is not specified.

### **namelist**

The name of the namelist.

## TLS CRL Namelist (SSLCRLNL)

The name of a namelist of authinfo objects which this queue manager uses to check certificate status.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

A namelist of authinfo objects is not specified.

### **namelist**

The name of the namelist.

## TLS Key Repository (SSLKEYR)

The location of a key repository for this queue manager.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*SYSTEM**

The queue manager uses the \*SYSTEM key repository. Setting the SSLKEYR repository to this value causes the queue manager to be registered as an application to Digital Certificate Manager. You can assign any client or server certificate in the \*SYSTEM store to the queue manager through Digital Certificate Manager. If you specify this value you are not required to set the key repository password (SSLKEYRPWD).

### **\*NONE**

A key repository is not specified.

**filename**

The location of the CMS key repository. If you specify this value you must ensure the key repository contains a correctly labeled digital certificate and also set the key repository password (SSLKEYRPWD) to enable channels to access the key repository. See the IBM MQ Security information for more details.

**TLS Repository Password (SSLKEYRPWD)**

The password of a key repository for this queue manager.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

A key repository password is not specified.

**password**

The password of the repository.



**Attention:** **V9.3.0** The keystore password is encrypted using the IBM MQ password protection system. Prior to setting the password, you should set a unique initial key in the queue manager.

**TLS key reset count (SSLRSTCNT)**

Specifies when TLS channel MCAs that initiate communication reset the secret key used for encryption on the channel. The value represents the total number of unencrypted bytes that are sent and received on the channel before the secret key is renegotiated. The number of bytes includes control information sent by the message channel agent.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Secret key renegotiation is disabled.

**key-reset-byte-count**

Specify a value in bytes, in the range 0 through 999999999. A value of 0 indicates that secret key renegotiation is disabled.

**IP protocol (IPADDRV)**

The IP protocol to use for channel connections.

This attribute is only relevant for systems enabled for both IPv4 and IPv6. The attribute affects channels with TRPTYPE defined as TCP when the CONNAME is defined as a host name that resolves to both an IPv4, and an IPv6 address, and one of the following is true:

- LOCLADDR is not specified.
- LOCLADDR also resolves to both an IPv4 and an IPv6 address.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*IPv4**

The IPv4 stack is used.

**\*IPv6**

The IPv6 stack is used.



## Cluster workload channels (CLWLMRUC)

Specifies the maximum number of most-recently-used cluster channels, to be considered for use by the cluster workload choice algorithm.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **maximum-cluster-workload-channels**

Specify a value in the range 0 through 999999999.

## Cluster workload queue use (CLWLUSEQ)

Specifies the behavior of an MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel then this attribute does not apply. This value is used for queues where the CLWLUSEQ value is \*QMGR.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*LOCAL**

The local queue will be the sole target of the MQPUT.

### **\*ANY**

The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

## Log recovery events enabled (LOGGEREVT)

Specifies whether log recovery events are generated.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NO**

Log recovery events are not generated.

### **\*YES**

Log recovery events are generated.

## Channel events enabled (CHLEVT)

Specifies whether channel events are generated.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NO**

Channel events are not generated.

### **\*EXCEPTION**

Exception channel events are generated.

Only the following channel events are generated:

- MQRC\_CHANNEL\_ACTIVATED
- MQRC\_CHANNEL\_CONV\_ERROR
- MQRC\_CHANNEL\_NOT\_ACTIVATED
- MQRC\_CHANNEL\_STOPPED

The channel events are issued with the following reason qualifiers:

- MQRQ\_CHANNEL\_STOPPED\_ERROR
- MQRQ\_CHANNEL\_STOPPED\_RETRY
- MQRQ\_CHANNEL\_STOPPED\_DISABLED
- MQRQ\_CHANNEL\_STOPPED\_BY\_USER

**\*YES**

All channel events are generated.

In addition to those generated by \*EXCEPTION the following channel events are also generated:

- MQRQ\_CHANNEL\_STARTED
  - MQRQ\_CHANNEL\_STOPPED
- with the following reason qualifier:
- MQRQ\_CHANNEL\_STOPPED\_OK

**TLS events enabled (SSLEVT)**

Specifies whether TLS events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

TLS events are not generated.

**\*YES**

TLS events are generated.

The following event is generated:

- MQRQ\_CHANNEL\_SSL\_ERROR

**Channel initiator control (SCHINIT)**

Specifies the channel initiator control.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

Start and stop the channel initiator with the queue manager.

**\*MANUAL**

Do not automatically start the channel initiator with the queue manager.

**Command server control (SCMDSERV)**

Specifies the command server control.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

Start and stop the command server with the queue manager.

**\*MANUAL**

Do not automatically start the command server with the queue manager.

## Queue Monitoring (MONQ)

Controls the collection of online monitoring data for queues.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

Online monitoring data for queues is disabled regardless of the setting of the MONQ queue attribute.

### **\*OFF**

Monitoring data collection is turned off for queues specifying \*QMGR in the MONQ queue attribute.

### **\*LOW**

Monitoring data collection is turned on with a low ratio of data collection for queues specifying \*QMGR in the MONQ queue attribute.

### **\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection for queues specifying \*QMGR in the MONQ queue attribute.

### **\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection for queues specifying \*QMGR in the MONQ queue attribute.

## Channel Monitoring (MONCHL)

Controls the collection of online monitoring data for channels.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

Online monitoring data for channels is disabled regardless of the setting of the MONCHL channel attribute.

### **\*OFF**

Monitoring data collection is turned off for channels specifying 'QMGR' in the MONCHL queue attribute.

### **\*LOW**

Monitoring data collection is turned on with a low ratio of data collection for channels specifying \*QMGR in the MONCHL channel attribute.

### **\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection for channels specifying \*QMGR in the MONCHL channel attribute.

### **\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection for channels specifying \*QMGR in the MONCHL channel attribute.

## Cluster Sender Monitoring (MONACLS)

Controls the collection of online monitoring data for auto-defined cluster sender channels. The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

Online monitoring data for auto-defined cluster sender channels is disabled.

**\*QMGR**

The collection of Online Monitoring Data is inherited from the setting of the MONCHL attribute in the QMGR object.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection for auto-defined cluster sender channels.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection for auto-defined cluster sender channels.

**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection for auto-defined cluster sender channels.

**Queue Manager Statistics (STATMQI)**

Controls the collection of statistics monitoring information for the queue manager. The possible values are:

**\*SAME**

The attribute is unchanged.

**\*OFF**

Data collection for MQI statistics is disabled.

**\*ON**

Data collection for MQI statistics is enabled.

**Queue Statistics (STATQ)**

Controls the collection of statistics data for queues. The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Data collection for queue statistics is disabled for all queues regardless of the setting of the STATQ queue attribute.

**\*OFF**

Statistics data collection is turned off for queues specifying \*QMGR in the STATQ queue attribute.

**\*ON**

Statistics data collection is turned on for queues specifying \*QMGR in the STATQ queue attribute.

**Channel Statistics (STATCHL)**

Controls the collection of statistics data for channels. The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Data collection for channel statistics is disabled for all channels regardless of the setting of the STATCHL channel attribute.

**\*OFF**

Statistics data collection is turned off for channels specifying \*QMGR in the STATCHL channel attribute.

**\*LOW**

Statistics data collection is turned on with a low ratio of data collection for channels specifying \*QMGR in the STATCHL channel attribute.

**\*MEDIUM**

Statistics data collection is turned on with a moderate ratio of data collection for channels specifying \*QMGR in the STATCHL channel attribute.

**\*HIGH**

Statistics data collection is turned on with a high ratio of data collection for channels specifying \*QMGR in the STATCHL channel attribute.

**Cluster Sender Statistics (STATACLS)**

Controls the collection of statistics data for auto-defined cluster sender channels. The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Statistics data collection for auto-defined cluster sender channels is disabled.

**\*LOW**

Statistics data collection for auto-defined cluster sender channels is enabled with a low ratio of data collection.

**\*MEDIUM**

Statistics data collection for auto-defined cluster sender channels is enabled with a moderate ratio of data collection.

**\*HIGH**

Statistics data collection for auto-defined cluster sender channels is enabled with a high ratio of data collection.

**Statistics Interval (STATINT)**

How often (in seconds) statistics monitoring data is written to the monitoring Queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**statistics-interval**

Specify a value in the range 1 through 604800.

**MQI Accounting (ACCTMQI)**

Controls the collection of accounting information for MQI data. The possible values are:

**\*SAME**

The attribute is unchanged.

**\*OFF**

API accounting data collection is disabled.

**\*ON**

API accounting data collection is enabled.

**Queue Accounting (ACCTQ)**

Controls the collection of accounting information for queues. The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Accounting data collection for queues is disabled and may not be overridden using the queue attribute ACCTQ.

**\*OFF**

Accounting data collection is turned off for queues specifying \*QMGR in the ACCTQ queue attribute.

**\*ON**

Accounting data collection is turned on for queues specifying \*QMGR in the ACCTQ queue attribute.

**Accounting Interval (ACCTINT)**

After how long in seconds, intermediate accounting records are written.

The possible values are:

**\*SAME**

The attribute is unchanged.

**accounting-interval**

Specify a value in the range 1 through 604800.

**Accounting Override (ACCTCONO)**

Whether applications can override the setting of the ACCTMQI and the ACCTQ values in the QMGR attribute. The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ENABLED**

Application may override the setting of the ACCTMQI and ACCTQ QMGR attributes using the Options field in the MQCNO structure on the MQCONN api call.

**\*DISABLED**

Application may not override the setting of the ACCTMQI and ACCTQ QMGR attributes using the Options field in the MQCNO structure on the MQCONN api call.

**Trace Route Recording (ROUTEREC)**

Controls the recording of trace route information.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*MSG**

Reply put to destination specified by the message.

**\*QUEUE**

Reply put to fixed name queue.

**\*DISABLED**

No appending to trace route messages allowed.

**Activity Recording (ACTIVREC)**

Controls the generation of activity reports.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*MSG**

Report put to destination specified by the message.

**\*QUEUE**

Report put to fixed name queue.

**\*DISABLED**

No activity reports are generated.

**Maximum Property Data Length (MAXPROPLEN)**

Specifies a maximum length for property data.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ANY**

There is no limit on the length of property data.

**max-property-data-length**

Specify a value in bytes, in the range 0 through 104857600 (ie: 10 MB).

**Message mark-browse interval (MARKINT)**

An approximate time interval in milliseconds, for which messages that have been marked-browsed by a call to MQGET with the get message option MQGMO\_MARK\_BROWSE\_CO\_OP are expected to remain marked-browsed.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ANY**

Messages will remain marked-browsed indefinitely.

**A time interval**

A time interval expressed in milliseconds, up to a maximum of 999999999. The default value is 5000.



**Attention:** You should not reduce the value below the default of 5000.

**PubSub max msg retry count (PSRTCNT)**

The number of retries when processing (under syncpoint) a failed command message.

The possible values are:

**\*SAME**

The attribute is unchanged.

**Retry count**

Specify a value in the range 0 through 999999999.

**PubSub NPM msg (PSNPMMSG)**

Whether to discard (or keep) a undelivered input message

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*DISCARD**

Non-persistent input messages may be discarded if they cannot be processed.

**\*KEEP**

Non-persistent input messages will not be discarded if they cannot be processed. In this situation the queued pubsub daemon will continue to retry processing the message. Subsequent input messages are not processed until the message is successfully processed.

## PubSub NPM msg response (PSNPMRES)

Controls the behavior of undelivered response messages

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NORMAL**

Non-persistent responses that cannot be placed on the reply queue are put on the dead letter queue. If they cannot be placed on the dead letter queue then they are discarded.

### **\*SAFE**

Non-persistent responses which cannot be placed on the reply queue are put on the dead letter queue. If the response cannot be placed on the dead letter queue then the message will be rolled back and then retried. Subsequent messages are not processed until the message is delivered.

### **\*DISCARD**

Non-persistent responses are not placed on the reply queue but are discarded.

### **\*KEEP**

Non-persistent responses that cannot be delivered will be rolled back and the delivery retried. Subsequent messages are not processed until the message is delivered.

## PubSub syncpoint (PSSYNCP)

Whether only persistent (or all) messages should be processed under syncpoint

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*IFPER**

This makes the queued pubsub daemon receive non-persistent messages outside syncpoint. If the daemon receives a publication outside syncpoint, the daemon forwards the publication to subscribers known to it outside syncpoint.

### **\*YES**

This makes the queued pubsub daemon receive all messages under syncpoint.

## Pubsub Engine Control (PSMODE)

Pubsub Engine Control.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*ENABLED**

The publish/subscribe engine and the queued publish/subscribe interface are running. It is therefore possible to publish/subscribe by using the application programming interface, the queues that are being monitored by the queued publish/subscribe interface, or both.

### **\*DISABLED**

The publish/subscribe engine and the queued publish/subscribe interface are not running. It is not possible to publish/subscribe by using the application programming interface. Any publish/subscribe messages put to the queues that are monitored by the queued publish/subscribe interface will not be acted upon.

### **\*COMPATIBLE**

The publish/subscribe engine is running. It is possible to publish subscribe by using the application programming interface. The queued publish/subscribe interface is not running. Any publish/subscribe messages put to the queues that are monitored by the queued publish/subscribe interface will not be



acted upon. Use this for compatibility with WebSphere Message Broker V6, or earlier versions, using this queue manager

### **Topic Tree Life Time (TREELIFE)**

Specifies a lifetime in seconds of non-administrative topics. Non-administrative topics are those created when an application publishes to, or subscribes on, a topic string that does not exist as an administrative node. When this non-administrative node no longer has any active subscriptions, this parameter determines how long the queue manager will wait before removing that node. Only non-administrative topics that are in use by a durable subscription remain after the queue manager is recycled.

The possible values are:

**\*SAME**

The attribute is unchanged.

**tree-life-time**

Specify a value in seconds, in the range 0 through 604000. A value of 0 means that non-administrative topics are not removed by the queue manager.

### **Configuration events enabled (CFGEVT)**

Specifies whether configuration events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Configuration events are not generated.

**\*YES**

Configuration events are generated. After setting this value, issue MQSC REFRESH QMGR TYPE(CONFIGEV) commands for all objects to bring the queue manager configuration up to date.

### **Command events enabled (CMDEVT)**

Specifies whether command events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Command events are not generated.

**\*YES**

Command events are generated for all successful commands.

**\*NODSP**

Command events are generated for all successful commands, other than DISPLAY commands.

### **ACTVTRC**

This attribute specifies whether MQI application activity tracing information is to be collected. See [Setting ACTVTRC to control collection of activity trace information](#).

**\*SAME**

The attribute is unchanged.

**\*OFF**

IBM MQ MQI application activity tracing information collection is not enabled.

**\*ON**

IBM MQ MQI application activity tracing information collection is enabled.

If the queue manager attribute ACTVCON0 is set to ENABLED, the value of this parameter can be overridden using the options field of the MQCNO structure.

## ACTVCON0

This attribute specifies whether applications can override the settings of the ACTVTRC queue manager parameter:

### \*SAME

The attribute is unchanged. This is the default value

### \*DISABLED

Applications cannot override the settings of the ACTVTRC queue manager parameter.

### \*ENABLED

Applications can override the settings of the ACTVTRC queue manager parameter by using the options field of the MQCNO structure of the MQCONN API call.

Changes to this parameter are effective for connections to the queue manager that occur after the change.

## CHLAUTH

This attribute specifies whether the rules defined by channel authentication records are used. CHLAUTH rules can still be set and displayed regardless of the value of this attribute.

Changes to this parameter take effect the next time that an inbound channel attempts to start.

Channels that are currently started are unaffected by changes to this parameter.

### \*SAME

The attribute is unchanged. This is the default value

### \*DISABLED

Channel authentication records are not checked.

### \*ENABLED

Channel authentication records are checked.

## Custom attribute (CUSTOM)

This attribute is reserved for the configuration of new features before separate attributes have been introduced. This description will be updated when features using this attribute are introduced. At the moment there are no meaningful values for *CUSTOM*, so leave it empty.

The possible values are:

### \*SAME

The attribute is unchanged.

### \*NONE

The text is set to a blank string.

### 128 character custom string

Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs must have the form NAME (VALUE) and be specified in uppercase. Single quotes must be escaped with another single quote.

## Default cluster transmission queue type (DFTCLXQ)

The **DEFCLXQ** attribute controls which transmission queue is selected by default by cluster-sender channels to get messages from, to send the messages to cluster-receiver channels.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*SCTQ**

All cluster-sender channels send messages from `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. The `correlID` of messages placed on the transmission queue identifies which cluster-sender channel the message is destined for.

SCTQ is set when a queue manager is defined.

**\*CHANNEL**

Each cluster-sender channel sends messages from a different transmission queue. Each transmission queue is created as a permanent dynamic queue from the model queue `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`.

**CERTLABL**

This attribute specifies the certificate label for this queue manager to use. The label identifies which personal certificate in the key repository has been selected.

The default and migrated queue manager values on IBM i, are:

- If you specified `SSLKEYR(*SYSTEM)`, the value is blank.

Note that it is forbidden to use a nonblank queue manager `CERTLABL` with `SSLKEYR(*SYSTEM)`. Attempting to do so results in an `MQRCCF_Q_MGR_ATTR_CONFLICT` error.

- Otherwise, *ibmwebspheremqxxxx* where *xxxx* is the queue manager name folded to lowercase.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*DFT**

Leaving `CERTLABL` as a blank value on the queue manager is interpreted by the system to mean the default values specified.

**REVDNS**

This attribute controls whether reverse lookup of the host name from a Domain Name Server (DNS) is done for the IP address from which a channel has connected. This attribute has an effect only on channels using a transport type (TRPTYPE) of TCP.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ENABLED**

DNS host names are reverse looked-up for the IP addresses of inbound channels when this information is required. This setting is required for matching against `CHLAUTH` rules that contain host names, and to include the host name in error messages. The IP address is still included in messages that provide a connection identifier.

This is the initial default value for the queue manager.

**\*DISABLED**

DNS host names are not reverse looked-up for the IP addresses of inbound channels. With this setting any `CHLAUTH` rules using host names are not matched.

## CONNAUTH

This attribute specifies the name of an authentication information object that is used to provide the location of user ID and password authentication. If **CONNAUTH** is \*NONE, no user ID and password checking is done by the queue manager.

Changes to this configuration, or the object to which it refers, take effect when a **REFRESH SECURITY TYPE(CONNAUTH)** command is issued.

If you set **CONNAUTH** to \*NONE, and attempt to connect to a channel that has the REQDADM option set in the **CHCKCLNT** field, the connection fails.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

No user ID and password checking is done by the queue manager

### **48 character conn auth string**

The specific name of an authentication information object that is used to provide the location of user ID and password authentication.

## IMGSCHED

This attribute specifies whether the queue manager automatically writes media images.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*AUTO**

The queue manager attempts to automatically write a media image for an object, before **IMGINTVL** minutes have elapsed, or **IMGLOGLN** megabytes of recovery log have been written, since the previous media image for the object was taken.

The previous media image might have been taken manually or automatically, depending on the settings of **IMGINTVL** or **IMGLOGLN**.

### **\*MANUAL**

Automatic media images are not written.

## IMGINTVL

This attribute specifies the target frequency with which the queue manager automatically writes media images, in minutes since the previous media image for the object.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **1 - 999 999 999**

The time in minutes at which the queue manager automatically writes media images.

### **\*OFF**

Automatic media images are not written on a time interval basis.

## IMGLOGLN

This attribute specifies the target size of recovery log, written before the queue manager automatically writes media images, in number of megabytes since the previous media image for the object. This limits the amount of log to be read when recovering an object.

The possible values are:

**\*SAME**

The attribute is unchanged.

**1 - 999 999 999**

The target size of the recovery log in megabytes.

**\*OFF**

Automatic media images are not written based on the size of log written.

**IMGRCOVO**

This attribute specifies whether authentication information, channel, client connection, listener, namelist, process, alias queue, remote queue, and service objects are recoverable from a media image, if linear logging is being used.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

The “[RCDMQMIMG \(Record MQ Object Image\)](#)” on page 1857 and “[RCRMQMOBJ \(Re-create MQ Object\)](#)” on page 1859 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

**\*YES**

These objects are recoverable.

**IMGRCOVQ**

This attribute specifies the **IMGRCOVQ** attribute for local and permanent dynamic queue objects, when used with this parameter.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

The **IMGRCOVQ** attribute for local and permanent dynamic queue objects is set to \*NO.

**\*YES**

The **IMGRCOVQ** attribute for local and permanent dynamic queue objects is set to \*YES.

 **CHGMQMAUTI (Change MQ AuthInfo object)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Change MQ AuthInfo object (CHGMQMAUTI) command changes the specified attributes of an existing MQ authentication information object.

**Parameters**

<i>Table 220. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>AINAME</u>	AuthInfo name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , *DFT	Optional, Key, Positional 2

Table 220. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>AUTHTYPE</u>	AuthInfo type	*CRLLDAP, *OCSP, *IDPWOS, *IDPWLDAP	Optional, Positional 3
<u>CONNNAME</u>	Connection name	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 4
<u>TEXT</u>	Text 'description'	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 5
<u>USERNAME</u>	User name	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 6
<u>PASSWORD</u>	User password	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 7
<u>OCSPURL</u>	OCSP Responder URL	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 8
<u>CHCKCLNT</u>	Authentication checks required	*ASQMGR, *REQUIRED, *REQADM	Optional, Positional 9
<u>CHCKLOCL</u>	Authentication checks required	*NONE, *OPTIONAL, *REQUIRED, *REQADM	Optional, Positional 10
<u>FAILDELAY</u>	Failure delay	<i>Integer value</i>	Optional, Positional 11
<u>BASEDNU</u>	Base user DN	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 12
<u>ADOPTCTX</u>	Context adoption	<i>Integer value</i>	Optional, Positional 13
<u>CLASSUSER</u>	LDAP object class	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 14
<u>USERFIELD</u>	LDAP user record	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 15
<u>SHORTUSER</u>	User record	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 16
<u>SECCOMM</u>	LDAP communications	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 17
<u>AUTHORMD</u>	Authorization method	<i>Character value</i> , <b>*OS</b> , *SEARCHGRP, *SEARCHUSR, *SRCHGRPSN	Optional, Positional 18
<u>BASEDNG</u>	Base DN for groups	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 19
<u>CLASSGRP</u>	Object class for group	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 20
<u>FINDGRP</u>	Attribute to find group membership	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 21
<u>GRPFIELD</u>	Simple name for group	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 22
<u>NESTGRP</u>	Group nesting	<b>*NO</b> *YES	Optional, Positional 23
<u>AUTHENMD</u>	Authentication method	<b>*OS</b> Cannot be changed	Optional, Positional 24

### **AuthInfo name (Ainame)**

The name of the authentication information object to change.

The possible values are:

#### **authentication-information-name**

Specify the name of the authentication information object. The maximum string length is 48 characters.

## Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.

### **queue-manager-name**

The name of an existing message queue manager. The maximum string length is 48 characters.

## Adopt context (ADOPTCTX)

Whether to use the presented credentials as the context for this application. This means that they are used for authorization checks, shown on administrative displays, and appear in messages.

### **YES**

The user ID presented in the MQCSP structure, which has been successfully validated by password, is adopted as the context to use for this application. Therefore, this user ID will be the credentials checked for authorization to use IBM MQ resources.

If the user ID presented is an LDAP user ID, and authorization checks are done using operating system user IDs, the [SHORTUSR](#) associated with the user entry in LDAP will be adopted as the credentials for authorization checks to be done against.

### **NO**

Authentication will be performed on the user ID and password presented in the MQCSP structure, but then the credentials will not be adopted for further use. Authorization will be performed using the user ID the application is running under.

This attribute is only valid for an AUTHTYPE of \*IDPWOS and \*IDPWLDAP.

## Authentication method (AUTHENMD)

The authentication method used for this application.

### **\*OS**

Use operating system groups to determine permissions associated with a user.

You can use only **\*OS** to set the authentication method.

This attribute is valid only for an **AUTHTYPE** of \*IDPWOS.

## Authorization method (AUTHORMD)

The authorization method used for this application.

### **\*OS**

Use operating system groups to determine permissions associated with a user.

This is how IBM MQ has previously worked, and is the default value.

### **\*SEARCHGRP**

A group entry in the LDAP repository contains an attribute listing the Distinguished Name of all the users belonging to that group. Membership is indicated by the attribute defined in [FINDGRP](#). This value is typically *member* or *uniqueMember*.

### **\*SEARCHUSR**

A user entry in the LDAP repository contains an attribute listing the Distinguished Name of all the groups to which the specified user belongs. The attribute to query is defined by the [FINDGRP](#) value, typically *memberOf*.

### **\*SRCHGRPSN**

A group entry in the LDAP repository contains an attribute listing the short user name of all the users belonging to that group. The attribute in the user record that contains the short user name is specified by SHORTUSR.

Membership is indicated by the attribute defined in FINDGRP. This value is typically *memberUid*.

**Note:** This authorization method should only be used if all user short names are distinct.

Many LDAP servers use an attribute of the group object to determine group membership and you should, therefore, set this value to *SEARCHGRP*.

Microsoft Active Directory typically stores group memberships as a user attribute. The IBM Tivoli Directory Server supports both methods.

In general, retrieving memberships through a user attribute will be faster than searching for groups that list the user as a member.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

### **AuthInfo type (AUTHTYPE)**

The type of the authentication information object. There is no default value

The possible values are:

#### **\*CRLLDAP**

The type of the authentication information object is CRLLDAP.

#### **\*OCSP**

The type of the authentication information objects is OCSPURL.

#### **\*IDPWOS**

Connection authentication user ID and password checking is done using the operating system.

#### **\*IDPWLDAP**

Connection authentication user ID and password checking is done using an LDAP server.

### **Base DN for groups (BASEDNG)**

In order to be able to find group names, this parameter must be set with the base DN to search for groups in the LDAP server.

This attribute is valid only for **AUTHTYPE** of *\*IDPWLDAP*.

### **Base user DN (BASEDNU)**

In order to be able to find the short user name attribute (see SHORTUSR) this parameter must be set with the base DN to search for users within the LDAP server. This attribute is valid only for **AUTHTYPE** of *\*IDPWLDAP*.

### **Check client (CHKCLNT)**

Whether connection authentication checks are required by all locally bound connections, or only checked when a user ID and password are provided in the MQCSP structure.

These attributes are valid only for an **AUTHTYPE** of *\*IDPWOS* or *\*IDPWLDAP*. The possible values are:

#### **\*ASQMGR**

In order for the connection to be allowed in, it must meet the connection authentication requirements defined on the queue manager. If the CONNAUTH field provides an authentication information object, and the value of CHKCLNT is *\*REQUIRED*, the connection will not be successful unless a valid user ID and password are supplied. If the CONNAUTH field does not provide an authentication information object, or the value of CHKCLNT is not *\*REQUIRED*, then the user ID and password are not required.



**\*REQUIRED**

Requires that all applications provide a valid user ID and password.

**\*REQDADM**

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the \*OPTIONAL setting.

**Check local (CHCKLOCL)**

Whether connection authentication checks are required by all locally bound connections, or only checked when a user ID and password are provided in the MQCSP structure.

These attributes are valid only for an **AUTHTYPE** of \*IDPWOS or \*IDPWLDAP. The possible values are:

**\*NONE**

Switches off checking.

**\*OPTIONAL**

Ensures that if a user ID and password are provided by an application, they are a valid pair, but that it is not mandatory to provide them. This option might be useful during migration, for example.

**\*REQUIRED**

Requires that all applications provide a valid user ID and password.

**\*REQDADM**

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the \*OPTIONAL setting.

**Class group (CLASSGRP)**

The LDAP object class used for group records in the LDAP repository.

If the value is blank, **groupOfNames** is used.

Other commonly used values include *groupOfUniqueNames* or *group*.

This attribute is valid only for **AUTHTYPE** of \*IDPWLDAP.

**Class user (CLASSUSR)**

The LDAP object class used for user records in the LDAP repository.

If blank, the value defaults to *inetOrgPerson*, which is generally the value needed.

For Microsoft Active Directory, the value you require required is often *user*.

This attribute is valid only for an **AUTHTYPE** of \*IDPWLDAP.

**Connection name (CONNAME)**

The DNS name or IP address of the host on which the LDAP server is running, together with an optional port number. The default port number is 389. No default is provided for the DNS name or IP address.

This field is only valid for \*CRLLDAP or \*IDPWLDAP authentication information objects, when it is required.

When used with IDPWLDAP authentication information objects, this can be a comma separated list of connection names.

The possible values are:

**\*SAME**

The connection name remains unchanged from the original authentication information object.

**connection-name**

Specify the fully qualified DNS name or IP address of the host together with an optional port number. The maximum string length is 264 characters.

## Failure delay (FAILDELAY)

When a user ID and password are provided for connection authentication, and the authentication fails due to the user ID or password being incorrect, this is the delay, in seconds, before the failure is returned to the application.

This can aid in avoiding busy loops from an application that simply retries, continuously, after receiving a failure.

The value must be in the range 0 - 60 seconds. The default value is 1.

This attribute is only valid for AUTHTYPE of \*IDPWOS and \*IDPWLDAP.

## Group membership attribute (FINDGRP)

Name of the attribute used within an LDAP entry to determine group membership.

When AUTHORMD = \*SEARCHGRP, this attribute is typically set to *member* or *uniqueMember*.

When AUTHORMD = \*SEARCHUSR, this attribute is typically set to *memberOf*.

When AUTHORMD = \*SRCHGRPSN, this attribute is typically set to *memberUid*.

When left blank, if:

- AUTHORMD = \*SEARCHGRP, this attribute defaults to *memberOf*
- AUTHORMD = \*SEARCHUSR, this attribute defaults to *member*
- AUTHORMD = \*SRCHGRPSN, this attribute defaults to *memberUid*

This attribute is valid only for an **AUTHTYPE** of \*IDPWLDAP.

## Simple name for group (GRPFIELD)

If the value is blank, commands like setmqaut must use a qualified name for the group. The value can either be a full DN, or a single attribute.

This attribute is valid only for an **AUTHTYPE** of \*IDPWLDAP.

## Group nesting (NESTGRP)

The possible values are:

### **\*NO**

Only the initially discovered groups are considered for authorization.

### **\*YES**

The group list is searched recursively to enumerate all the groups to which a user belongs.

The group's Distinguished Name is used when searching the group list recursively, regardless of the authorization method selected in AUTHORMD.

This attribute is valid only for an **AUTHTYPE** of \*IDPWLDAP.

## OCSP Responder URL (OCSPURL)

The URL of the OCSP Responder used to check for certificate revocation. This must be an HTTP URL containing the host name and port number of the OCSP Responder. If the OCSP Responder is using port 80, which is the default for HTTP, then the port number may be omitted.

This field is only valid for OCSP authentication information objects.

The possible values are:

### **\*SAME**

The OCSP Responder URL is unchanged.

### **OCSP-Responder-URL**

The OCSP Responder URL. The maximum string length is 256 characters.

### **Secure comms (SECCOMM)**

Whether connectivity to the LDAP server should be done securely using TLS

#### **YES**

Connectivity to the LDAP server is made securely using TLS.

The certificate used is the default certificate for the queue manager, named in CERTLABL on the queue manager object, or if that is blank, the one described in [Digital certificate labels, understanding the requirements](#).

The certificate is located in the key repository specified in SSLKEYR on the queue manager object. A cipherspec will be negotiated that is supported by both IBM MQ and the LDAP server.

If the queue manager is configured to use SSLFIPS(YES) or SUITEB cipher specs, then this is taken account of in the connection to the LDAP server as well.

#### **ANON**

Connectivity to the LDAP server is made securely using TLS just as for SECCOMM(YES) with one difference.

No certificate is sent to the LDAP server; the connection will be made anonymously. To use this setting, ensure that the key repository specified in SSLKEYR, on the queue manager object, does not contain a certificate marked as the default.

#### **NO**

Connectivity to the LDAP server does not use TLS.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*

### **Short user (SHORTUSR)**

A field in the user record to be used as a short user name in IBM MQ.

This field must contain values of 12 characters or less. This short user name is used for the following purposes:

- If LDAP authentication is enabled, but LDAP authorization is not enabled, this is used as an operating system user ID for authorization checks. In this case, the attribute must represent an operating system user ID.
- If LDAP authentication and authorization are both enabled, this is used as the user ID carried with the message in order for the LDAP user name to be rediscovered when the user ID inside the message needs to be used.

For example, on another queue manager, or when writing report messages. In this case, the attribute does not need to represent an operating system user ID, but must be a unique string. An employee serial number is an example of a good attribute for this purpose.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP* and is mandatory.

### **Text 'description' (TEXT)**

A short text description of the authentication information object.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

#### **\*SAME**

The text string is unchanged.

**\*NONE**

The text is set to a blank string.

**description**

The string length can be up to 64 characters enclosed in apostrophes.

**User name (USERNAME)**

The distinguished name of the user that is binding to the directory. The default user name is blank.

This field is only valid for *\*CRLLDAP* or *\*IDPWLDAP* authentication information objects.

The possible values are:

**\*SAME**

The user name is unchanged.

**\*NONE**

The user name is blank.

**LDAP-user-name**

Specify the distinguished name of the LDAP user. The maximum string length is 1024 characters.

**User field (USRFIELD)**

If the user ID provided by an application for authentication does not contain a qualifier for the field in the LDAP user record, that is, it does not contain an '=' sign, this attribute identifies the field in the LDAP user record that is used to interpret the provided user ID.

This field can be blank. If this is the case, any unqualified user IDs use the [SHORTUSR](#) parameter to interpret the provided user ID.

The contents of this field will be concatenated with an '=' sign, together with the value provided by the application, to form the full user ID to be located in an LDAP user record. For example, the application provides a user of fred and this field has the value cn, then the LDAP repository will be searched for cn=fred.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

**User password (PASSWORD)**

The password for the LDAP user.

This field is only valid for *\*CRLLDAP* or *\*IDPWLDAP* authentication information objects.

The possible values are:

**\*SAME**

The password is unchanged.

**\*NONE**

The password is blank.

**LDAP-password**

The LDAP user password. The maximum string length is 32 characters.

 **CHGMQMCHL (Change MQ Channel)****Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Change MQ Channel (CHGMQMCHL) command changes the specified attributes of an existing MQ channel definition.

**Note:**

- Changes take effect after the channel is next started.
- For cluster channels, if an attribute can be set on both channels, set it on both and ensure that the settings are identical. If there is any discrepancy between the settings, those that you specify on the cluster receiver channel are likely to be used, This is explained in [Cluster channels](#).
- If you change the XMITQ name or the CONNAME, you must reset the sequence number at both ends of the channel. (See “RESET CHANNEL (reset message sequence number for a channel)” on page 895 for information about the SEQNUM parameter.)

**Parameters**

<i>Table 221. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<a href="#">CHLNAME</a>	Channel name	Character value	Required, Key, Positional 1
<a href="#">MQMNAME</a>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 2
<a href="#">CHLTYPE</a>	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Optional, Key, Positional 3
<a href="#">TRPTYPE</a>	Transport type	*LU62, *TCP, <b>*SAME</b>	Optional, Positional 4
<a href="#">TEXT</a>	Text 'description'	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 5
<a href="#">TGTMQMNAME</a>	Target Queue Manager	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 6
<a href="#">CONNAME</a>	Connection name	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 7
<a href="#">TPNAME</a>	Transaction Program Name	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 8
<a href="#">MODENAME</a>	Mode Name	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 9
<a href="#">TMQNAME</a>	Transmission queue	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 10
<a href="#">MCANAME</a>	Message channel agent	Single values: <b>*SAME</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 11
	Qualifier 1: Message channel agent	Name	
	Qualifier 2: Library	<i>Name</i> , <b>*CURLIB</b>	
<a href="#">MCAUSRID</a>	Message channel agent user ID	<i>Character value</i> , *NONE, *PUBLIC, <b>*SAME</b>	Optional, Positional 12
<a href="#">MCATYPE</a>	Message channel agent Type	*PROCESS, *THREAD, <b>*SAME</b>	Optional, Positional 13
<a href="#">BATCHINT</a>	Batch Interval	0-999999999, <b>*SAME</b>	Optional, Positional 14
<a href="#">BATCHSIZE</a>	Batch size	1-9999, <b>*SAME</b>	Optional, Positional 15

Table 221. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>DSCITV</u>	Disconnect interval	0-999999, <b>*SAME</b>	Optional, Positional 16
<u>SHORTTMR</u>	Short retry interval	0-999999999, <b>*SAME</b>	Optional, Positional 17
<u>SHORTRTY</u>	Short retry count	0-999999999, <b>*SAME</b>	Optional, Positional 18
<u>LONGTMR</u>	Long retry interval	0-999999999, <b>*SAME</b>	Optional, Positional 19
<u>LONGRTY</u>	Long retry count	0-999999999, <b>*SAME</b>	Optional, Positional 20
<u>SCYEXIT</u>	Security exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values: <i>Qualified object name</i>	Optional, Positional 21
	Qualifier 1: Security exit	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	
<u>CSCYEXIT</u>	Security exit	Character value, <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 22
<u>SCYUSRDATA</u>	Security exit user data	Character value, <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 23
<u>SNDEXIT</u>	Send exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 24
	Qualifier 1: Send exit	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	
<u>CSNDEXIT</u>	Send exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Character value</i>	Optional, Positional 25
<u>SNDUSRDATA</u>	Send exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 26
<u>RCVEXIT</u>	Receive exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 27
	Qualifier 1: Receive exit	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	
<u>CRCVEXIT</u>	Receive exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Character value</i>	Optional, Positional 28
<u>RCVUSRDATA</u>	Receive exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 29

Table 221. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MSGEXIT</u>	Message exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 30
	Qualifier 1: Message exit	Name	
	Qualifier 2: Library	<i>Name</i> , <b>*CURLIB</b>	
<u>MSGUSRDATA</u>	Message exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 31
<u>MSGRTYEXIT</u>	Message retry exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values: <i>Qualified object name</i>	Optional, Positional 32
	Qualifier 1: Message retry exit	Name	
	Qualifier 2: Library	<i>Name</i> , <b>*CURLIB</b>	
<u>MSGRTYDATA</u>	Message retry exit data	<i>Character value</i> , <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 33
<u>MSGRTYNBR</u>	Number of message retries	0-999999999, <b>*SAME</b>	Optional, Positional 34
<u>MSGRTYITV</u>	Message retry interval	0-999999999, <b>*SAME</b>	Optional, Positional 35
<u>CVTMSG</u>	Convert message	<b>*YES</b> , <b>*NO</b> , <b>*SAME</b>	Optional, Positional 36
<u>PUTAUT</u>	Put authority	<b>*DFT</b> , <b>*CTX</b> , <b>*SAME</b>	Optional, Positional 37
<u>SEQNUMWRAP</u>	Sequence number wrap	100-999999999, <b>*SAME</b>	Optional, Positional 38
<u>MAXMSGLEN</u>	Maximum message length	0-104857600, <b>*SAME</b>	Optional, Positional 39
<u>HRTBTINTVL</u>	Heartbeat interval	0-999999999, <b>*SAME</b>	Optional, Positional 40
<u>NPMSPEED</u>	Non Persistent Message Speed	<b>*FAST</b> , <b>*NORMAL</b> , <b>*SAME</b>	Optional, Positional 41
<u>CLUSTER</u>	Cluster Name	<i>Character value</i> , <b>*NONE</b> , <b>*SAME</b>	Optional, Positional 42
<u>CLUSNL</u>	Cluster Name List	<i>Character value</i> , <b>*NONE</b> , <b>*SAME</b>	Optional, Positional 43
<u>NETPRTY</u>	Network Connection Priority	0-9, <b>*SAME</b>	Optional, Positional 44

Table 221. Command parameters (continued)

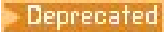
Keyword	Description	Choices	Notes
<a href="#">SSLCIPH</a>	TLS CipherSpec	Supported CipherSpecs are listed here: <a href="#">CipherSpecs you can use with IBM MQ TLS support</a> .   <b>Deprecated</b> CipherSpecs that you can re-enable if necessary are listed here: <a href="#">Deprecated CipherSpecs</a> .	Optional, Positional 45
<a href="#">SSLCAUTH</a>	TLS Client Authentication	*REQUIRED, *OPTIONAL, <b>*SAME</b>	Optional, Positional 46
<a href="#">SSLPEER</a>	TLS Peer name	Character value, *NONE, <b>*SAME</b>	Optional, Positional 47
<a href="#">LOCLADDR</a>	Local communication address	Character value, *NONE, <b>*SAME</b>	Optional, Positional 48
<a href="#">BATCHHB</a>	Batch Heartbeat Interval	0-999999999, <b>*SAME</b>	Optional, Positional 49
<a href="#">USERID</a>	Task user identifier	Character value, *NONE, <b>*SAME</b>	Optional, Positional 50
<a href="#">PASSWORD</a>	Password	Character value, *NONE, <b>*SAME</b>	Optional, Positional 51
<a href="#">KAINT</a>	Keep Alive Interval	0-99999, <b>*SAME</b> , *AUTO	Optional, Positional 52
<a href="#">COMPHDR</a>	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, <b>*SAME</b>	Optional, Positional 53
<a href="#">COMPMSG</a>	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST, <b>*SAME</b>	Optional, Positional 54
<a href="#">MONCHL</a>	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, <b>*SAME</b>	Optional, Positional 55
<a href="#">STATCHL</a>	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, <b>*SAME</b>	Optional, Positional 56
<a href="#">CLWLRANK</a>	Cluster Workload Rank	0-9, <b>*SAME</b>	Optional, Positional 57
<a href="#">CLWLPRTY</a>	Cluster Workload Priority	0-9, <b>*SAME</b>	Optional, Positional 58
<a href="#">CLWLWGHT</a>	Cluster Channel Weight	1-99, <b>*SAME</b>	Optional, Positional 59
<a href="#">SHARECNV</a>	Sharing Conversations	0-999999999, <b>*SAME</b>	Optional, Positional 60
<a href="#">PROPCTL</a>	Property Control	*COMPAT, *NONE, *ALL, <b>*SAME</b>	Optional, Positional 61
<a href="#">MAXINST</a>	Maximum Instances	0-999999999, <b>*SAME</b>	Optional, Positional 62
<a href="#">MAXINSTC</a>	Maximum Instances Per Client	0-999999999, <b>*SAME</b>	Optional, Positional 63



Table 221. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>CLNTWGHT</u>	Client Channel Weight	0-99, <b>*SAME</b>	Optional, Positional 64
<u>AFFINITY</u>	Connection Affinity	*PREFERRED, *NONE, <b>*SAME</b>	Optional, Positional 65
<u>BATCHLIM</u>	Batch Data Limit	0-999999, <b>*SAME</b>	Optional, Positional 66
<u>DFTRECON</u>	Default client reconnection	*NO, *YES, *QMGR, *DISABLED, <b>*SYSDFTCHL</b>	Optional, Positional 67

### Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

#### channel-name

Specify the channel name.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

#### message-queue-manager-name

The name of a message queue manager.

### Channel type (CHLTYPE)

Specifies the type of the channel being changed.

The possible values are:

#### \*SDR

Sender channel

#### \*SVR

Server channel

#### \*RCVR

Receiver channel

#### \*RQSTR

Requester channel

#### \*SVRCN

Server-connection channel

#### \*CLUSSDR

Cluster-sender channel

#### \*CLUSRCVR

Cluster-receiver channel

#### \*CLTCN

Client-connection channel

## Transport type (TRPTYPE)

Specifies the transmission protocol.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*LU62**

SNA LU 6.2.

### **\*TCP**

Transmission Control Protocol / Internet Protocol (TCP/IP).

## Text 'description' (TEXT)

Specifies text that briefly describes the channel definition.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*BLANK**

The text is set to a blank string.

### **description**

Specify no more than 64 characters enclosed in apostrophes.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

## Target Queue Manager (TGTMQMNAME)

Specifies the name of the target queue manager.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The name of the target queue manager for a client connection channel (CHLTYPE) \*CLTCN is unspecified.

### **message-queue-manager-name**

The name of the target message queue manager for a client connection channel (CHLTYPE) \*CLTCN.

For other channel types this parameter must not be specified.

## Connection name (CONNAME)

Specifies the name of the machine to connect.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The connection name is blank.

### **connection-name**

Specify the connection name as required by the transmission protocol:

- For \*LU62, specify the name of the CSI object.

- For \*TCP, specify either the host name, or the network address of the remote machine (or the local machine for cluster-receiver channels). This can be followed by an optional port number enclosed in parentheses.

**Multi** On [Multiplatforms](#), the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

```
(1415)
```

The generated **CONNNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

Where a port is not specified the default port 1414 is assumed.

For cluster-receiver channels the connection name relates to the local queue manager, and for other channels it relates to the target queue manager.

This parameter is required for channels with channel type (CHLTYPE) of \*SDR, \*RQSTR, \*CLTCN and \*CLUSSDR. It is optional for \*SVR and \*CLUSRCVR channels, and is not valid for \*RCVR or \*SVRCN channels.

## Transaction Program Name (TPNAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2 only.

This parameter must be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it must be set to blanks. The name is taken instead from the CPI-C Communications Side Object.

This parameter is not valid for channels with a CHLTYPE defined as \*RCVR.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **\*NONE**

No transaction program name is specified.

### **\*BLANK**

The transaction program name is taken from CPI-C Communications Side Object. The side object name must be specified in the CONNAME parameter.

### **transaction-program-name**

Specify the SNA transaction program name.

## Mode Name (MODENAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2. If TRPTYPE is not defined as LU 6.2 the data is ignored and no error message is issued.

If specified, the value must be set to the SNA mode name, unless the CONNAME contains a side-object name, in which case it must be set to blanks. The name is then taken from the CPI-C Communications Side Object.

This parameter is not valid for channels with CHLTYPE defined as \*RCVR or \*SVRCONN.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

**\*NONE**

No mode name is specified.

**\*BLANK**

Name will be taken from the CPI-C Communications Side Object. This must be specified in the CONNAME parameter.

**SNA-mode-name**

Specify the SNA Mode Name

**Transmission queue (TMQNAME)**

Specifies the name of the transmission queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**transmission-queue-name**

Specify the name of the transmission queue. A transmission queue name is required if the CHLTYPE is defined as \*SDR or \*SVR.

For other channel types this parameter must not be specified.

**Message channel agent (MCANAME)**

This parameter is reserved and should not be used.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The MCA program name is blank.

This parameter cannot be specified if the CHLTYPE is defined as \*RCVR, \*SVRCN, or \*CLTCN.

**Message channel agent user ID (MCAUSRID)**

Specifies the message channel agent user identifier which is to be used by the message channel agent for authorization to access MQ resources, including (if PUTAUT is \*DFT) authorization to put the message to the destination queue for receiver or requester channels.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The message channel agent uses its default user identifier.

**\*PUBLIC**

Uses the public authority.

**mca-user-identifier**

Specify the user identifier to be used.

This parameter cannot be specified for a channel type (CHLTYPE) of \*CLTCN.

**Message channel agent Type (MCATYPE)**

Specifies whether the message channel agent program should run as a thread or as a process.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*PROCESS**

The message channel agent runs as a separate process.

**\*THREAD**

The message channel agent runs as a separate thread.

This parameter can only be specified for channels with CHLTYPE defined as \*SDR, \*SVR, \*RQSTR, \*CLUSSDR or \*CLUSRCVR.

**Batch Interval (BATCHINT)**

The minimum amount of time, in milliseconds, that a channel will keep a batch open.

The batch is terminated by which ever of the following occurs first: BATCHSZ messages have been sent, BATCHLIM bytes have been sent, or the transmission queue is empty and BATCHINT is exceeded.

The default value is 0, which means that the batch is terminated as soon as the transmission queue becomes empty (or the BATCHSZ limit is reached).

The value must be in the range 0 through 999999999.

This parameter is valid for channels with CHLTYPE defined as \*SDR, \*SVR, \*CLUSSDR, or \*CLUSRCVR.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**batch-interval**

Specify a value ranging from 0 through 999999999.

**Batch size (BATCHSIZE)**

Specifies the maximum number of messages that can be sent down a channel before a checkpoint is taken.

The possible values are:

**\*SAME**

The attribute is unchanged.

**batch-size**

Specify a value ranging from 1 through 9999.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Disconnect interval (DSCITV)**

Specifies the disconnect interval, which defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before closing the channel.

The possible values are:

**\*SAME**

The attribute is unchanged.

**disconnect-interval**

Specify a value ranging from 0 through 999999.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR or \*CLTCN.

## Short retry interval (SHORTTMR)

Specifies the short retry wait interval for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the interval between attempts to establish a connection to the remote machine.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **short-retry-interval**

Specify a value ranging from 0 through 999999999.

## Short retry count (SHORTRTY)

Specifies the short retry count for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of attempts that are made to establish a connection to the remote machine, at intervals specified by SHORTTMR, before the (normally longer) LONGRTY and LONGTMR are used.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **short-retry-count**

Specify a value ranging from 0 through 999999999. A value of 0 means that no retries are allowed.

## Long retry interval (LONGTMR)

Specifies the long retry wait interval for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **long-retry-interval**

Specify a value in the range 0 through 999999999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this are treated as 999999.

## Long retry count (LONGRTY)

Specifies the long retry count for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by LONGTMR, after the count specified by SHORTRTY has been exhausted. An error message is logged if the connection is not established after the defined number of attempts.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **long-retry-count**

Specify a value in the range 0 through 999999999. A value of 0 means that no retries are allowed.

## Security exit (SCYEXIT)

Specifies the name of the program to be called as the security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.

- On receipt of a response to a security message flow.

Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The security exit program is not invoked.

### **security-exit-name**

Specify the name of the security exit program.

### **library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

## Security exit (CSCYEXIT)

Specifies the name of the program to be called as the client security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.

- On receipt of a response to a security message flow.

Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The client security exit program is not invoked.

### **security-exit-name**

Specify the name of the client security exit program.

## Security exit user data (SCYUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the security exit program.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The user data for the security exit program is not specified.

### **security-exit-user-data**

Specify the user data for the security exit.

## Send exit (SNDEXIT)

Specifies the entry point of the program to be called as the send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The send exit program is not invoked.

**send-exit-name**

Specify the name of the send exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

## Send exit (CSNDEXIT)

Specifies the entry point of the program to be called as the client send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The client send exit program is not invoked.

**send-exit-name**

Specify the name of the client send exit program.

## Send exit user data (SNDUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the send exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the send exit program is not specified.

**send-exit-user-data**

Specify the user data for the send exit program.

## Receive exit (CRCVEXIT)

Specifies the entry point of the program to be called as the client receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The client receive exit program is not invoked.



**receive-exit-name**

Specify the name of the client receive exit program.

**Receive exit (RCVEXIT)**

Specifies the entry point of the program to be called as the receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The receive exit program is not invoked.

**receive-exit-name**

Specify the name of the receive exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

**Receive exit user data (RCVUSRDATA)**

Specifies a maximum of 32 characters of user data that is passed to the receive exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the receive exit program is not specified.

**receive-exit-user-data**

Specify a maximum of 32 characters of user data for the receive exit.

**Message exit (MSGEXIT)**

Specifies the entry point of the program to be called as the message exit. If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The message exit program is not invoked.

**message-exit-name**

Specify the name of the message exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Message exit user data (MSGUSRDATA)**

Specifies user data that is passed to the message exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the message exit program is not specified.

**message-exit-user-data**

Specify a maximum of 32 characters of user data that is passed to the message exit program.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Message retry exit (MSGRTYEXIT)**

Specifies the entry point of the program to be called as the message retry exit.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The message retry exit program is not invoked.

**message-retry-exit-name**

Specify the name of the message retry exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

**Message retry exit data (MSGRTYDATA)**

Specifies user data that is passed to the message retry exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the message retry exit program is not specified.

**message-retry-exit-user-data**

Specify a maximum of 32 characters of user data that is passed to the message retry exit program.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

**Number of message retries (MSGRTYNBR)**

Specifies the number of times the channel will retry before it decides it cannot deliver the message.

This parameter is used by the channel as an alternative to a message retry exit when MSGRTYEXIT is defined as \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**message-retry-number**

Specify a value ranging from 0 through 999999999. A value of 0 indicates no retries will be performed.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

### **Message retry interval (MSGRTYITV)**

Specifies the minimum interval of time that must pass before the channel can retry the MQPUT operation. This time is in milliseconds.

This parameter is used by the channel as an alternative to a message retry exit when MSGRTYEXIT is defined as \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**message-retry-number**

Specify a value ranging from 0 through 999999999. A value of 0 indicates that the retry will be performed as soon as possible.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

### **Convert message (CVTMSG)**

Specifies whether the application data in the message should be converted before the message is transmitted.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*YES**

The application data in the message is converted before sending.

**\*NO**

The application data in the message is not converted before sending.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

### **Put authority (PUTAUT)**

Specifies whether the user identifier in the context information associated with a message is used to establish authority to put the message on the destination queue. This applies only to receiver and requester (\*CLUSRCVR, \*RCVR and \*RQSTR) channels.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*DFT**

No authority check is made before the message is put on the destination queue.

**\*CTX**

The user identifier in the message context information is used to establish authority to put the message.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

### **Sequence number wrap (SEQNUMWRAP)**

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

**Note:** The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

The possible values are:

**\*SAME**

The attribute is unchanged.

**sequence-number-wrap-value**

Specify a value ranging from 100 through 999999999.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

### **Maximum message length (MAXMSGLEN)**

Specifies the maximum message length that can be transmitted on the channel. This is compared with the value for the remote channel and the actual maximum is the lower of the two values.

The possible values are:

**\*SAME**

The attribute is unchanged.

**maximum-message-length**

Specify a value ranging from 0 through 104857600. A value of 0 indicates that the maximum length is unlimited.

### **Heartbeat interval (HRTBTINTVL)**

Specifies the time, in seconds, between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. The heartbeat exchange gives the receiving MCA the opportunity to quiesce the channel. This applies only to sender, server, cluster sender and cluster receiver (\*SDR, \*SVR, \*CLUSDR and \*CLUSRCVR) channels.

The possible values are:

**\*SAME**

The attribute is unchanged.

**heart-beat-interval**

Specify a value ranging from 0 through 999999999. A value of 0 means that no heartbeat exchanges are to take place.

### **Non Persistent Message Speed (NPMSPEED)**

Specifies whether the channel supports fast non persistent messages.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*FAST**

The channel supports fast non persistent messages.

**\*NORMAL**

The channel does not support fast non persistent messages.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

### **Cluster Name (CLUSTER)**

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

This parameter is valid only for \*CLUSDR and \*CLUSRCVR channels. If the CLUSNL parameter is non-blank, this parameter must be blank.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*NONE**

No cluster name is specified.

**cluster-name**

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

## Cluster Name List (CLUSNL)

The name of the namelist that specifies a list of clusters to which the channel belongs

This parameter is valid only for \*CLUSDR and \*CLUSRCVR channels. If the CLUSTER parameter is non-blank, this parameter must be blank.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*NONE**

No cluster namelist is specified.

**cluster-name-list**

The name of the namelist specifying a list of clusters to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

## Network Connection Priority (NETPTY)

The priority for the network connection. Distributed queuing chooses the path with the highest priority if there are multiple paths available. The value must be in the range between 0 and 9 where 0 is the lowest priority.

This parameter is valid only for \*CLUSRCVR channels.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**network-connection-priority**

Specify a value ranging from 0 through 9 where 0 is the lowest priority.

## TLS CipherSpec (SSLCIPH)


SSLCIPH specifies the CipherSpec used in TLS channel negotiation. The possible values are:

**\*SAME**

The value of this attribute does not change.

**cipherspec**

The name of the CipherSpec.

**Note:**  From IBM MQ 8.0.0 Fix Pack 2, the SSLv3 protocol and the use of some IBM MQ CipherSpecs is deprecated. For more information, see [Deprecated CipherSpecs](#).

## TLS Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel carries out client authentication over TLS. The parameter is used only for channels with SSLCIPH specified.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*REQUIRED**

Client authentication is required.

**\*OPTIONAL**

Client authentication is optional.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*CLTCN or \*CLUSSDR.

**TLS Peer name (SSLPEER)**

SSLPEER specifies the X500 peer name used in TLS channel negotiation. The possible values are:

**\*SAME**

The value of this attribute does not change.

**x500peername**

The X500 peer name to use.

**Note:** An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both SSLPEER on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns in order to connect. For more information, see [Channel authentication records](#).

**Local communication address (LOCLADDR)**

Specifies the local communication address for the channel.

This parameter is only valid for \*SDR, \*SVR, \*RQSTR, \*CLUSSDR, \*CLUSRCVR and \*CLTCN channels.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The connection is blank.

**local-address**

Only valid for transport type TCP/IP. Specify the optional IP address and optional port or port range used for outbound TCP/IP communications. The format is:

```
LOCLADDR([ip-addr][low-port[,high-port]][, [ip-addr][low-port[,high-port]])])
```

**Batch Heartbeat Interval (BATCHEB)**

The time in milliseconds used to determine whether batch heartbeating occurs on this channel. Batch heartbeating allows channels to determine whether the remote channel instance is still active before going indoubt. A batch heartbeat will occur if a channel MCA has not communicated with the remote channel within the specified time.

The possible values are:

**\*SAME**

The attribute is unchanged.

**batch-heartbeat-interval**

Specify a value ranging from 0 through 999999999. A value of 0 indicates that batch heartbeating is not to be used.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

## Task user identifier (USERID)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

This parameter is valid only for channels with a channel type (CHLTYPE) of \*SDR, \*SVR, \*RQSTR, \*CLTCN or \*CLUSSDR.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **\*NONE**

No user identifier is specified.

### **user-identifier**

Specify the task user identifier.

## Password (PASSWORD)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

This parameter is valid only for channels with a channel type (CHLTYPE) of \*SDR, \*SVR, \*RQSTR, \*CLTCN or \*CLUSSDR.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **\*NONE**

No password is specified.

### **password**

Specify the password.

## Keep Alive Interval (KAINT)

Specifies the keep alive timing interval for this channel.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*AUTO**

The keep alive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated HBINT is greater than 0, keep alive interval is set to that value plus 60 seconds.
- If the negotiated HBINT is 0, the value used is that specified by the KEEPALIVEOPTIONS statement in the TCP profile configuration data set.

### **keep-alive-interval**

Specify a value ranging from 0 through 99999.

## Header Compression (COMPHDR)

The list of header data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (\*SDR, \*SVR, \*CLUSSDR, \*CLUSRCVR and \*CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No header data compression is performed.

**\*SYSTEM**

Header data compression is performed.

## Message Compression (COMPMSG)

The list of message data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (\*SDR, \*SVR, \*CLUSSDR, \*CLUSRCVR and \*CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No message data compression is performed.

**\*RLE**

Message data compression is performed using run-length encoding.

**\*ZLIBFAST**

Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

**\*ZLIBHIGH**

Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

**\*ANY**

Any compression technique supported by the queue manager can be used. This option is only valid for channel types receiver, requester and server connection (\*RCVR, \*RQSTR and \*SVRCN).

## Channel Monitoring (MONCHL)

Controls the collection of online monitoring data.

Online monitoring data is not collected when the queue manager attribute MONCHL is set to \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

The collection of online monitoring data is inherited from the setting of the queue manager attribute MONCHL.

**\*OFF**

Online Monitoring Data collection for this channel is switched off.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.



**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for a channel type (CHLTYPE) of \*CLTCN.

**Channel Statistics (STATCHL)**

Controls the collection of statistics data.

Statistics data is not collected when the queue manager attribute STATCHL is set to \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

Statistics data collection is based upon the setting of the queue manager attribute STATCHL.

**\*OFF**

Statistics data collection for this channel is disabled.

**\*LOW**

Statistics data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Statistics data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Statistics data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Cluster Workload Rank (CLWLRANK)**

Specifies the cluster workload rank of the channel.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-rank**

The cluster workload rank of the channel in the range 0 through 9.

**Cluster Workload Priority (CLWLPRTY)**

Specifies the cluster workload priority of the channel.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-priority**

The cluster workload priority of the channel in the range 0 through 9.

**Cluster Channel Weight (CLWLWGHT)**

Specifies the cluster workload weight of the channel.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-weight**

The cluster workload weight of the channel in the range 1 through 99.

## Sharing Conversations (SHARECNV)

Specifies the maximum the number of conversations which can be shared over a particular TCP/IP client channel instance (socket).

This parameter is valid for channels with CHLTYPE defined as \*CLTCN or \*SVRCN.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **0**

Specifies no sharing of conversations over a TCP/IP socket. The channel instance runs in a mode prior to that of IBM WebSphere MQ 7.0, with regard to:

- Administrator stop-quiesce
- Heartbeating
- Read ahead

### **1**

Specifies no sharing of conversations over a TCP/IP socket. Client heartbeating and read ahead are available, whether in an MQGET call or not, and channel quiescing is more controllable.

### **shared-conversations**

The number of shared conversations in the range 2 through 999999999.

This parameter is only valid for client-connection and server-connection channels.

**Note:** If the client-connection SHARECNV value does not match the server-connection SHARECNV value, the lower of the two values is used.

## Property Control (PROPCTL)

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor).

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*COMPAT**

If the message contains a property with a prefix of "mcd.", "jms.", "usr." or "mqext." then all optional message properties, except those in the message descriptor (or extension) will be placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.

### **\*NONE**

All properties of the message, except those in the message descriptor (or extension), will be removed from the message before the message is sent to the remote queue manager.

### **\*ALL**

All properties of the message will be included with the message when it is sent to the remote queue manager. The properties, except those in the message descriptor (or extension), will be placed in one or more MQRFH2 headers in the message data.

## Maximum Instances (MAXINST)

Specifies the maximum number of clients that can simultaneously connect to the queue manager via this server-connection channel object.

This attribute is valid only for server-connection channels.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **maximum-instances**

The maximum number of simultaneous instances of the channel in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

## **Maximum Instances Per Client (MAXINSTC)**

Specifies the maximum number of simultaneous instances of an individual server-connection channel which can be started from a single client.

In this context, multiple client connections originating from the same remote network address are considered to be a single client.

This attribute is valid only for server-connection channels.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **maximum-instances-per-client**

The maximum number of simultaneous instances of the channel which can be in the started from a single client in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running from individual clients, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

## **Client Channel Weight (CLNTWGHT)**

The client channel weighting attribute is used so client channel definitions can be selected at random based on their weighting when more than one suitable definition is available.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **client-channel-weight**

The client channel weight in the range 0 through 99.

## **Connection Affinity (AFFINITY)**

The channel affinity attribute is used so client applications that connect multiple times using the same queue manager name can choose whether to use the same client channel definition for each connection.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*PREFERRED**

The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions based on the weighting with any applicable CLNTWGHT(0) definitions first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non CLNTWGHT(0) definitions are moved to the end of the list. CLNTWGHT(0) definitions remain at the start of the list and are selected first for each connection.

### **\*NONE**

The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable CLNTWGHT(0) definitions selected first in alphabetical order.

## Batch Data Limit (BATCHLIM)

The limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point. A sync point is taken after the message that caused the limit to be reached has flowed across the channel. A value of zero in this attribute means that no data limit is applied to batches over this channel.

The batch is terminated when one of the following conditions is met:

- **BATCHSZ** messages have been sent.
- **BATCHLIM** bytes have been sent.
- The transmission queue is empty and **BATCHINT** is exceeded.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

The value must be in the range 0 - 999999. The default value is 5000.

The **BATCHLIM** parameter is supported on all platforms.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **batch-data-limit**

Specify a value ranging from 0 through 999999.

This parameter can only be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLUSSDR, or \*CLUSRCVR.

## Default client reconnection (DFTRECON)

Specifies whether a client connection automatically reconnects a client application if its connection breaks.

### **\*SAME**

The value of this attribute does not change.

### **\*NO**

Unless overridden by **MQCONN**, the client is not reconnected automatically.

### **\*YES**

Unless overridden by **MQCONN**, the client reconnects automatically.

### **\*QMGR**

Unless overridden by **MQCONN**, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as MQCNO\_RECONNECT\_Q\_MGR.

### **\*DISABLED**

Reconnection is disabled, even if requested by the client program using the **MQCONN** MQI call.

This parameter is specified for a client connection channel, (CHLTYPE) \*CLTCN



## CHGMQMJRN (Change Queue Manager Journal)

### **Where allowed to run**

All environments (\*ALL)

### **Threadsafe**

Yes

The Change Queue Manager Journal command (CHGMQMJRN) changes a queue manager journal. This command can be used, for example, to change the type of remote journal replication used for a backup or multi-instance queue manager.

## Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 1
<u>JRN</u>	Queue Manager Journal	Character value, <b>*DFT</b>	Optional, Positional 2
<u>RMTJRNRDB</u>	Remote Relational Database	Character value	Optional, Positional 3
<u>RMTJRNSTS</u>	Remote Journal Status	<b>*ACTIVE</b> , <b>*INACTIVE</b>	Optional, Positional 4
<u>RMTJRNDLV</u>	Remote Journal Delivery	<b>*SYNC</b> , <b>*ASYN</b> C	Optional, Positional 5
<u>RMTJRNTIMO</u>	Remote Journal Sync. Timeout	1-3600, <b>*DFT</b>	Optional, Positional 6

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager associated with the journal.

#### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### Queue Manager Journal (JRN)

Specifies the journal name to create.

The possible values are:

#### **\*DFT**

The journal name is chosen by the system. If a local journal already exists for the queue manager on this system - the existing local journal name is used, otherwise a unique name is generated of the format AMQxJRN where x is a character in the range 'A - Z'.

#### journal-name

Specify the name of the journal. The name can contain up to 10 characters. Journal receiver names will be derived from this journal name by truncating at the 4th character (or at the last character if the journal name is shorter than 4 characters) and appending zeroes. If the local queue manager library already contains a local journal, its name must match that supplied. Only one local journal can exist in a queue manager library. DLTMQM will not remove journal artifacts from a queue manager library unless they are prefixed with "AMQ".

### Remote Relational Database (RMTJRNRDB)

Specifies the name of the relational database directory entry that contains the remote location name of the target system. Use the WRKRDBDIRE command to locate and existing entry or configure a new relational database directory entry for the target system.

#### relational-database-directory-entry

Specify the name of the relational database directory entry. The name can contain up to 18 characters.

### Remote Journal Status (RMTJRNSTS)

Specifies whether the remote journal is ready to receive journal entries from the queue managers local journal.

The possible values are:

**\*ACTIVE**

The remote journal is ready to receive journal entries from the local queue manager journal. Replication of journal entries starts with the oldest local journal receiver required to perform a full media recovery and queue manager restart. If these recovery points do not exist, replication starts with the currently attached local journal receiver.

**\*INACTIVE**

The remote journal is not ready to receive journal entries from the local queue manager journal.

**Remote Journal Delivery (RMTJRNDLV)**

Specifies whether the journal entries are replicated synchronously or asynchronously when the remote journal is activated. Note that this parameter is ignored when RMTJRNSTS(\*INACTIVE) is specified.

The possible values are:

**\*SYNC**

The remote journal is replicated synchronously with the local queue manager journal.

**\*ASYNC**

The remote journal is replicated asynchronously with the local queue manager journal.

**Remote Journal Sync. Timeout (RMTJRNTIMO)**

Specifies the maximum amount of time in seconds to wait for a response from the remote system when using synchronous replication with remote journaling. If a response is not received from the remote system within the timeout period, the remote journal environment will automatically be deactivated. Note that this parameter is ignored when RMTJRNDLV(\*ASYNC) or RMTJRNSTS(\*INACTIVE) are specified.

The possible values are:

**\*DFT**

The system uses the default value of 60 seconds to wait for a response from the remote system.

**1-3600**

Specify the maximum number of seconds to wait for a response from the remote system. Note that this option is only available on IBM i V6R1M0 and later operating systems.


**CHGMQMLSR (Change MQ Listener)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Change MQ Listener (CHGMQMLSR) command changes the specified attributes of an existing MQ listener definition.

**Parameters**

<i>Table 223. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>LSRNAME</u>	Listener name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , *DFT	Optional, Key, Positional 2
<u>TEXT</u>	Text 'description'	<i>Character value</i> , *BLANK, *SAME	Optional, Positional 3

Table 223. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>CONTROL</u>	Listener control	<b>*SAME</b> , *MANUAL, *QMGR, *STARTONLY	Optional, Positional 4
<u>PORT</u>	Port number	0-65535, <b>*SAME</b>	Optional, Positional 5
<u>IPADDR</u>	IP Address	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 6
<u>BACKLOG</u>	Listener backlog	0-999999999, <b>*SAME</b>	Optional, Positional 7

### Listener name (LSRNAME)

The name of the listener definition to be changed.

The possible values are:

#### listener-name

Specify the name of the listener definition. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.

### Text 'description' (TEXT)

Specifies text that briefly describes the listener definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

#### \*SAME

The attribute is unchanged.

#### \*BLANK

The text is set to a blank string.

#### description

Specify no more than 64 characters enclosed in apostrophes.

### Listener control (CONTROL)

Whether the listener starts automatically when the queue manager is started.

The possible values are:

#### \*SAME

The attribute is unchanged.

#### \*MANUAL

The listener is not automatically started or stopped.

#### \*QMGR

The listener is started and stopped as the queue manager is started and stopped.

**\*STARTONLY**

The listener is started as the queue manager is started, but is not automatically stopped when the queue manager is stopped.

**Port number (PORT)**

The port number to be used by the listener.

The possible values are:

**\*SAME**

The attribute is unchanged.

**port-number**

The port number to be used.

**IP Address (IPADDR)**

The IP address to be used by the listener.

The possible values are:

**\*SAME**

The attribute is unchanged.

**ip-addr**

The IP address to be used.

**Listener backlog (BACKLOG)**

The number of concurrent connection requests the listener supports.

The possible values are:

**\*SAME**

The attribute is unchanged.

**backlog**

The number of concurrent connection requests supported.

 **CHGMQMNL (Change MQ Namelist)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Change MQ Namelist (CHGMQMNL) command changes a list of names in the namelist specified on the selected local queue manager.

**Parameters**

<i>Table 224. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>NAMELIST</u>	Namelist	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 2
<u>TEXT</u>	Text 'description'	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 3



Table 224. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>NAMES</u>	List of Names	Values (up to 256 repetitions): <i>Character value</i> , *BLANKS, *SAME, *NONE	Optional, Positional 4

### Namelist (NAMELIST)

The name of the namelist to be changed.

#### namelist

Specify the name of the namelist. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

The default queue manager is used.

#### message-queue-manager-name

Specify the name of the queue manager.

### Text 'description' (TEXT)

Specifies text that briefly describes the namelist.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

#### \*SAME

The attribute is unchanged.

#### description

Specify no more than 64 characters enclosed in apostrophes.

### List of Names (NAMES)

List of names. This is the list of names to be created. The names can be of any type, but must conform to the rules for naming MQ objects.

#### \*SAME

The attribute is unchanged.

#### namelist

The list to create. An empty list is valid.

## CHGMQMPRC (Change MQ Process)

#### Where allowed to run

All environments (\*ALL)

#### Threadsafe

Yes

The Change MQ Process (CHGMQMPRC) command changes the specified attributes of an existing MQ process definition.

## Parameters

Table 225. Command parameters			
Keyword	Description	Choices	Notes
<u>PRCNAME</u>	Process name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Key, Positional 2
<u>TEXT</u>	Text 'description'	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 3
<u>APPTYPE</u>	Application type	Integer, <b>*DEF</b> , *CICS, *UNIX, *OS400, *WINDOWS, *WINDOWS_NT,	Optional, Positional 4
<u>APPID</u>	Application identifier	Character value, <b>*SAME</b>	Optional, Positional 5
<u>USRDATA</u>	User data	Character value, <b>*SAME</b> , *NONE	Optional, Positional 6
<u>ENVDATA</u>	Environment data	Character value, <b>*SAME</b> , *NONE	Optional, Positional 7

### Process name (PRCNAME)

The name of the process definition to be changed.

The possible values are:

#### process-name

Specify the name of the process definition. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.

### Text 'description' (TEXT)

Specifies text that briefly describes the process definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

#### **\*SAME**

The attribute is unchanged.

#### **\*BLANK**

The text is set to a blank string.

#### description

Specify no more than 64 characters enclosed in apostrophes.

## Application type (APPTYPE)

The type of application started.

The possible values are:

### **\*DEF**

Specifying DEF causes the default application type for the platform at which the command is interpreted to be stored in the process definition. This default cannot be changed by the installation. If the platform supports clients, the default is interpreted as the default application type of the server.

### **\*CICS**

Represents a CICS/400® application.

### **\*UNIX**

Represents a UNIX or Linux application.

### **\*OS400**

Represents an IBM i application.

### **\*WINDOWS**

Represents a Windows application.

### **\*WINDOWS\_NT**

Represents a Windows NT application.

### **integer**

User-defined application type in the range 65536 through 999999999.

## Application identifier (APPID)

Application identifier. This is the name of the application to be started, on the platform for which the command is processing. It is typically a program name and library name.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **application-id**

The maximum length is 256 characters.

## User data (USRDATA)

A character string that contains user information pertaining to the application, as defined by APPID, to start.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The user data is blank.

### **user-data**

Specify up to 128 characters of user data.

## Environment data (ENVDATA)

A character string that contains environment information pertaining to the application, as defined by APPID, to start.

The possible values are:

### **\*SAME**


The attribute is unchanged.

**\*NONE**

The environment data is blank.

**environment-data**

The maximum length is 128 characters.


**CHGMQM (Change MQ Queue)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Change MQ Queue ( **CHGMQM** ) command changes the specified attributes of an existing MQ queue.

**Parameters**

Table 226. Command parameters

Keyword	Description	Choices	Notes
<u>QNAME</u>	Queue name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Key, Positional 2
<u>QTYPE</u>	Queue type	Character value	Optional, Positional 3
<u>FORCE</u>	Force	<b>*NO</b> , *YES	Optional, Positional 4
<u>TEXT</u>	Text 'description'	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 5
<u>PUTENBL</u>	Put enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 6
<u>DFTPTY</u>	Default message priority	0-9, <b>*SAME</b>	Optional, Positional 7
<u>DFTMSGPST</u>	Default message persistence	<b>*SAME</b> , *NO, *YES	Optional, Positional 8
<u>PRCNAME</u>	Process name	Character value, *NONE, <b>*SAME</b>	Optional, Positional 9
<u>TRGENBL</u>	Triggering enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 10
<u>GETENBL</u>	Get enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 11
<u>SHARE</u>	Sharing enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 12
<u>DFTSHARE</u>	Default share option	<b>*SAME</b> , *NO, *YES	Optional, Positional 13
<u>MSGDLYSEQ</u>	Message delivery sequence	<b>*SAME</b> , *PTY, *FIFO	Optional, Positional 14
<u>HDNBKTCNT</u>	Harden backout count	<b>*SAME</b> , *NO, *YES	Optional, Positional 15
<u>TRGTYPE</u>	Trigger type	<b>*SAME</b> , *FIRST, *ALL, *DEPTH, *NONE	Optional, Positional 16
<u>TRGDEPTH</u>	Trigger depth	1-999999999, <b>*SAME</b>	Optional, Positional 17
<u>TRGMSGPTY</u>	Trigger message priority	0-9, <b>*SAME</b>	Optional, Positional 18
<u>TRGDATA</u>	Trigger data	Character value, *NONE, <b>*SAME</b>	Optional, Positional 19

Table 226. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>RTNITV</u>	Retention interval	0-999999999, <b>*SAME</b>	Optional, Positional 20
<u>MAXDEPTH</u>	Maximum queue depth	0-999999999, <b>*SAME</b>	Optional, Positional 21
<u>MAXMSGLEN</u>	Maximum message length	0-104857600, <b>*SAME</b>	Optional, Positional 22
<u>BKTTHLD</u>	Backout threshold	0-999999999, <b>*SAME</b>	Optional, Positional 23
<u>BKTQNAME</u>	Backout requeue name	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 24
<u>INITQNAME</u>	Initiation queue	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 25
<u>USAGE</u>	Usage	<b>*SAME</b> , *NORMAL, *TMQ	Optional, Positional 26
<u>DFNTYPE</u>	Definition type	<b>*SAME</b> , *TEMPDYN, *PERMDYN	Optional, Positional 27
<u>TGTQNAME</u>	Target object	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 28
<u>RMTQNAME</u>	Remote queue	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 29
<u>RMTMQMNAME</u>	Remote Message Queue Manager	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 30
<u>TMQNAME</u>	Transmission queue	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 31
<u>HIGHTHLD</u>	Queue depth high threshold	0-100, <b>*SAME</b>	Optional, Positional 32
<u>LOWTHLD</u>	Queue depth low threshold	0-100, <b>*SAME</b>	Optional, Positional 33
<u>FULLEVT</u>	Queue full events enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 34
<u>HIGHEVT</u>	Queue high events enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 35
<u>LOWEVT</u>	Queue low events enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 36
<u>SRVITV</u>	Service interval	0-999999999, <b>*SAME</b>	Optional, Positional 37
<u>SRVEVT</u>	Service interval events	<b>*SAME</b> , *HIGH, *OK, *NONE	Optional, Positional 38
<u>DISTLIST</u>	Distribution list support	<b>*SAME</b> , *NO, *YES	Optional, Positional 39
<u>CLUSTER</u>	Cluster Name	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 40
<u>CLUSNL</u>	Cluster Name List	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 41
<u>DEFBIND</u>	Default Binding	<b>*SAME</b> , *OPEN, *NOTFIXED, *GROUP	Optional, Positional 42
<u>CLWLRANK</u>	Cluster Workload Rank	0-9, <b>*SAME</b>	Optional, Positional 43
<u>CLWLPRTY</u>	Cluster Workload Priority	0-9, <b>*SAME</b>	Optional, Positional 44

Table 226. Command parameters (continued)

Keyword	Description	Choices	Notes
<a href="#">CLWLUSEQ</a>	Cluster workload queue use	<b>*SAME</b> , *QMGR, *LOCAL, *ANY	Optional, Positional 45
<a href="#">MONQ</a>	Queue Monitoring	<b>*SAME</b> , *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 46
<a href="#">STATQ</a>	Queue Statistics	<b>*SAME</b> , *QMGR, *OFF, *ON	Optional, Positional 47
<a href="#">ACCTQ</a>	Queue Accounting	<b>*SAME</b> , *QMGR, *OFF, *ON	Optional, Positional 48
<a href="#">NPMCLASS</a>	Non Persistent Message Class	<b>*SAME</b> , *NORMAL, *HIGH	Optional, Positional 49
<a href="#">MSGREADAHD</a>	Message Read Ahead	<b>*SAME</b> , *DISABLED, *NO, *YES	Optional, Positional 50
<a href="#">DFTPUTRESP</a>	Default Put Response	<b>*SAME</b> , *SYNC, *ASYNCR	Optional, Positional 51
<a href="#">PROPCTL</a>	Property Control	<b>*SAME</b> , *COMPAT, *NONE, *ALL, *FORCE, *V6COMPAT	Optional, Positional 52
<a href="#">TARGTYPE</a>	Target Type	<b>*SAME</b> , *QUEUE, *TOPIC	Optional, Positional 53
<a href="#">CUSTOM</a>	Custom attribute	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 54
<a href="#">“CLCHNAME” on page 1637</a>	Cluster-sender channel name	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 55
<a href="#">IMGRCOVQ</a>	Queue object attribute	<b>*SAME</b> , *NO, *YES, *QMGR	Optional, Positional 57

### Queue name (QNAME)

The name of the queue to be changed.

The possible values are:

#### **queue-name**

Specify the name of the queue.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

#### **queue-manager-name**

Specify the name of the queue manager.

### Queue type (QTYPE)

Specifies the type of queue that is to be changed.

The possible values are:

**\*ALS**

An alias queue.

**\*LCL**

A local queue.

**\*RMT**

A remote queue.

**\*MDL**

A model queue.

**Force (FORCE)**

Specifies whether the command should be forced to complete when conditions are such that completing the command affects an open queue. The conditions depend on the type of the queue that is being changed:

**Alias Queue**

The TGTQNAME keyword is specified with a queue name and an application has the alias queue open.

**Local Queue**

Either of the following conditions indicate that a local queue will be affected:

- SHARE(\*NO) is specified and more than one application has the local queue open for input.
- The USAGE attribute is changed and one or more applications has the local queue open, or, there are one or more messages on the queue. (The USAGE attribute should not normally be changed while there are messages on the queue; the format of messages changes when they are put on a transmission queue.)

**Remote Queue**

Either of the following conditions indicate that a remote queue will be affected:

- The TMQNAME keyword is specified with a transmission-queue name (or \*NONE) and an application with the remote queue open will be affected by this change.
- Any of the RMTQNAME, RMTMQMNAME or TMQNAME keywords is specified with a queue or queue manager name, and one or more applications has a queue open that resolves through this definition as a queue manager alias.

**Note:** FORCE(\*YES) is not required if this definition is in use as a reply-to queue definition only.

The possible values are:

**\*NO**

The command fails if the relevant conditions are true.

**\*YES**

The command is forced to complete successfully even if the relevant conditions are true.

**Text 'description' (TEXT)**

Specifies text that briefly describes the queue definition.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### **Put enabled (PUTENBL)**

Specifies whether messages can be put on the queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Messages cannot be added to the queue.

**\*YES**

Messages can be added to the queue by authorized applications.

### **Default message priority (DFTPTY)**

Specifies the default priority of messages put on the queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**priority-value**

Specify a value ranging from 0 through 9, where 9 is the highest priority.

### **Default message persistence (DFTMSGPST)**

Specifies the default for message-persistence on the queue. Message persistence determines whether messages are preserved across restarts of the queue manager.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

By default, messages are lost across a restart of the queue manager.

**\*YES**

By default, messages are preserved across a restart of the queue manager.

### **Process name (PRCNAME)**

Specifies the local name of the MQ process that identifies the application that should be started when a trigger event occurs.

The process does not have to be available when the queue is created, but it must be available for a trigger event to occur.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The process name is blank.

**process-name**

Specify the name of the MQ process.

### **Triggering enabled (TRGENBL)**

Specifies whether trigger messages are written to the initiation queue.



**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Triggering is not enabled. Trigger messages are not written to the initiation queue.

**\*YES**

Triggering is enabled. Trigger messages are written to the initiation queue.

### **Get enabled (GETENBL)**

Specifies whether applications are to be permitted to get messages from this queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Applications cannot retrieve messages from the queue.

**\*YES**

Suitably authorized applications can retrieve messages from the queue.

### **Sharing enabled (SHARE)**

Specifies whether multiple instances of applications can open this queue for input simultaneously.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Only a single application instance can open the queue for input.

**\*YES**

More than one application instance can open the queue for input.

### **Default share option (DFTSHARE)**

Specifies the default share option for applications opening this queue for input.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

By default, the open request is for exclusive use of the queue for input.

**\*YES**

By default, the open request is for shared use of the queue for input.

### **Message delivery sequence (MSGDLYSEQ)**

Specifies the message delivery sequence.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*PTY**

Messages are delivered in first-in-first-out (FIFO) order within priority.

**\*FIFO**

Messages are delivered in FIFO order regardless of priority.

**Harden backout count (HDNBKTCNT)**

Specifies whether the count of backed out messages is saved (hardened) across restarts of the message queue manager.

**Note:** On IBM MQ for IBM i the count is ALWAYS hardened, regardless of the setting of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

The backout count is not hardened.

**\*YES**

The backout count is hardened.

**Trigger type (TRGTYPE)**

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*FIRST**

When the number of messages on the queue goes from 0 to 1.

**\*ALL**

Every time a message arrives on the queue.

**\*DEPTH**

When the number of messages on the queue equals the value of the TRGDEPTH attribute.

**\*NONE**

No trigger messages are written.

**Trigger depth (TRGDEPTH)**

Specifies, for TRGTYPE(\*DEPTH), the number of messages that initiate a trigger message to the initiation queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**depth-value**

Specify a value ranging from 1 through 999999999.

**Trigger message priority (TRGMSGPTY)**

Specifies the minimum priority that a message must have before it can result in a trigger event.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**priority-value**

Specify a value ranging from 0 through 9, where 9 is the highest priority.

### **Trigger data (TRGDATA)**

Specifies up to 64 characters of user data that the queue manager includes in the trigger message. This data is made available to the monitoring application that processes the initiation queue, and to the application started by the monitor.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No trigger data is specified.

**trigger-data**

Specify up to 64 characters enclosed in apostrophes. For a transmission queue you can use this parameter to specify the name of the channel to be started.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### **Retention interval (RTNITV)**

Specifies the retention interval. This interval is the number of hours for which the queue might be needed, based on the date and time when the queue was created.

This information is available to a housekeeping application or an operator and can be used to determine when a queue is no longer required.

**Note:** The message queue manager does not delete queues, nor does it prevent your queues from being deleted if their retention interval has not expired. It is your responsibility to take any required action.

The possible values are:

**\*SAME**

The attribute is unchanged.

**interval-value**

Specify a value ranging from 0 through 999999999.

### **Maximum queue depth (MAXDEPTH)**

Specifies the maximum number of messages allowed on the queue. However, other factors can cause the queue to be treated as full; for example, it appears to be full if there is no storage available for a message.

**Note:** If this value is subsequently reduced by using the CHGMQM command, any messages that are on the queue remain intact even if they cause the new maximum to be exceeded.

The possible values are:

**\*SAME**

The attribute is unchanged.

**depth-value**

Specify a value ranging from 0 through 999999999.

## Maximum message length (MAXMSGLEN)

Specifies the maximum length for messages on the queue.

**Note:** If this value is subsequently reduced by using the CHGMQM command, any messages that are on the queue remain intact even if they exceed the new maximum length.

Applications might use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore change the value only if you know this will not cause an application to operate incorrectly.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **length-value**

Specify a value ranging from 0 through 100 MB in bytes. The default is 4MB.

## Backout threshold (BKTTHLD)

Specifies the backout threshold.

Applications running inside of WebSphere Application Server and those that use the IBM MQ Application Server Facilities will use this attribute to determine if a message should be backed out. For all other applications, apart from allowing this attribute to be queried, the queue manager takes no action based on the value of the attribute.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **threshold-value**

Specify a value ranging from 0 through 999999999.

## Backout requeue name (BKTQNAME)

Specifies the backout-queue name.

Applications running inside of WebSphere Application Server and those that use the IBM MQ Application Server Facilities will use this attribute to determine where messages that have been backed out should go. For all other applications, apart from allowing this attribute to be queried, the queue manager takes no action based on the value of the attribute.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

No backout queue is specified.

### **backout-queue-name**

Specify the backout queue name.

## Initiation queue (INITQNAME)

Specifies the name of the initiation queue.

**Note:** The initiation queue must be on the same instance of a message queue manager.

The possible values are:

### **\*SAME**

The attribute is unchanged.

**\*NONE**

No initiation queue is specified.

**initiation-queue-name**

Specify the initiation queue name.

**Usage (USAGE)**

Specifies whether the queue is for normal usage, or for transmitting messages to a remote message queue manager.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NORMAL**

Normal usage (the queue is not a transmission queue)

**\*TMQ**

The queue is a transmission queue that is used to hold messages destined for a remote message queue manager. If the queue is intended for use in situations where a transmission queue name is not explicitly specified, the queue name must be the same as the name of the remote message queue manager. For further information, see IBM MQ Intercommunication.

**Definition type (DFNTYPE)**

Specifies the type of dynamic queue definition that is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor.

**Note:** This parameter only applies to a model queue definition.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*TEMPDYN**

A temporary dynamic queue is created. This value should not be specified with a DEFMSGPST value of \*YES.

**\*PERMDYN**

A permanent dynamic queue is created.

**Target object (TGTQNAME)**

Specifies the name of the object for which this queue is an alias.

The object can be a local or remote queue, a topic or a message queue manager.

**Note:** The target object does not need to exist at this time but it must exist when a process attempts to open the alias queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**target-object-name**

Specify the name of the target object.

**Remote queue (RMTQNAME)**

Specifies the name of the remote queue. That is, the local name of the remote queue as defined on the queue manager specified by RMTMQMNAME.

If this definition is used for a queue manager alias definition, RMTQNAME must be blank when the open occurs.

If this definition is used for a reply-to alias, this name is the name of the queue that is to be the reply-to queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No remote-queue name is specified (that is, the name is blank). This can be used if the definition is a queue manager alias definition.

**remote-queue-name**

Specify the name of the queue at the remote queue manager.

**Note:** The name is not checked to ensure that it contains only those characters normally allowed for queue names.

## Remote Message Queue Manager (RMTMQMNAME)

Specifies the name of the remote queue manager on which the queue RMTQNAME is defined.

If an application opens the local definition of a remote queue, RMTMQMNAME must not be the name of the connected queue manager. If TMQNAME is blank there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a queue manager alias, RMTMQMNAME is the name of the queue manager, which can be the name of the connected queue manager. Otherwise, if TMQNAME is blank, when the queue is opened there must be a local queue of this name, with USAGE(\*TMQ) specified, which is to be used as the transmission queue.

If this definition is used for a reply-to alias, this name is the name of the queue manager that is to be the reply-to queue manager.

The possible values are:

**\*SAME**

The attribute is unchanged.

**remote-queue-manager-name**

Specify the name of the remote queue manager.

**Note:** Ensure this name contains only those characters normally allowed for queue manager names.

## Transmission queue (TMQNAME)

Specifies the local name of the transmission queue to be used for messages destined for the remote queue, for either a remote queue or for a queue manager alias definition.

If TMQNAME is blank, a queue with the same name as RMTMQMNAME is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue manager alias and RMTMQMNAME is the name of the connected queue manager.

It is also ignored if the definition is used as a reply-to queue alias definition.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No specific transmission queue name is defined for this remote queue. The value of this attribute is set to all blanks.

**transmission-queue-name**

Specify the transmission queue name.

**Queue depth high threshold (HIGHTHLD)**

Specifies the threshold against which the queue depth is compared to generate a queue depth high event.

The possible values are:

**\*SAME**

The attribute is unchanged.

**threshold-value**

Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

**Queue depth low threshold (LOWTHLD)**

Specifies the threshold against which the queue depth is compared to generate a queue depth low event.

The possible values are:

**\*SAME**

The attribute is unchanged.

**threshold-value**

Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

**Queue full events enabled (FULLEVT)**

Specifies whether queue full events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Queue full events are not generated.

**\*YES**

Queue full events are generated.

**Queue high events enabled (HIGHEVT)**

Specifies whether queue depth high events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Queue depth high events are not generated.

**\*YES**

Queue depth high events are generated.

**Queue low events enabled (LOWEVT)**

Specifies whether queue depth low events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Queue depth low events are not generated.

**\*YES**

Queue depth low events are generated.

**Service interval (SRVITV)**

Specifies the service interval. This interval is used for comparison to generate service interval high and service interval OK events.

The possible values are:

**\*SAME**

The attribute is unchanged.

**interval-value**

Specify a value ranging from 0 through 999999999. The value is in units of milliseconds.

**Service interval events (SRVEVT)**

Specifies whether service interval high or service interval OK events are generated.

A service interval high event is generated when a check indicates that no messages have been retrieved from the queue for the time indicated by the SRVITV parameter as a minimum.

A service interval OK event is generated when a check indicates that messages have been retrieved from the queue within the time indicated by the SRVITV parameter.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*HIGH**

Service interval high events are generated.

**\*OK**

Service interval OK events are generated.

**\*NONE**

No service interval events are generated.

**Distribution list support (DISTLIST)**

Specifies whether the queue supports distribution lists.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

The queue will not support distribution lists.

**\*YES**

The queue will support distribution lists.

**Cluster Name (CLUSTER)**

The name of the cluster to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are already open.

This parameter cannot be set for dynamic, transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx or SYSTEM.COMMAND.xx queues.

The possible values are:



**\*SAME**

The attribute is unchanged.

**cluster-name**

Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

**Cluster Name List (CLUSNL)**

The name of the namelist which specifies a list of clusters to which the queue belongs. Changes to this parameter do not affect instances of the queue that are already open.

This parameter cannot be set for dynamic, transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx or SYSTEM.COMMAND.xx queues.

The possible values are:

**\*SAME**

The attribute is unchanged.

**namelist-name**

Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

**Default Binding (DEFBIND)**

Specifies the binding to be used when the application specifies MQOO\_BIND\_AS\_Q\_DEF on the MQOPEN call and the queue is a cluster queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*OPEN**

The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

**\*NOTFIXED**

The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using MQPUT and to change that selection subsequently if necessary.

The MQPUT1 call always behaves as if NOTFIXED had been specified.

**\*GROUP**

When the queue is opened, the queue handle is bound to a specific instance of the cluster queue for as long as there are messages in a message group. All messages in a message group are allocated to the same destination instance.

**Cluster Workload Rank (CLWLRANK)**

Specifies the cluster workload rank of the queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-rank**

Specify a value ranging from 0 through 9.

**Cluster Workload Priority (CLWLPRTY)**

Specifies the cluster workload priority of the queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-priority**

Specify a value ranging from 0 through 9.

**Cluster workload queue use (CLWLUSEQ)**

Specifies the behavior of an MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel then this attribute does not apply.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

The value is inherited from the Queue Manager CLWLUSEQ attribute.

**\*LOCAL**

The local queue will be the sole target of the MQPUT.

**\*ANY**

The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

**Queue Monitoring (MONQ)**

Controls the collection of Online Monitoring Data.

Online Monitoring Data is not collected when the queue manager attribute MONQ is set to \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

The collection of online monitoring data is inherited from the setting of the queue manager attribute MONQ.

**\*OFF**

Online monitoring data collection for this queue is disabled.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

**Queue Statistics (STATQ)**

Controls the collection of statistics data.

Online monitoring data is not collected when the queue manager attribute STATQ is set to \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

Statistics data collection is based upon the setting of the queue manager attribute STATQ.

**\*OFF**

Statistics data collection for this queue is disabled.

**\*ON**

Statistics data collection is enabled for this queue.

**Queue Accounting (ACCTQ)**

Controls the collection of accounting data.

Accounting data is not collected when the queue manager attribute ACCTQ is set to \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

Accounting data collection is based upon the setting of the queue manager attribute ACCTQ.

**\*OFF**

Accounting data collection for this queue is disabled.

**\*ON**

Accounting data collection is enabled for this queue.

**Non Persistent Message Class (NPMCLASS)**

Specifies the level of reliability for non-persistent messages put to this queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NORMAL**

Non-persistent messages put to this queue are only lost following a failure, or a queue manager shutdown. Non-persistent message put to this queue will be discarded in the event of a queue manager restart.

**\*HIGH**

Non-persistent messages put to this queue are not discarded in the event of a queue manager restart. Non-persistent messages put to this queue may still be lost in the event of a failure.

**Message Read Ahead (MSGREADAHD)**

Specifies whether non persistent messages are sent to the client ahead of an application requesting them.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*DISABLED**

Read ahead is disabled for this queue. Messages are not sent to the client ahead of an application requesting them regardless of whether read ahead is requested by the client application.

**\*NO**

Non-persistent messages are not sent to the client ahead of an application requesting them. A maximum of one non-persistent message can be lost if the client ends abnormally.

**\*YES**

Non-persistent messages are sent to the client ahead of an application requesting them. Non-persistent messages can be lost if the client ends abnormally or if the client application does not consume all the messages it is sent.

## Default Put Response (DFTPUTRESP)

The default put response type (DFTPUTRESP) attribute specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO\_RESPONSE\_AS\_Q\_DEF option.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*SYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application. This is the default value supplied with IBM MQ, but your installation might have changed it.

### **\*ASYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application; but an improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

## Property Control (PROPCTL)

Specifies what happens to properties of messages that are retrieved from queues using the MQGET call when the MQGMO\_PROPERTIES\_AS\_Q\_DEF option is specified.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*COMPAT**

If the message contains a property with a prefix of mcd . , jms . , usr . or mqext . then all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

### **\*NONE**

All properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

### **\*ALL**

All properties of the message, except those contained in the message descriptor (or extension), are contained in one or more MQRFH2 headers in the message data.

### **\*FORCE**

Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

### **\*V6COMPAT**

When set, \*V6COMPAT must be set both on one of the queue definitions resolved by MQPUT and one of the queue definitions resolved by MQGET. It must also be set on any other intervening transmission queues. It causes an MQRFH2 header to be passed unchanged from the sending application to the receiving application. It overrides other settings of **PROPCTL** found in a queue name resolution chain. If the property is set on a cluster queue, the setting is not cached locally on other queue managers. You must set \*V6COMPAT on an alias queue that resolves to the cluster queue. Define the alias queue on the same queue manager that the putting application is connected to.

## Target Type (TARGTYPE)

Specifies the type of object to which the alias resolves.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QUEUE**

Queue object.

**\*TOPIC**

Topic object.

**Custom attribute (CUSTOM)**

This attribute is reserved for the configuration of new features before separate attributes have been introduced. This description will be updated when features using this attribute are introduced. At the moment there are no meaningful values for *CUSTOM*, so leave it empty.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**custom**

Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs must have the form NAME (VALUE) and be specified in uppercase. Single quotes must be escaped with another single quote.

**CLCHNAME**

This parameter is supported only on transmission queues.

**\*SAME**

The attribute is unchanged.

**\*NONE**

The attribute is removed.

**cluster-sender channel name**

`ClusterChannelName` is the generic name of the cluster-sender channels that use this queue as a transmission queue. The attribute specifies which cluster-sender channels send messages to a cluster-receiver channel from this cluster transmission queue.

By specifying asterisks, "\*", in **ClusterChannelName**, you can associate a transmission queue with a set of cluster-sender channels. The asterisks can be at the beginning, end, or any number of places in the middle of the channel name string. **ClusterChannelName** is limited to a length of 20 characters: `MQ_CHANNEL_NAME_LENGTH`.

**IMGRCOVQ**

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*YES**

These queue objects are recoverable.

**\*NO**

The ["RCDMQMIMG \(Record MQ Object Image\)"](#) on page 1857 and ["RCRMQMOBJ \(Re-create MQ Object\)"](#) on page 1859 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

## \*QMGR

If you specify \*QMGR, and the **IMGRCOVQ** attribute for the queue manager specifies \*YES, these queue objects are recoverable.

If you specify \*QMGR and the **IMGRCOVQ** attribute for the queue manager specifies \*NO, the “[RCDMQMIMG \(Record MQ Object Image\)](#)” on page 1857 and “[RCRMQM OBJ \(Re-create MQ Object\)](#)” on page 1859 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

## IBM i CHGMQMSUB (Change MQ Subscription)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Change MQ Subscription (CHGMQMSUB) command changes the specified attributes of an existing MQ subscription.

### Parameters

Table 227. Command parameters

Keyword	Description	Choices	Notes
<u>SUBID</u>	Subscription identifier	Character value, *SAME	Optional, Key, Positional 2
<u>SUBNAME</u>	Subscription name	Character value, *SAME	Optional, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Key, Positional 3
<u>TOPICSTR</u>	Topic string	Character value, *NONE, *SAME	Optional, Positional 4
<u>TOPICOBJ</u>	Topic object	Character value, *NONE, *SAME	Optional, Positional 5
<u>DEST</u>	Destination	Character value, *SAME	Optional, Positional 6
<u>DESTMQM</u>	Destination Queue Manager	Character value, *NONE, *SAME	Optional, Positional 7
<u>DESTCRLID</u>	Destination Correlation Id	Character value, *NONE, *SAME	Optional, Positional 8
<u>PUBACCT</u>	Publish Accounting Token	Character value, *NONE, *SAME	Optional, Positional 9
<u>PUBAPPID</u>	Publish Application Id	Character value, *NONE, *SAME	Optional, Positional 10
<u>SUBUSER</u>	Subscription User Id	Character value, *SAME	Optional, Positional 11
<u>USERDATA</u>	Subscription User Data	Character value, *NONE, *SAME	Optional, Positional 12
<u>SELECTOR</u>	Selector String	Character value, *NONE, *SAME	Optional, Positional 13
<u>PSPROP</u>	PubSub Property	*SAME, *NONE, *COMPAT, *RFH2, *MSGPROP	Optional, Positional 14

Table 227. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>DESTCLASS</u>	Destination Class	<b>*SAME</b> , *MANAGED, *PROVIDED	Optional, Positional 15
<u>VARUSER</u>	Variable User	<b>*SAME</b> , *ANY, *FIXED	Optional, Positional 16
<u>REQONLY</u>	Request Publications	<b>*SAME</b> , *YES, *NO	Optional, Positional 17
<u>PUBPTY</u>	Publish Priority	0-9, <b>*SAME</b> , *AS PUB, *AS QDEF	Optional, Positional 18
<u>WSHEMA</u>	Wildcard Schema	<b>*SAME</b> , *CHAR, *TOPIC	Optional, Positional 19
<u>EXPIRY</u>	Expiry Time	0-999999999, <b>*SAME</b> , *UNLIMITED	Optional, Positional 20

### Subscription identifier (SUBID)

The subscription identifier of the subscription to be changed.

The possible values are:

#### subscription-identifier

Specify the 48 character hexadecimal string representing the 24 byte subscription identifier.

### Subscription name (SUBNAME)

The name of the subscription to be changed.

The possible values are:

#### subscription-name

Specify a maximum of 256 bytes for the subscription name.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

### Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

#### \*DFT

Use the default Queue Manager.

#### queue-manager-name

The name of a Queue Manager.

### Topic string (TOPICSTR)

Specifies the topic string associated with this subscription.

The possible values are:

#### topic-string

Specify a maximum of 256 bytes for the topic string.

**Note:** Topic strings of greater than 256 bytes can be specified using MQSC.

### Topic object (TOPICOBJ)

Specifies the topic object associated with this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**topic-object**

Specify the name of the topic object.

### **Destination (DEST)**

Specifies the destination queue for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**destination-queue**

Specify the name of the destination queue.

### **Destination Queue Manager (DESTMQM)**

Specifies the destination queue manager for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No destination queue manager is specified.

**destination-queue**

Specify the name of the destination queue manager.

### **Destination Correlation Id (DESTCRLID)**

Specifies the correlation identifier for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Messages are placed on the destination with a correlation identifier of MQCI\_NONE.

**correlation-identifier**

Specify the 48 character hexadecimal string representing the 24 byte correlation identifier.

### **Publish Accounting Token (PUBACCT)**

Specifies the accounting token for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Messages are placed on the destination with an accounting token of MQACT\_NONE.

**publish-accounting-token**

Specify the 64 character hexadecimal string representing the 32 byte publish accounting token.

### **Publish Application Id (PUBAPPID)**

Specifies the publish application identity for messages published to this subscription.



The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No publish application identifier is specified.

**publish-application-identifier**

Specify the publish application identifier.

### **Subscription User Id (SUBUSER)**

Specifies the user profile that owns this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**user-profile**

Specify the user profile.

### **Subscription User Data (USERDATA)**

Specifies the user data associated with the subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No user data is specified.

**user-data**

Specify a maximum of 256 bytes for user data.

**Note:** User data of greater than 256 bytes can be specified using MQSC.

### **Selector String (SELECTOR)**

Specifies the SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No selection string is specified.

**selection-string**

Specify a maximum of 256 bytes for selection string.

**Note:** Selection strings of greater than 256 bytes can be specified using MQSC.

### **PubSub Property (PSPROP)**

Specifies the manner in which publish / subscribe related message properties are added to messages sent to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Publish / subscribe properties are not added to the message.

**\*COMPAT**

Publish / subscribe properties are added to the message to maintain compatibility with IBM MQ V6.0  
Publish / Subscribe.

**\*RFH2**

Publish / subscribe properties are added to the message within an RFH 2 header.

**\*MSGPROP**

Publish / subscribe properties are added as message properties.

**Destination Class (DESTCLASS)**

Specifies whether this is a managed subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*MANAGED**

The destination is managed.

**\*PROVIDED**

The destination is a queue.

**Variable User (VARUSER)**

Specifies whether user profiles other than the creator of the subscription can connect to it (subject to topic and destination authority checks).

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ANY**

Any user profiles can connect to the subscription.

**\*FIXED**

Only the user profile that created the subscription can connect to it.

**Request Publications (REQONLY)**

Specifies whether the subscriber will poll for updates via MQSUBRQ API, or whether all publications are delivered to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*YES**

Publications are only delivered to this subscription in response to an MQSUBRQ API.

**\*NO**

All publications on the topic are delivered to this subscription.

**Publish Priority (PUBPTY)**

Specifies the priority of the message sent to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPUB**

The priority of the message sent to this subscription is taken from that supplied in the published message.

**\*ASQDEF**

The priority of the message sent to this subscription is taken from the default priority of the queue defined as the destination.

**priority-value**

Specify a priority ranging from 0 through 9.

**Wildcard Schema (WSHEMA)**

Specifies the schema to be used when interpreting wildcard characters in the topic string.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*TOPIC**

Wildcard characters represent portions of the topic hierarchy.

**\*CHAR**

Wildcard characters represent portions of strings.

**Expiry Time (EXPIRY)**

Specifies the expiry time of the subscription. After a subscription's expiry time has elapsed, it becomes eligible to be discarded by the queue manager and will receive no further publications.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*UNLIMITED**

The subscription does not expire.

**expiry-time**

Specify an expiry time in tenths of a second ranging from 0 through 999999999.

 **CHGMQMSVC (Change MQ Service)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Change MQ Service (CHGMQMSVC) command changes the specified attributes of an existing MQ service definition.

**Parameters**

<i>Table 228. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>SVCNAME</u>	Service name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Key, Positional 2

Table 228. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>TEXT</u>	Text 'description'	Character value, *BLANK, *SAME	Optional, Positional 3
<u>STRCMD</u>	Start program	Single values: *SAME, *NONE Other values: Qualified object name	Optional, Positional 4
	Qualifier 1: Start program	Name	
	Qualifier 2: Library	Name	
<u>STRARG</u>	Start program arguments	Character value, *BLANK, *SAME	Optional, Positional 5
<u>ENDCMD</u>	End program	Single values: *SAME, *NONE Other values: Qualified object name	Optional, Positional 6
	Qualifier 1: End program	Name	
	Qualifier 2: Library	Name	
<u>ENDARG</u>	End program arguments	Character value, *BLANK, *SAME	Optional, Positional 7
<u>STDOUT</u>	Standard output	Character value, *BLANK, *SAME	Optional, Positional 8
<u>STDERR</u>	Standard error	Character value, *BLANK, *SAME	Optional, Positional 9
<u>TYPE</u>	Service type	*SAME, *CMD, *SVR	Optional, Positional 10
<u>CONTROL</u>	Service control	*SAME, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 11

### Service name (SVCNAME)

The name of the service definition to be changed.

The possible values are:

#### service-name

Specify the name of the service definition. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.

### Text 'description' (TEXT)

Specifies text that briefly describes the service definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

### **Start program (STRCMD)**

The name of the program to run.

The possible values are:

**\*SAME**

The attribute is unchanged.

**start-command**

The name of the start command executable.

### **Start program arguments (STRARG)**

The arguments passed to the program at startup.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

No arguments are passed to the start command.

**start-command-arguments**

The arguments passed to the start command.

### **End program (ENDCMD)**

The name of the executable to run when the service is requested to stop.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

No end command is executed.

**end-command**

The name of the end command executable.

### **End program arguments (ENDARG)**

The arguments passed to the end program when the service is requested to stop.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

No arguments are passed to the end command.

**end-command-arguments**

The arguments passed to the end command.

**Standard output (STDOUT)**

The path to a file to which the standard output of the service program is redirected.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The standard output is discarded.

**stdout-path**

The standard output path.

**Standard error (STDERR)**

The path to a file to which the standard error of the service program is redirected.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The standard error is discarded.

**stderr-path**

The standard error path.

**Service type (TYPE)**

Mode in which to run service.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*CMD**

When started the command is executed but no status is collected or displayed.

**\*SVR**

The status of the executable started will be monitored and displayed.

**Service control (CONTROL)**

Whether the service should be started automatically at queue manager start.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*MANUAL**

The service is automatically started or stopped.

**\*QMGR**

The service is started and stopped as the queue manager is started and stopped.

**\*STARTONLY**

The service is started as the queue manager is started, but will not be requested to stop when the queue manager is stopped.

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Change MQ Topic (CHGMQMTOP) command changes the specified attributes of an existing MQ topic object.

**Parameters**

<i>Table 229. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>TOPNAME</u>	Topic name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Key, Positional 2
<u>TEXT</u>	Text 'description'	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 3
<u>TOPICSTR</u>	Topic string	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 4
<u>DURSUB</u>	Durable subscriptions	<b>*SAME</b> , *ASPARENT, *YES, *NO	Optional, Positional 5
<u>MGDDURMDL</u>	Durable model queue	Character value, *NONE, <b>*SAME</b>	Optional, Positional 6
<u>MGDNDURMDL</u>	Non-durable model queue	Character value, *NONE, <b>*SAME</b>	Optional, Positional 7
<u>PUBENBL</u>	Publish	<b>*SAME</b> , *ASPARENT, *YES, *NO	Optional, Positional 8
<u>SUBENBL</u>	Subscribe	<b>*SAME</b> , *ASPARENT, *YES, *NO	Optional, Positional 9
<u>DFTPTY</u>	Default message priority	0-9, <b>*SAME</b> , *ASPARENT	Optional, Positional 10
<u>DFTMSGPST</u>	Default message persistence	<b>*SAME</b> , *ASPARENT, *YES, *NO	Optional, Positional 11
<u>DFTPURRESP</u>	Default Put Response	<b>*SAME</b> , *ASPARENT, *SYNC, *ASYN	Optional, Positional 12
<u>WILDCARD</u>	Wildcard behavior	<b>*SAME</b> , *PASSTHRU, *BLOCK	Optional, Positional 13
<u>PMSGDLV</u>	Persistent message delivery	<b>*SAME</b> , *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 14
<u>NPMSGDLV</u>	Non-persistent message deliver	<b>*SAME</b> , *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 15
<u>CUSTOM</u>	Custom attribute	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 16

## Topic name (TOPNAME)

The name of the topic object to be changed.

The possible values are:

### **topic-name**

Specify the name of the topic object. The maximum length of the string is 48 bytes.

## Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

### **\*DFT**

Use the default Queue Manager.

### **queue-manager-name**

The name of a Queue Manager.

## Text 'description' (TEXT)

Specifies text that briefly describes the topic object.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*BLANK**

The text is set to a blank string.

### **description**

Specify no more than 64 characters enclosed in apostrophes.

## Topic string (TOPICSTR)

Specifies the topic string represented by this topic object definition.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **topic-string**

Specify a maximum of 256 bytes for the topic string.

**Note:** Topic strings of greater than 256 bytes can be specified using MQSC.

## Durable subscriptions (DURSUB)

Specifies whether applications are permitted to make durable subscriptions on this topic.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*ASPARENT**

Whether durable subscriptions can be made on this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **\*YES**

Durable subscriptions can be made on this topic.



**\*NO**

Durable subscriptions cannot be made on this topic.

**Durable model queue (MGDDURMDL)**

Specifies the name of the model queue to be used for durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

**\*SAME**

The attribute is unchanged.

**durable-model-queue**

Specify the name of the model queue.

**Non-durable model queue (MGDNDURMDL)**

Specifies the name of the model queue to be used for non-durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

**\*SAME**

The attribute is unchanged.

**non-durable-model-queue**

Specify the name of the model queue.

**Publish (PUBENBL)**

Specifies whether messages can be published to the topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

Whether messages can be published to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*YES**

Messages can be published to the topic.

**\*NO**

Messages cannot be published to the topic.

**Subscribe (SUBENBL)**

Specifies whether applications are to be permitted to subscribe to this topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

Whether applications can subscribe to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*YES**

Subscriptions can be made to this topic.

**\*NO**

Applications cannot subscribe to this topic.

## Default message priority (DFTPTY)

Specifies the default priority of messages published to the topic.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*ASPARENT**

The default priority is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **priority-value**

Specify a value ranging from 0 through 9.

## Default message persistence (DFTMSGPST)

Specifies the message persistence to be used when applications specify the MQPER\_PERSISTENCE\_AS\_TOPIC\_DEF option.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*ASPARENT**

The default persistence is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **\*YES**

Messages on this queue survive a restart of the queue manager.

### **\*NO**

Messages on this queue are lost across a restart of the queue manager.

## Default Put Response (DFTPURRESP)

Specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO\_RESPONSE\_AS\_Q\_DEF option.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*ASPARENT**

The default response type is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **\*SYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application.

### **\*ASYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application. An improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

## Wildcard behavior (WILDCARD)

Specifies the behavior of wildcard subscriptions with respect to this topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*PASSTHRU**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will receive publications made to this topic and to topic strings more specific than this topic.

**\*BLOCK**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will not receive publications made to this topic or to topic strings more specific than this topic.

**Persistent message delivery (PMSGDLV)**

Specifies the delivery mechanism for persistent messages published to this topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*ALL**

Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**\*ALLDUR**

Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

**\*ALLAVAIL**

Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**Non-persistent message delivery (NPMSGDLV)**

Specifies the delivery mechanism for non-persistent messages published to this topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*ALL**

Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**\*ALLDUR**

Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

**\*ALLAVAIL**

Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

## Custom attribute (CUSTOM)

This attribute is reserved for the configuration of new features before separate attributes have been introduced. This description will be updated when features using this attribute are introduced. At the moment there are no meaningful values for *CUSTOM*, so leave it empty.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*BLANK**

The text is set to a blank string.

### **custom**

Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs must have the form NAME (VALUE) and be specified in uppercase. Single quotes must be escaped with another single quote.

## **CLRMQMBRK (Clear MQ Pub/Sub Broker)**

### **Where allowed to run**

All environments (\*ALL)

### **Threadsafe**

Yes

The Clear IBM MQ broker (CLRMQMBRK) command does not perform any function and is only provided for compatibility with previous releases of IBM MQ.

## Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1
<u>BRKPARENT</u>	Break Parent link	<b>*NO</b> , *YES	Optional, Positional 2
<u>CHILDMQM</u>	Child Message Queue Manager	Character value	Optional, Positional 3

### **Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

#### **queue-manager-name**

Specify the name of the queue manager.

### **Break Parent link (BRKPARENT)**

Specifies how the broker is ended.

The possible values are:

#### **\*YES**

Specifies that the link is to be broken with the parent broker. If you specify this parameter you must not specify a value for CHILDMQM.

#### **\*NO**

Specifies that the link is to be broken with a child broker. Use the CHILDMQM parameter to specify the name of the queue manager that hosts the child broker.

## Child Message Queue Manager (CHILDMQM)

Specifies the name of the queue manager that hosts the child broker that the link is to be broken with.

## CLRMQM (Clear MQ Queue)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Clear MQ Queue (CLRMQM) command deletes all of the messages from a local queue.

The command fails if the queue contains uncommitted messages, or if an application has the queue open.

## Parameters

Keyword	Description	Choices	Notes
<u>QNAME</u>	Queue name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , *DFT	Optional, Positional 2

### Queue name (QNAME)

The name of the queue to be cleared.

The possible values are:

#### queue-name

Specify the name of the queue.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

Specify the name of the queue manager.

## CLRMQMTOP (Clear MQ Topic String)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Clear MQ Topic String (CLRMQMTOP) command clears the specified topic string.

## Parameters

Keyword	Description	Choices	Notes
<u>TOPICSTR</u>	Topic string	Character value	Required, Positional 1

Table 232. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>CLRTYPE</u>	Clear type	<b>*RETAINED</b>	Optional, Positional 3

### Topic string (TOPICSTR)

The topic string to be cleared.

The possible values are:

#### topic-string

Specify a maximum of 256 bytes for the topic string.

**Note:** Topic strings of greater than 256 bytes can be specified using MQSC.

### Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

#### **\*DFT**

Use the default Queue Manager.

#### queue-manager-name

The name of a Queue Manager.

### Clear type (CLRTYPE)

The type of clear topic string to be performed.

The value must be:

#### **\*RETAINED**

Remove the retained publication from the specified topic string.

## CPYMQMAUTI (Copy MQ AuthInfo object)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Copy MQ AuthInfo object (CPYMQMAUTI) command creates an authentication information object of the same type and, for attributes not specified in the command, with the same attribute values as an existing object.

### Parameters

Table 233. Command parameters

Keyword	Description	Choices	Notes
<u>FROMAI</u>	From AuthInfo name	Character value	Required, Key, Positional 1
<u>TOAI</u>	To AuthInfo name	Character value	Required, Key, Positional 2

Table 233. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 3
<u>AUTHTYPE</u>	AuthInfo type	*CRLLDAP, *OCSP, *IDPWOS, *IDPWLDAP	Optional, Positional 4
<u>CONNAME</u>	Connection name	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 5
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 6
<u>TEXT</u>	Text 'description'	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 7
<u>USERNAME</u>	User name	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 8
<u>PASSWORD</u>	User password	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 9
<u>OCSPURL</u>	OCSP Responder URL	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 10
<u>CHKCLNT</u>	Authentication checks required	*ASQMGR, *REQUIRED, *REQADM	Optional, Positional 11
<u>CHKLOCL</u>	Authentication checks required	*NONE, *OPTIONAL, *REQUIRED, *REQADM	Optional, Positional 12
<u>FAILDELAY</u>	Failure delay	<i>Integer value</i>	Optional, Positional 13
<u>BASEDNU</u>	Base user DN	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 14
<u>ADOPTCTX</u>	Context adoption	<i>Integer value</i>	Optional, Positional 15
<u>CLASSUSR</u>	LDAP object class	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 16
<u>SHORTUSR</u>	Short user name	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 17
<u>USRFIELD</u>	User field	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 18
<u>SECCOMM</u>	LDAP communications	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 19
<u>AUTHORMD</u>	Authorization method	<i>Character value</i> , <b>*OS</b> , *SEARCHGRP, *SEARCHUSR, *SRCHGRPSN	Optional, Positional 20
<u>BASEDNG</u>	Base DN for groups	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 21
<u>CLASSGRP</u>	Object class for group	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 22
<u>FINDGRP</u>	Attribute to find group membership	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 23
<u>GRPFIELD</u>	Simple name for group	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 24
<u>NESTGRP</u>	Group nesting	<b>*NO</b> *YES	Optional, Positional 25
<u>AUTHENMD</u>	Authentication method	<b>*OS</b> Cannot be changed	Optional, Positional 26

### From AuthInfo name (FROMAI)

The name of an existing authentication information object to provide values for the attributes not specified in this command.

The possible values are:

**authentication-information-name**

Specify the name of the authentication information object. The maximum string length is 48 characters.

**To AuthInfo name (TOAI)**

The name of the new authentication information object to create.

If an authentication information object with this name already exists, REPLACE(\*YES) must be specified.

The possible values are:

**authentication-information-name**

Specify the name of the authentication information object. The maximum string length is 48 characters.

**Message Queue Manager name (MQMNAME)**

The name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of an existing message queue manager. The maximum string length is 48 characters.

**Adopt context (ADOPTCTX)**

Whether to use the presented credentials as the context for this application. This means that they are used for authorization checks, shown on administrative displays, and appear in messages.

**YES**

The user ID presented in the MQCSP structure, which has been successfully validated by password, is adopted as the context to use for this application. Therefore, this user ID will be the credentials checked for authorization to use IBM MQ resources.

If the user ID presented is an LDAP user ID, and authorization checks are done using operating system user IDs, the *SHORTUSR* associated with the user entry in LDAP will be adopted as the credentials for authorization checks to be done against.

**NO**

Authentication will be performed on the user ID and password presented in the MQCSP structure, but then the credentials will not be adopted for further use. Authorization will be performed using the user ID the application is running under.

This attribute is only valid for an AUTHTYPE of \*IDPWOS and \*IDPWLDAP.

**Authentication method (AUTHENMD)**

The authentication method used for this application.

**\*OS**

Use operating system groups to determine permissions associated with a user.

You can use only **\*OS** to set the authentication method.

This attribute is valid only for an AUTHTYPE of \*IDPWOS.

**Authorization method (AUTHORMD)**

The authorization method used for this application.



**\*OS**

Use operating system groups to determine permissions associated with a user.

This is how IBM MQ has previously worked, and is the default value.

**\*SEARCHGRP**

A group entry in the LDAP repository contains an attribute listing the Distinguished Name of all the users belonging to that group. Membership is indicated by the attribute defined in [FINDGRP](#). This value is typically *member* or *uniqueMember*.

**\*SEARCHUSR**

A user entry in the LDAP repository contains an attribute listing the Distinguished Name of all the groups to which the specified user belongs. The attribute to query is defined by the [FINDGRP](#) value, typically *memberOf*.

**\*SRCHGRPSN**

A group entry in the LDAP repository contains an attribute listing the short user name of all the users belonging to that group. The attribute in the user record that contains the short user name is specified by [SHORTUSR](#).

Membership is indicated by the attribute defined in [FINDGRP](#). This value is typically *memberUid*.

**Note:** This authorization method should only be used if all user short names are distinct.

Many LDAP servers use an attribute of the group object to determine group membership and you should, therefore, set this value to *SEARCHGRP*.

Microsoft Active Directory typically stores group memberships as a user attribute. The IBM Tivoli Directory Server supports both methods.

In general, retrieving memberships through a user attribute will be faster than searching for groups that list the user as a member.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

**AuthInfo type (AUTHTYPE)**

The type of the authentication information object. There is no default value

The possible values are:

**\*CRLLDAP**

The type of the authentication information object is CRLLDAP.

**\*OCSP**

The type of the authentication information objects is OCSPURL.

**\*IDPWOS**

Connection authentication user ID and password checking is done using the operating system.

**\*IDPWLDAP**

Connection authentication user ID and password checking is done using an LDAP server.

**Base DN for groups (BASEDNG)**

In order to be able to find group names, this parameter must be set with the base DN to search for groups in the LDAP server.

This attribute is valid only for **AUTHTYPE** of *\*IDPWLDAP*.

**Base user DN (BASEDNU)**

In order to be able to find the short user name attribute (see [SHORTUSR](#)) this parameter must be set with the base DN to search for users within the LDAP server. This attribute is valid only for **AUTHTYPE** of *\*IDPWLDAP*.

## Check client (CHCKCLNT)

Whether connection authentication checks are required by all locally bound connections, or only checked when a user ID and password are provided in the MQCSP structure.

These attributes are valid only for an **AUTHTYPE** of *\*IDPWOS* or *\*IDPWLDAP*. The possible values are:

### **\*ASQMGR**

In order for the connection to be allowed in, it must meet the connection authentication requirements defined on the queue manager. If the CONNAUTH field provides an authentication information object, and the value of CHCKCLNT is *\*REQUIRED*, the connection will not be successful unless a valid user ID and password are supplied. If the CONNAUTH field does not provide an authentication information object, or the value of CHCKCLNT is not *\*REQUIRED*, then the user ID and password are not required.

### **\*REQUIRED**

Requires that all applications provide a valid user ID and password.

### **\*REQDADM**

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the *\*OPTIONAL* setting.

## Check local (CHCKLOCL)

Whether connection authentication checks are required by all locally bound connections, or only checked when a user ID and password are provided in the MQCSP structure.

These attributes are valid only for an **AUTHTYPE** of *\*IDPWOS* or *\*IDPWLDAP*. The possible values are:

### **\*NONE**

Switches off checking.

### **\*OPTIONAL**

Ensures that if a user ID and password are provided by an application, they are a valid pair, but that it is not mandatory to provide them. This option might be useful during migration, for example.

### **\*REQUIRED**

Requires that all applications provide a valid user ID and password.

### **\*REQDADM**

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the *\*OPTIONAL* setting.

## Class group (CLASSGRP)

The LDAP object class used for group records in the LDAP repository.

If the value is blank, **groupOfNames** is used.

Other commonly used values include *groupOfUniqueNames* or *group*.

This attribute is valid only for **AUTHTYPE** of *\*IDPWLDAP*.

## Class user (CLASSUSR)

The LDAP object class used for user records in the LDAP repository.

If blank, the value defaults to *inetOrgPerson*, which is generally the value needed.

For Microsoft Active Directory, the value you require required is often *user*.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

## Connection name (CONNAME)

The DNS name or IP address of the host on which the LDAP server is running, together with an optional port number. The default port number is 389. No default is provided for the DNS name or IP address.

This field is only valid for *\*CRLLDAP* or *\*IDPWLDAP* authentication information objects, when it is required.

When used with *IDPWLDAP* authentication information objects, this can be a comma separated list of connection names.

The possible values are:

**\*SAME**

The connection name remains unchanged from the original authentication information object.

**connection-name**

Specify the fully qualified DNS name or IP address of the host together with an optional port number. The maximum string length is 264 characters.

### **Failure delay (FAILDELAY)**

When a user ID and password are provided for connection authentication, and the authentication fails due to the user ID or password being incorrect, this is the delay, in seconds, before the failure is returned to the application.

This can aid in avoiding busy loops from an application that simply retries, continuously, after receiving a failure.

The value must be in the range 0 - 60 seconds. The default value is 1.

This attribute is only valid for an **AUTHTYPE** of *\*IDPWOS* and *\*IDPWLDAP*.

### **Group membership attribute (FINDGRP)**

Name of the attribute used within an LDAP entry to determine group membership.

When AUTHORMD = *\*SEARCHGRP*, this attribute is typically set to *member* or *uniqueMember*.

When AUTHORMD = *\*SEARCHUSR*, this attribute is typically set to *memberOf*.

When AUTHORMD = *\*SRCHGRPSN*, this attribute is typically set to *memberUid*.

When left blank, if:

- AUTHORMD = *\*SEARCHGRP*, this attribute defaults to *memberOf*
- AUTHORMD = *\*SEARCHUSR*, this attribute defaults to *member*
- AUTHORMD = *\*SRCHGRPSN*, this attribute defaults to *memberUid*

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

### **Simple name for group (GRPFIELD)**

If the value is blank, commands like setmqaut must use a qualified name for the group. The value can either be a full DN, or a single attribute.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

### **Group nesting (NESTGRP)**

The possible values are:

**\*NO**

Only the initially discovered groups are considered for authorization.

**\*YES**

The group list is searched recursively to enumerate all the groups to which a user belongs.

The group's Distinguished Name is used when searching the group list recursively, regardless of the authorization method selected in AUTHORMD.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

## Replace (REPLACE)

Specifies whether the new authentication information object should replace an existing authentication information object with the same name.

The possible values are:

### **\*NO**

This definition does not replace any existing authentication information object with the same name. The command fails if the named authentication information object already exists.

### **\*YES**

Replace an existing authentication information object. A new object is created if the named authentication information object does not exist.

## Secure comms (SECCOMM)

Whether connectivity to the LDAP server should be done securely using TLS

### **YES**

Connectivity to the LDAP server is made securely using TLS.

The certificate used is the default certificate for the queue manager, named in CERTLABL on the queue manager object, or if that is blank, the one described in [Digital certificate labels, understanding the requirements](#).

The certificate is located in the key repository specified in SSLKEYR on the queue manager object. A cipherspec will be negotiated that is supported by both IBM MQ and the LDAP server.

If the queue manager is configured to use SSLFIPS(YES) or SUITEB cipher specs, then this is taken account of in the connection to the LDAP server as well.

### **ANON**

Connectivity to the LDAP server is made securely using TLS just as for SECCOMM(YES) with one difference.

No certificate is sent to the LDAP server; the connection will be made anonymously. To use this setting, ensure that the key repository specified in SSLKEYR, on the queue manager object, does not contain a certificate marked as the default.

### **NO**

Connectivity to the LDAP server does not use TLS.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*

## Short user (SHORTUSR)

A field in the user record to be used as a short user name in IBM MQ.

This field must contain values of 12 characters or less. This short user name is used for the following purposes:

- If LDAP authentication is enabled, but LDAP authorization is not enabled, this is used as an operating system user ID for authorization checks. In this case, the attribute must represent an operating system user ID.
- If LDAP authentication and authorization are both enabled, this is used as the user ID carried with the message in order for the LDAP user name to be rediscovered when the user ID inside the message needs to be used.

For example, on another queue manager, or when writing report messages. In this case, the attribute does not need to represent an operating system user ID, but must be a unique string. An employee serial number is an example of a good attribute for this purpose.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP* and is mandatory.

## Text 'description' (TEXT)

A short text description of the authentication information object.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

### **\*SAME**

The text string is unchanged.

### **\*NONE**

The text is set to a blank string.

### **description**

The string length can be up to 64 characters enclosed in apostrophes.

## User field (USRFIELD)

If the user ID provided by an application for authentication does not contain a qualifier for the field in the LDAP user record, that is, it does not contain an '=' sign, this attribute identifies the field in the LDAP user record that is used to interpret the provided user ID.

This field can be blank. If this is the case, any unqualified user IDs use the [SHORTUSR](#) parameter to interpret the provided user ID.

The contents of this field will be concatenated with an '=' sign, together with the value provided by the application, to form the full user ID to be located in an LDAP user record. For example, the application provides a user of fred and this field has the value cn, then the LDAP repository will be searched for cn=fred.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

## User name (USERNAME)

The distinguished name of the user that is binding to the directory. The default user name is blank.

This field is only valid for *\*CRLLDAP* or *\*IDPWLDAP* authentication information objects.

The possible values are:

### **\*SAME**

The user name is unchanged.

### **\*NONE**

The user name is blank.

### **LDAP-user-name**

Specify the distinguished name of the LDAP user. The maximum string length is 1024 characters.

## User password (PASSWORD)

The password for the LDAP user.

This field is only valid for *\*CRLLDAP* or *\*IDPWLDAP* authentication information objects.

The possible values are:

### **\*SAME**

The password is unchanged.

### **\*NONE**

The password is blank.

### **LDAP-password**

The LDAP user password. The maximum string length is 32 characters.

## OCSP Responder URL (OCSPURL)

The URL of the OCSP Responder used to check for certificate revocation. This must be an HTTP URL containing the host name and port number of the OCSP Responder. If the OCSP Responder is using port 80, which is the default for HTTP, then the port number may be omitted.

This field is only valid for OCSP authentication information objects.

The possible values are:

### **\*SAME**

The OCSP Responder URL is unchanged.

### **OCSP-Responder-URL**

The OCSP Responder URL. The maximum string length is 256 characters.

## Examples

None

## Error messages

Unknown

## CPYMQMCHL (Copy MQ Channel)

### **Where allowed to run**

All environments (\*ALL)

### **Threadsafe**

Yes

The Copy MQ Channel (CPYMQMCHL) command creates a new MQ channel definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing channel definition.

## Parameters

Table 234. Command parameters

Keyword	Description	Choices	Notes
<u>FROMCHL</u>	From channel	Character value	Required, Key, Positional 1
<u>TOCHL</u>	To channel	Character value	Required, Key, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Key, Positional 3
<u>CHLTYPE</u>	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Optional, Key, Positional 4
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 5
<u>TRPTYPE</u>	Transport type	*LU62, *TCP, <b>*SAME</b>	Optional, Positional 6
<u>TEXT</u>	Text 'description'	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 7

Table 234. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>TGTMQNAME</u>	Target Queue Manager	Character value, *NONE, <b>*SAME</b>	Optional, Positional 8
<u>CONNNAME</u>	Connection name	Character value, *NONE, <b>*SAME</b>	Optional, Positional 9
<u>TPNAME</u>	Transaction Program Name	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 10
<u>MODENAME</u>	Mode Name	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 11
<u>TMQNAME</u>	Transmission queue	Character value, <b>*SAME</b>	Optional, Positional 12
<u>MCANAME</u>	Message channel agent	Single values: <b>*SAME</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 13
	Qualifier 1: Message channel agent	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	
<u>MCAUSRID</u>	Message channel agent user ID	Character value, *NONE, *PUBLIC, <b>*SAME</b>	Optional, Positional 14
<u>MCATYPE</u>	Message channel agent Type	*PROCESS, *THREAD, <b>*SAME</b>	Optional, Positional 15
<u>BATCHINT</u>	Batch Interval	0-999999999, <b>*SAME</b>	Optional, Positional 16
<u>BATCHSIZE</u>	Batch size	1-9999, <b>*SAME</b>	Optional, Positional 17
<u>DSCITV</u>	Disconnect interval	0-999999, <b>*SAME</b>	Optional, Positional 18
<u>SHORTTMR</u>	Short retry interval	0-999999999, <b>*SAME</b>	Optional, Positional 19
<u>SHORTRTY</u>	Short retry count	0-999999999, <b>*SAME</b>	Optional, Positional 20
<u>LONGTMR</u>	Long retry interval	0-999999999, <b>*SAME</b>	Optional, Positional 21
<u>LONGRTY</u>	Long retry count	0-999999999, <b>*SAME</b>	Optional, Positional 22
<u>SCYEXIT</u>	Security exit	Single values: <b>*SAME</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 23
	Qualifier 1: Security exit	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	
<u>CSCYEXIT</u>	Security exit	Character value, <b>*SAME</b> , *NONE	Optional, Positional 24
<u>SCYUSRDATA</u>	Security exit user data	Character value, <b>*SAME</b> , *NONE	Optional, Positional 25

Table 234. Command parameters (continued)

Keyword	Description	Choices	Notes
<a href="#">SNDEXIT</a>	Send exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 26
	Qualifier 1: Send exit	Name	
	Qualifier 2: Library	<i>Name</i> , <b>*CURLIB</b>	
<a href="#">CSNDEXIT</a>	Send exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Character value</i>	Optional, Positional 27
<a href="#">SNDUSRDATA</a>	Send exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 28
<a href="#">RCVEXIT</a>	Receive exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 29
	Qualifier 1: Receive exit	Name	
	Qualifier 2: Library	<i>Name</i> , <b>*CURLIB</b>	
<a href="#">CRCVEXIT</a>	Receive exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Character value</i>	Optional, Positional 30
<a href="#">RCVUSRDATA</a>	Receive exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 31
<a href="#">MSGEXIT</a>	Message exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 32
	Qualifier 1: Message exit	Name	
	Qualifier 2: Library	<i>Name</i> , <b>*CURLIB</b>	
<a href="#">MSGUSRDATA</a>	Message exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 33
<a href="#">MSGRTYEXIT</a>	Message retry exit	Single values: <b>*SAME</b> , <b>*NONE</b> Other values: <i>Qualified object name</i>	Optional, Positional 34
	Qualifier 1: Message retry exit	Name	
	Qualifier 2: Library	<i>Name</i> , <b>*CURLIB</b>	
<a href="#">MSGRTYDATA</a>	Message retry exit data	<i>Character value</i> , <b>*SAME</b> , <b>*NONE</b>	Optional, Positional 35



Table 234. Command parameters (continued)


Keyword	Description	Choices	Notes
<a href="#">MSGRTYNBR</a>	Number of message retries	0-999999999, <b>*SAME</b>	Optional, Positional 36
<a href="#">MSGRTYITV</a>	Message retry interval	0-999999999, <b>*SAME</b>	Optional, Positional 37
<a href="#">CVTMSG</a>	Convert message	*YES, *NO, <b>*SAME</b>	Optional, Positional 38
<a href="#">PUTAUT</a>	Put authority	*DFT, *CTX, <b>*SAME</b>	Optional, Positional 39
<a href="#">SEQNUMWRAP</a>	Sequence number wrap	100-999999999, <b>*SAME</b>	Optional, Positional 40
<a href="#">MAXMSGLEN</a>	Maximum message length	0-104857600, <b>*SAME</b>	Optional, Positional 41
<a href="#">HRTBTINTVL</a>	Heartbeat interval	0-999999999, <b>*SAME</b>	Optional, Positional 42
<a href="#">NPMSPEED</a>	Non Persistent Message Speed	*FAST, *NORMAL, <b>*SAME</b>	Optional, Positional 43
<a href="#">CLUSTER</a>	Cluster Name	Character value, *NONE, <b>*SAME</b>	Optional, Positional 44
<a href="#">CLUSNL</a>	Cluster Name List	Character value, *NONE, <b>*SAME</b>	Optional, Positional 45
<a href="#">NETPRTY</a>	Network Connection Priority	0-9, <b>*SAME</b>	Optional, Positional 46
<a href="#">SSLCIPH</a>	TLS CipherSpec	Supported CipherSpecs are listed here: <a href="#">CipherSpecs you can use with IBM MQ TLS support</a> .  <b>Deprecated</b> CipherSpecs that you can re-enable if necessary are listed here: <a href="#">Deprecated CipherSpecs</a> .	Optional, Positional 47
<a href="#">SSLCAUTH</a>	TLS Client Authentication	*REQUIRED, *OPTIONAL, <b>*SAME</b>	Optional, Positional 48
<a href="#">SSLPEER</a>	TLS Peer name	Character value, *NONE, <b>*SAME</b>	Optional, Positional 49
<a href="#">LOCLADDR</a>	Local communication address	Character value, *NONE, <b>*SAME</b>	Optional, Positional 50
<a href="#">BATCHHB</a>	Batch Heartbeat Interval	0-999999999, <b>*SAME</b>	Optional, Positional 51
<a href="#">USERID</a>	Task user identifier	Character value, *NONE, <b>*SAME</b>	Optional, Positional 52
<a href="#">PASSWORD</a>	Password	Character value, *NONE, <b>*SAME</b>	Optional, Positional 53
<a href="#">KAINT</a>	Keep Alive Interval	0-99999, <b>*SAME</b> , *AUTO	Optional, Positional 54
<a href="#">COMPHDR</a>	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, <b>*SAME</b>	Optional, Positional 55

Table 234. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>COMPMSG</u>	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST, *SAME	Optional, Positional 56
<u>MONCHL</u>	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, *SAME	Optional, Positional 57
<u>STATCHL</u>	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, *SAME	Optional, Positional 58
<u>CLWLRANK</u>	Cluster Workload Rank	0-9, *SAME	Optional, Positional 59
<u>CLWLPRTY</u>	Cluster Workload Priority	0-9, *SAME	Optional, Positional 60
<u>CLWLWGHT</u>	Cluster Channel Weight	1-99, *SAME	Optional, Positional 61
<u>SHARECNV</u>	Sharing Conversations	0-999999999, *SAME	Optional, Positional 62
<u>PROPCTL</u>	Property Control	*COMPAT, *NONE, *ALL, *SAME	Optional, Positional 63
<u>MAXINST</u>	Maximum Instances	0-999999999, *SAME	Optional, Positional 64
<u>MAXINSTC</u>	Maximum Instances Per Client	0-999999999, *SAME	Optional, Positional 65
<u>CLNTWGHT</u>	Client Channel Weight	0-99, *SAME	Optional, Positional 66
<u>AFFINITY</u>	Connection Affinity	*PREFERRED, *NONE, *SAME	Optional, Positional 67
<u>BATCHLIM</u>	Batch Data Limit	0-999999, *SAME	Optional, Positional 68
<u>DFTRECON</u>	Default client reconnection	*NO, *YES, *QMGR, *DISABLED, *SYSDFTCHL	Optional, Positional 69

### From channel (FROMCHL)

Specifies the name of the existing channel definition that contains values for the attributes that are not specified in this command.

The possible values are:

#### from-channel-name

Specify the name of the source MQ channel.

### To channel (TOCHL)

Specifies the name of the new channel definition. The name can contain a maximum of 20 characters. Channel names must be unique. If a channel definition with this name already exists, REPLACE(\*YES) must be specified.

The possible values are:

#### to-channel-name

Specify the name of MQ channel being created.

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

### **\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

### **message-queue-manager-name**

The name of a message queue manager.

## Channel type (CHLTYPE)

Specifies the type of the channel being copied.

The possible values are:

### **\*SDR**

Sender channel

### **\*SVR**

Server channel

### **\*RCVR**

Receiver channel

### **\*RQSTR**

Requester channel

### **\*SVRCN**

Server-connection channel

### **\*CLUSSDR**

Cluster-sender channel

### **\*CLUSRCVR**

Cluster-receiver channel

### **\*CLTCN**

Client-connection channel

## Replace (REPLACE)

Specifies whether the new channel definition replaces an existing channel definition with the same name.

The possible values are:

### **\*NO**

Do not replace the existing channel definition. The command fails if the named channel definition already exists.

### **\*YES**

Replace the existing channel definition. If there is no definition with the same name a new definition is created.

## Transport type (TRPTYPE)

Specifies the transmission protocol.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*LU62**

SNA LU 6.2.

**\*TCP**

Transmission Control Protocol / Internet Protocol (TCP/IP).

**Text 'description' (TEXT)**

Specifies text that briefly describes the channel definition.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Target Queue Manager (TGTMQMNAME)**

Specifies the name of the target queue manager.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The name of the target queue manager for a client connection channel (CHLTYPE) \*CLTCN is unspecified.

**message-queue-manager-name**

The name of the target message queue manager for a client connection channel (CHLTYPE) \*CLTCN.

For other channel types this parameter must not be specified.

**Connection name (CONNAME)**

Specifies the name of the machine to connect.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The connection name is blank.

**connection-name**

Specify the connection name as required by the transmission protocol:

- For \*LU62, specify the name of the CSI object.
- For \*TCP, specify either the host name, or the network address of the remote machine (or the local machine for cluster-receiver channels). This can be followed by an optional port number enclosed in parentheses.



On Multiplatforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override

the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

```
(1415)
```

The generated **CONNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

Where a port is not specified the default port 1414 is assumed.

For cluster-receiver channels the connection name relates to the local queue manager, and for other channels it relates to the target queue manager.

This parameter is required for channels with channel type (CHLTYPE) of \*SDR, \*RQSTR, \*CLTCN and \*CLUSSDR. It is optional for \*SVR and \*CLUSRCVR channels, and is not valid for \*RCVR or \*SVRCN channels.

### Transaction Program Name (TPNAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2 only.

This parameter must be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it must be set to blanks. The name is taken instead from the CPI-C Communications Side Object.

This parameter is not valid for channels with a CHLTYPE defined as \*RCVR.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*NONE**

No transaction program name is specified.

**\*BLANK**

The transaction program name is taken from CPI-C Communications Side Object. The side object name must be specified in the CONNAME parameter.

**transaction-program-name**

Specify the SNA transaction program name.

### Mode Name (MODENAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2. If TRPTYPE is not defined as LU 6.2 the data is ignored and no error message is issued.

If specified, the value must be set to the SNA mode name, unless the CONNAME contains a side-object name, in which case it must be set to blanks. The name is then taken from the CPI-C Communications Side Object.

This parameter is not valid for channels with CHLTYPE defined as \*RCVR or \*SVRCONN.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*NONE**

No mode name is specified.

**\*BLANK**

Name will be taken from the CPI-C Communications Side Object. This must be specified in the CONNAME parameter.

**SNA-mode-name**

Specify the SNA Mode Name

## Transmission queue (TMQNAME)

Specifies the name of the transmission queue.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **transmission-queue-name**

Specify the name of the transmission queue. A transmission queue name is required if the CHLTYPE is defined as \*SDR or \*SVR.

For other channel types this parameter must not be specified.

## Message channel agent (MCANAME)

This parameter is reserved and should not be used.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The MCA program name is blank.

This parameter cannot be specified if the CHLTYPE is defined as \*RCVR, \*SVRCN, or \*CLTCN.

## Message channel agent user ID (MCAUSRID)

Specifies the message channel agent user identifier which is to be used by the message channel agent for authorization to access MQ resources, including (if PUTAUT is \*DFT) authorization to put the message to the destination queue for receiver or requester channels.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The message channel agent uses its default user identifier.

### **\*PUBLIC**

Uses the public authority.

### **mca-user-identifier**

Specify the user identifier to be used.

This parameter cannot be specified for a channel type (CHLTYPE) of \*CLTCN.

## Message channel agent Type (MCATYPE)

Specifies whether the message channel agent program should run as a thread or as a process.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*PROCESS**

The message channel agent runs as a separate process.

### **\*THREAD**

The message channel agent runs as a separate thread.

This parameter can only be specified for channels with CHLTYPE defined as \*SDR, \*SVR, \*RQSTR, \*CLUSSDR or \*CLUSRCVR.

## Batch Interval (BATCHINT)

The minimum amount of time, in milliseconds, that a channel will keep a batch open.

The batch is terminated by which ever of the following occurs first: BATCHSZ messages have been sent, BATCHLIM bytes have been sent, or the transmission queue is empty and BATCHINT is exceeded.

The default value is 0, which means that the batch is terminated as soon as the transmission queue becomes empty (or the BATCHSZ limit is reached).

The value must be in the range 0 through 999999999.

This parameter is valid for channels with CHLTYPE defined as \*SDR, \*SVR, \*CLUSSDR, or \*CLUSRCVR.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **batch-interval**

Specify a value ranging from 0 through 999999999

## Batch size (BATCHSIZE)

Specifies the maximum number of messages that can be sent down a channel before a checkpoint is taken.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **batch-size**

Specify a value ranging from 1 through 9999.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

## Disconnect interval (DSCITV)

Specifies the disconnect interval, which defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before closing the channel.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **disconnect-interval**

Specify a value ranging from 0 through 999999.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR or \*CLTCN.

## Short retry interval (SHORTTMR)

Specifies the short retry wait interval for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the interval between attempts to establish a connection to the remote machine.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **short-retry-interval**

Specify a value ranging from 0 through 999999999.

## Short retry count (SHORTRTY)

Specifies the short retry count for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of attempts that are made to establish a connection to the remote machine, at intervals specified by SHORTTMR, before the (normally longer) LONGRTY and LONGTMR are used.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **short-retry-count**

Specify a value ranging from 0 through 999999999. A value of 0 means that no retries are allowed.

## Long retry interval (LONGTMR)

Specifies the long retry wait interval for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSDR or \*CLUSRCVR) that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **long-retry-interval**

Specify a value in the range 0 through 999999999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this are treated as 999999.

## Long retry count (LONGRTY)

Specifies the long retry count for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by LONGTMR, after the count specified by SHORTRTY has been exhausted. An error message is logged if the connection is not established after the defined number of attempts.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **long-retry-count**

Specify a value in the range 0 through 999999999. A value of 0 means that no retries are allowed.

## Security exit (SCYEXIT)

Specifies the name of the program to be called as the security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.

- On receipt of a response to a security message flow.

Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:



**\*SAME**

The attribute is unchanged.

**\*NONE**

The security exit program is not invoked.

**security-exit-name**

Specify the name of the security exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

**Security exit (CSCYEXIT)**

Specifies the name of the program to be called as the client security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.

- On receipt of a response to a security message flow.

Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The client security exit program is not invoked.

**security-exit-name**

Specify the name of the client security exit program.

**Security exit user data (SCYUSRDATA)**

Specifies a maximum of 32 characters of user data that is passed to the security exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the security exit program is not specified.

**security-exit-user-data**

Specify the user data for the security exit.

**Send exit (SNDEXIT)**

Specifies the entry point of the program to be called as the send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The send exit program is not invoked.

**send-exit-name**

Specify the name of the send exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

**Send exit (CSNDEXIT)**

Specifies the entry point of the program to be called as the client send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The client send exit program is not invoked.

**send-exit-name**

Specify the name of the client send exit program.

**Send exit user data (SNDUSRDATA)**

Specifies a maximum of 32 characters of user data that is passed to the send exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the send exit program is not specified.

**send-exit-user-data**

Specify the user data for the send exit program.

**Receive exit (RCVEXIT)**

Specifies the entry point of the program to be called as the receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The receive exit program is not invoked.

**receive-exit-name**

Specify the name of the receive exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

**Receive exit (RCVEXIT)**

Specifies the entry point of the program to be called as the client receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The client receive exit program is not invoked.

**receive-exit-name**

Specify the name of the client receive exit program.

**Receive exit user data (RCVUSRDATA)**

Specifies a maximum of 32 characters of user data that is passed to the receive exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the receive exit program is not specified.

**receive-exit-user-data**

Specify a maximum of 32 characters of user data for the receive exit.

**Message exit (MSGEXIT)**

Specifies the entry point of the program to be called as the message exit. If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The message exit program is not invoked.

**message-exit-name**

Specify the name of the message exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Message exit user data (MSGUSRDATA)**

Specifies user data that is passed to the message exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the message exit program is not specified.

**message-exit-user-data**

Specify a maximum of 32 characters of user data that is passed to the message exit program.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Message retry exit (MSGRTYEXIT)**

Specifies the entry point of the program to be called as the message retry exit.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The message retry exit program is not invoked.

**message-retry-exit-name**

Specify the name of the message retry exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

### **Message retry exit data (MSGRTYDATA)**

Specifies user data that is passed to the message retry exit program.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data for the message retry exit program is not specified.

**message-retry-exit-user-data**

Specify a maximum of 32 characters of user data that is passed to the message retry exit program.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

### **Number of message retries (MSGRTYNBR)**

Specifies the number of times the channel will retry before it decides it cannot deliver the message.

This parameter is used by the channel as an alternative to a message retry exit when MSGRTYEXIT is defined as \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**message-retry-number**

Specify a value ranging from 0 through 999999999. A value of 0 indicates no retries will be performed.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

### **Message retry interval (MSGRTYITV)**

Specifies the minimum interval of time that must pass before the channel can retry the MQPUT operation. This time is in milliseconds.

This parameter is used by the channel as an alternative to a message retry exit when MSGRTYEXIT is defined as \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**message-retry-number**

Specify a value ranging from 0 through 999999999. A value of 0 indicates that the retry will be performed as soon as possible.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

**Convert message (CVTMSG)**

Specifies whether the application data in the message should be converted before the message is transmitted.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*YES**

The application data in the message is converted before sending.

**\*NO**

The application data in the message is not converted before sending.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

**Put authority (PUTAUT)**

Specifies whether the user identifier in the context information associated with a message is used to establish authority to put the message on the destination queue. This applies only to receiver and requester (\*CLUSRCVR, \*RCVR and \*RQSTR) channels.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*DFT**

No authority check is made before the message is put on the destination queue.

**\*CTX**

The user identifier in the message context information is used to establish authority to put the message.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

**Sequence number wrap (SEQNUMWRAP)**

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

**Note:** The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

The possible values are:

**\*SAME**

The attribute is unchanged.

**sequence-number-wrap-value**

Specify a value ranging from 100 through 999999999.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

## Maximum message length (MAXMSGLEN)

Specifies the maximum message length that can be transmitted on the channel. This is compared with the value for the remote channel and the actual maximum is the lower of the two values.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **maximum-message-length**

Specify a value ranging from 0 through 104857600. A value of 0 indicates that the maximum length is unlimited.

## Heartbeat interval (HRTBTINTVL)

Specifies the time, in seconds, between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. The heartbeat exchange gives the receiving MCA the opportunity to quiesce the channel. This applies only to sender, server, cluster sender and cluster receiver (\*SDR, \*SVR, \*CLUSDR and \*CLUSRCVR) channels.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **heart-beat-interval**

Specify a value ranging from 0 through 999999999. A value of 0 means that no heartbeat exchanges are to take place.

## Non Persistent Message Speed (NPMSPEED)

Specifies whether the channel supports fast non persistent messages.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **\*FAST**

The channel supports fast non persistent messages.

### **\*NORMAL**

The channel does not support fast non persistent messages.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

## Cluster Name (CLUSTER)

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

This parameter is valid only for \*CLUSDR and \*CLUSRCVR channels. If the CLUSNL parameter is non-blank, this parameter must be blank.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **\*NONE**

No cluster name is specified.

### **cluster-name**

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

## Cluster Name List (CLUSNL)

The name of the namelist that specifies a list of clusters to which the channel belongs

This parameter is valid only for \*CLUSDR and \*CLUSRCVR channels. If the CLUSTER parameter is non-blank, this parameter must be blank.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **\*NONE**

No cluster namelist is specified.

### **cluster-name-list**

The name of the namelist specifying a list of clusters to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

## Network Connection Priority (NETPRTY)

The priority for the network connection. Distributed queuing chooses the path with the highest priority if there are multiple paths available. The value must be in the range between 0 and 9 where 0 is the lowest priority.

This parameter is valid only for \*CLUSRCVR channels.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **network-connection-priority**

Specify a value ranging from 0 through 9 where 0 is the lowest priority.

## TLS CipherSpec (SSLCIPH)


SSLCIPH specifies the CipherSpec used in TLS channel negotiation. The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **cipherspec**

The name of the CipherSpec.

**Note:**  From IBM MQ 8.0.0 Fix Pack 2, the SSLv3 protocol and the use of some IBM MQ CipherSpecs is deprecated. For more information, see [Deprecated CipherSpecs](#).

## TLS Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel carries out client authentication over TLS. The parameter is used only for channels with SSLCIPH specified.

The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **\*REQUIRED**

Client authentication is required.

### **\*OPTIONAL**

Client authentication is optional.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*CLTCN or \*CLUSDR.

## TLS Peer name (SSLPEER)

SSLPEER specifies the X500 peer name used in TLS channel negotiation. The possible values are:

### **\*SAME**

The value of this attribute does not change.

### **x500peername**

The X500 peer name to use.

**Note:** An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both SSLPEER on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns in order to connect. For more information, see [Channel authentication records](#).

## Local communication address (LOCLADDR)

Specifies the local communication address for the channel.

This parameter is only valid for \*SDR, \*SVR, \*RQSTR, \*CLUSSDR, \*CLUSRCVR and \*CLTCN channels.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

The connection is blank.

### **local-address**

Only valid for transport type TCP/IP. Specify the optional IP address and optional port or port range used for outbound TCP/IP communications. The format is:

```
LOCLADDR([ip-addr][low-port[,high-port]][, [ip-addr][low-port[,high-port]])
```

## Batch Heartbeat Interval (BATCHEHB)

The time in milliseconds used to determine whether batch heartbeating occurs on this channel. Batch heartbeating allows channels to determine whether the remote channel instance is still active before going indoubt. A batch heartbeat will occur if a channel MCA has not communicated with the remote channel within the specified time.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **batch-heartbeat-interval**

Specify a value ranging from 0 through 999999999. A value of 0 indicates that batch heartbeating is not to be used.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

## Task user identifier (USERID)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

This parameter is valid only for channels with a channel type (CHLTYPE) of \*SDR, \*SVR, \*RQSTR, \*CLTCN or \*CLUSSDR.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

The possible values are:



**\*SAME**

The value of this attribute does not change.

**\*NONE**

No user identifier is specified.

**user-identifier**

Specify the task user identifier.

**Password (PASSWORD)**

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

This parameter is valid only for channels with a channel type (CHLTYPE) of \*SDR, \*SVR, \*RQSTR, \*CLTCN or \*CLUSSDR.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*NONE**

No password is specified.

**password**

Specify the password.

**Keep Alive Interval (KAINT)**

Specifies the keep alive timing interval for this channel.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*AUTO**

The keep alive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated HBINT is greater than 0, keep alive interval is set to that value plus 60 seconds.
- If the negotiated HBINT is 0, the value used is that specified by the KEEPALIVEOPTIONS statement in the TCP profile configuration data set.

**keep-alive-interval**

Specify a value ranging from 0 through 99999.

**Header Compression (COMPHDR)**

The list of header data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (\*SDR, \*SVR, \*CLUSSDR, \*CLUSRCVR and \*CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No header data compression is performed.

**\*SYSTEM**

Header data compression is performed.

## Message Compression (COMPMSG)

The list of message data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (\*SDR, \*SVR, \*CLUSDR, \*CLUSRCVR and \*CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

No message data compression is performed.

### **\*RLE**

Message data compression is performed using run-length encoding.

### **\*ZLIBFAST**

Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

### **\*ZLIBHIGH**

Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

### **\*ANY**

Any compression technique supported by the queue manager can be used. This option is only valid for channel types receiver, requester and server connection (\*RCVR, \*RQSTR and \*SVRCN).

## Channel Monitoring (MONCHL)

Controls the collection of online monitoring data.

Online monitoring data is not collected when the queue manager attribute MONCHL is set to \*NONE.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*QMGR**

The collection of online monitoring data is inherited from the setting of the queue manager attribute MONCHL.

### **\*OFF**

Online Monitoring Data collection for this channel is disabled.

### **\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

### **\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.

### **\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for a channel type (CHLTYPE) of \*CLTCN.

## Channel Statistics (STATCHL)

Controls the collection of statistics data.

Statistics data is not collected when the queue manager attribute STATCHL is set to \*NONE.

The possible values are:

### **\*SAME**

The attribute is unchanged.

**\*QMGR**

Statistics data collection is based upon the setting of the queue manager attribute STATCHL.

**\*OFF**

Statistics data collection for this channel is disabled.

**\*LOW**

Statistics data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Statistics data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Statistics data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Cluster Workload Rank (CLWLRANK)**

Specifies the cluster workload rank of the channel.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-rank**

The cluster workload rank of the channel in the range 0 through 9.

**Cluster Workload Priority (CLWLPRTY)**

Specifies the cluster workload priority of the channel.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-priority**

The cluster workload priority of the channel in the range 0 through 9.

**Cluster Channel Weight (CLWLWGHT)**

Specifies the cluster workload weight of the channel.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-weight**

The cluster workload weight of the channel in the range 1 through 99.

**Sharing Conversations (SHARECNV)**

Specifies the maximum the number of conversations which can be shared over a particular TCP/IP client channel instance (socket).

This parameter is valid for channels with CHLTYPE defined as \*CLTCN or \*SVRCN.

The possible values are:

**\*SAME**

The attribute is unchanged.

**0**

Specifies no sharing of conversations over a TCP/IP socket. The channel instance runs in a mode prior to that of IBM WebSphere MQ 7.0, with regard to:

- Administrator stop-quiesce
- Heartbeating
- Read ahead

## 1

Specifies no sharing of conversations over a TCP/IP socket. Client heartbeating and read ahead are available, whether in an MQGET call or not, and channel quiescing is more controllable.

### **shared-conversations**

The number of shared conversations in the range 2 through 999999999.

This parameter is only valid for client-connection and server-connection channels.

**Note:** If the client-connection SHARECNV value does not match the server-connection SHARECNV value, the lower of the two values is used.

## **Property Control (PROPCTL)**

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor).

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*COMPAT**

If the message contains a property with a prefix of "mcd.", "jms.", "usr." or "mqext." then all optional message properties, except those in the message descriptor (or extension) will be placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.

### **\*NONE**

All properties of the message, except those in the message descriptor (or extension), will be removed from the message before the message is sent to the remote queue manager.

### **\*ALL**

All properties of the message will be included with the message when it is sent to the remote queue manager. The properties, except those in the message descriptor (or extension), will be placed in one or more MQRFH2 headers in the message data.

## **Maximum Instances (MAXINST)**

Specifies the maximum number of clients that can simultaneously connect to the queue manager via this server-connection channel object.

This attribute is valid only for server-connection channels.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **maximum-instances**

The maximum number of simultaneous instances of the channel in the range 0 through 999999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

## **Maximum Instances Per Client (MAXINSTC)**

Specifies the maximum number of simultaneous instances of an individual server-connection channel which can be started from a single client.

In this context, multiple client connections originating from the same remote network address are considered to be a single client.

This attribute is valid only for server-connection channels.

The possible values are:

**\*SAME**

The attribute is unchanged.

**maximum-instances-per-client**

The maximum number of simultaneous instances of the channel which can be in the started from a single client in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running from individual clients, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

## Client Channel Weight (CLNTWGHT)

The client channel weighting attribute is used so client channel definitions can be selected at random based on their weighting when more than one suitable definition is available.

The possible values are:

**\*SAME**

The attribute is unchanged.

**client-channel-weight**

The client channel weight in the range 0 through 99.

## Connection Affinity (AFFINITY)

The channel affinity attribute is used so client applications that connect multiple times using the same queue manager name can choose whether to use the same client channel definition for each connection.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*PREFERRED**

The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions based on the weighting with any applicable CLNTWGHT(0) definitions first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non CLNTWGHT(0) definitions are moved to the end of the list. CLNTWGHT(0) definitions remain at the start of the list and are selected first for each connection.

**\*NONE**

The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable CLNTWGHT(0) definitions selected first in alphabetical order.

## Batch Data Limit (BATCHLIM)

The limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point. A sync point is taken after the message that caused the limit to be reached has flowed across the channel. A value of zero in this attribute means that no data limit is applied to batches over this channel.

The batch is terminated when one of the following conditions is met:

- **BATCHSZ** messages have been sent.
- **BATCHLIM** bytes have been sent.
- The transmission queue is empty and **BATCHINT** is exceeded.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

The value must be in the range 0 - 999999. The default value is 5000.

The **BATCHLIM** parameter is supported on all platforms.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**batch-data-limit**

Specify a value ranging from 0 through 999999.

This parameter can only be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLUSSDR, or \*CLUSRCVR.

**Default client reconnection (DFTRECON)**

Specifies whether a client connection automatically reconnects a client application if its connection breaks.

**\*SAME**

The value of this attribute does not change.

**\*NO**

Unless overridden by **MQCONN**, the client is not reconnected automatically.

**\*YES**

Unless overridden by **MQCONN**, the client reconnects automatically.

**\*QMGR**

Unless overridden by **MQCONN**, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as MQCNO\_RECONNECT\_Q\_MGR.

**\*DISABLED**

Reconnection is disabled, even if requested by the client program using the **MQCONN** MQI call.

This parameter is specified for a client connection channel, (CHLTYPE) \*CLTCN

 **CPYMQMLSR (Copy MQ Listener)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Copy MQ Listener (CPYMQMLSR) command creates an MQ listener definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing listener definition.

**Parameters**

<i>Table 235. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>FROMLSR</u>	From Listener	Character value	Required, Key, Positional 1
<u>TOLSR</u>	To Listener	Character value	Required, Key, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 3
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 4

Table 235. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>TEXT</u>	Text 'description'	Character value, *BLANK, *SAME	Optional, Positional 5
<u>CONTROL</u>	Listener control	*SAME, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 6
<u>PORT</u>	Port number	0-65535, *SAME	Optional, Positional 7
<u>IPADDR</u>	IP Address	Character value, *BLANK, *SAME	Optional, Positional 8
<u>BACKLOG</u>	Listener backlog	0-999999999, *SAME	Optional, Positional 9

### From Listener (FROMLSR)

Specifies the name of the existing listener definition to provide values for the attributes not specified in this command.

The possible values are:

#### from-listener-name

Specify the name of the source MQ listener.

### To Listener (TOLSR)

Specifies the name of the new listener definition to be created. The name can contain a maximum of 48 characters.

If a listener definition with this name already exists, REPLACE(\*YES) must be specified.

The possible values are:

#### to-listener-name

Specify the name of the new listener being created.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.

### Replace (REPLACE)

Specifies whether the new listener definition will replace an existing listener definition with the same name.

The possible values are:

#### \*NO

This definition does not replace any existing listener definition with the same name. The command fails if the named listener definition already exists.

#### \*YES

Replace the existing listener definition. If there is no definition with the same name, a new definition is created.

## Text 'description' (TEXT)

Specifies text that briefly describes the listener definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*BLANK**

The text is set to a blank string.

### **description**

Specify no more than 64 characters enclosed in apostrophes.

## Listener control (CONTROL)

Whether the listener starts automatically when the queue manager is started.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*MANUAL**

The listener is not automatically started or stopped.

### **\*QMGR**

The listener is started and stopped as the queue manager is started and stopped.

### **\*STARTONLY**

The listener is started as the queue manager is started, but is not automatically stopped when the queue manager is stopped.

## Port number (PORT)

The port number to be used by the listener.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **port-number**

The port number to be used.

## IP Address (IPADDR)

The IP address to be used by the listener.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **ip-addr**

The IP address to be used.

## Listener backlog (BACKLOG)

The number of concurrent connection requests the listener supports.

The possible values are:



**\*SAME**

The attribute is unchanged.

**backlog**

The number of concurrent connection requests supported.


**CPYMQMNL (Copy MQ Namelist)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Copy MQ Namelist (CPYMQMNL) command copies an MQ namelist.

**Parameters**

<i>Table 236. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>FROMNL</u>	From Namelist	Character value	Required, Key, Positional 1
<u>TONL</u>	To Namelist	Character value	Required, Key, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 3
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 4
<u>TEXT</u>	Text 'description'	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 5
<u>NAMES</u>	List of Names	Values (up to 256 repetitions): <i>Character value</i> , *BLANKS, <b>*SAME</b> , *NONE	Optional, Positional 6

**From Namelist (FROMNL)**

Specifies the name of the existing namelist, to provide values for the attributes not specified in this command.

**from-namelist**

Specify the name of the source namelist.

**To Namelist (TONL)**

The name of the new namelist to be created. The name can contain a maximum of 48 characters.

If a namelist with this name already exists, REPLACE(\*YES) must be specified.

**to-namelist**

Specify the name of the MQ namelist being created.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used.

**message-queue-manager-name**

Specify the name of the queue manager.

**Replace (REPLACE)**

Specifies whether the new namelist should replace an existing namelist with the same name.

**\*NO**

Do not replace the existing namelist. The command fails if the named namelist already exists.

**\*YES**

Replace the existing namelist. If there is no namelist with the same name, a new namelist is created.

**Text 'description' (TEXT)**

Specifies text that briefly describes the namelist.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**\*SAME**

The attribute is unchanged.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**List of Names (NAMES)**

List of names. This is the list of names to be created. The names can be of any type, but must conform to the rules for naming MQ objects.

**\*SAME**

The attribute is unchanged.

**namelist**

The list to create. An empty list is valid.

 **CPYMQMPCRC (Copy MQ Process)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Copy MQ Process (CPYMQMPCRC) command creates an MQ process definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing process definition.

**Parameters**

<i>Table 237. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>FROMPRC</u>	From process	Character value	Required, Key, Positional 1
<u>TOPRC</u>	To process	Character value	Required, Key, Positional 2

Table 237. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Key, Positional 3
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 4
<u>TEXT</u>	Text 'description'	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 5
<u>APPTYPE</u>	Application type	Integer, <b>*DEF</b> , *CICS, *UNIX, *OS400, *WINDOWS, *WINDOWS_NT,	Optional, Positional 6
<u>APPID</u>	Application identifier	Character value, <b>*SAME</b>	Optional, Positional 7
<u>USRDATA</u>	User data	Character value, <b>*SAME</b> , *NONE	Optional, Positional 8
<u>ENVDATA</u>	Environment data	Character value, <b>*SAME</b> , *NONE	Optional, Positional 9

### From process (FROMPRC)

Specifies the name of the existing process definition to provide values for the attributes not specified in this command.

The possible values are:

#### from-process-name

Specify the name of the source MQ process.

### To process (TOPRC)

The name of the new process definition to be created. The name can contain a maximum of 48 characters.

If a process definition with this name already exists, REPLACE(\*YES) must be specified.

The possible values are:

#### to-process-name

Specify the name of the MQ process being created.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.

### Replace (REPLACE)

Specifies whether the new process definition should replace an existing process definition with the same name.

The possible values are:

**\*NO**

This definition does not replace any existing process definition with the same name. The command fails if the named process definition already exists.

**\*YES**

Replace the existing process definition. If there is no definition with the same name, a new definition is created.

**Text 'description' (TEXT)**

Specifies text that briefly describes the process definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**Application type (APPTYPE)**

The type of application started.

The possible values are:

**\*DEF**

Specifying DEF causes the default application type for the platform at which the command is interpreted to be stored in the process definition. This default cannot be changed by the installation. If the platform supports clients, the default is interpreted as the default application type of the server.

**\*CICS**

Represents a CICS/400 application.

**\*UNIX**

Represents a UNIX or Linux application.

**\*OS400**

Represents an IBM i application.

**\*WINDOWS**

Represents a Windows application.

**\*WINDOWS\_NT**

Represents a Windows NT application.

**integer**

User-defined application type in the range 65536 through 999999999.

**Application identifier (APPID)**

Application identifier. This is the name of the application to be started, on the platform for which the command is processing. It is typically a program name and library name.

The possible values are:

**\*SAME**

The attribute is unchanged.

**application-id**

The maximum length is 256 characters.

## User data (USRDATA)

A character string that contains user information pertaining to the application, as defined by APPID, to start.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The user data is blank.

**user-data**

Specify up to 128 characters of user data.

## Environment data (ENVDATA)

A character string that contains environment information pertaining to the application, as defined by APPID, to start.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The environment data is blank.

**environment-data**

The maximum length is 128 characters.

## CPYMQMQ (Copy MQ Queue)

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Copy MQ Queue ( **CPYMQMQ** ) command creates a queue definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing queue definition.

## Parameters

Table 238. Command parameters

Keyword	Description	Choices	Notes
<u>FROMQ</u>	From queue name	Character value	Required, Key, Positional 1
<u>TOQ</u>	To queue name	Character value	Required, Key, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Key, Positional 3
<u>QTYPE</u>	Queue type	Character value	Optional, Positional 4
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 5
<u>TEXT</u>	Text 'description'	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 6
<u>PUTENBL</u>	Put enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 7

Table 238. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<a href="#">DFTPTY</a>	Default message priority	0-9, <b>*SAME</b>	Optional, Positional 8
<a href="#">DFTMSGPST</a>	Default message persistence	<b>*SAME</b> , *NO, *YES	Optional, Positional 9
<a href="#">PRCNAME</a>	Process name	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 10
<a href="#">TRGENBL</a>	Triggering enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 11
<a href="#">GETENBL</a>	Get enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 12
<a href="#">SHARE</a>	Sharing enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 13
<a href="#">DFTSHARE</a>	Default share option	<b>*SAME</b> , *NO, *YES	Optional, Positional 14
<a href="#">MSGDLYSEQ</a>	Message delivery sequence	<b>*SAME</b> , *PTY, *FIFO	Optional, Positional 15
<a href="#">HDNBKTCNT</a>	Harden backout count	<b>*SAME</b> , *NO, *YES	Optional, Positional 16
<a href="#">TRGTYPE</a>	Trigger type	<b>*SAME</b> , *FIRST, *ALL, *DEPTH, *NONE	Optional, Positional 17
<a href="#">TRGDEPTH</a>	Trigger depth	1-999999999, <b>*SAME</b>	Optional, Positional 18
<a href="#">TRGMSGPTY</a>	Trigger message priority	0-9, <b>*SAME</b>	Optional, Positional 19
<a href="#">TRGDATA</a>	Trigger data	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 20
<a href="#">RTNITV</a>	Retention interval	0-999999999, <b>*SAME</b>	Optional, Positional 21
<a href="#">MAXDEPTH</a>	Maximum queue depth	0-999999999, <b>*SAME</b>	Optional, Positional 22
<a href="#">MAXMSGLEN</a>	Maximum message length	0-104857600, <b>*SAME</b>	Optional, Positional 23
<a href="#">BKTTHLD</a>	Backout threshold	0-999999999, <b>*SAME</b>	Optional, Positional 24
<a href="#">BKTQNAME</a>	Backout requeue name	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 25
<a href="#">INITQNAME</a>	Initiation queue	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 26
<a href="#">USAGE</a>	Usage	<b>*SAME</b> , *NORMAL, *TMQ	Optional, Positional 27
<a href="#">DFNTYPE</a>	Definition type	<b>*SAME</b> , *TEMPDYN, *PERMDYN	Optional, Positional 28
<a href="#">TGTQNAME</a>	Target object	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 29
<a href="#">RMTQNAME</a>	Remote queue	<i>Character value</i> , <b>*SAME</b> , *NONE	Optional, Positional 30
<a href="#">RMTMQMNAME</a>	Remote Message Queue Manager	<i>Character value</i> , <b>*SAME</b>	Optional, Positional 31
<a href="#">TMQNAME</a>	Transmission queue	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 32
<a href="#">HIGHTHLD</a>	Queue depth high threshold	0-100, <b>*SAME</b>	Optional, Positional 33

Table 238. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>LOWTHLD</u>	Queue depth low threshold	0-100, <b>*SAME</b>	Optional, Positional 34
<u>FULLEVT</u>	Queue full events enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 35
<u>HIGHEVT</u>	Queue high events enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 36
<u>LOWEVT</u>	Queue low events enabled	<b>*SAME</b> , *NO, *YES	Optional, Positional 37
<u>SRVITV</u>	Service interval	0-999999999, <b>*SAME</b>	Optional, Positional 38
<u>SRVEVT</u>	Service interval events	<b>*SAME</b> , *HIGH, *OK, *NONE	Optional, Positional 39
<u>DISTLIST</u>	Distribution list support	<b>*SAME</b> , *NO, *YES	Optional, Positional 40
<u>CLUSTER</u>	Cluster Name	Character value, <b>*SAME</b> , *NONE	Optional, Positional 41
<u>CLUSNL</u>	Cluster Name List	Character value, *NONE, <b>*SAME</b>	Optional, Positional 42
<u>DEFBIND</u>	Default Binding	<b>*SAME</b> , *OPEN, *NOTFIXED, *GROUP	Optional, Positional 43
<u>CLWLRANK</u>	Cluster Workload Rank	0-9, <b>*SAME</b>	Optional, Positional 44
<u>CLWLPRTY</u>	Cluster Workload Priority	0-9, <b>*SAME</b>	Optional, Positional 45
<u>CLWLUSEQ</u>	Cluster workload queue use	<b>*SAME</b> , *QMGR, *LOCAL, *ANY	Optional, Positional 46
<u>MONQ</u>	Queue Monitoring	<b>*SAME</b> , *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 47
<u>STATQ</u>	Queue Statistics	<b>*SAME</b> , *QMGR, *OFF, *ON	Optional, Positional 48
<u>ACCTQ</u>	Queue Accounting	<b>*SAME</b> , *QMGR, *OFF, *ON	Optional, Positional 49
<u>NPMCLASS</u>	Non Persistent Message Class	<b>*SAME</b> , *NORMAL, *HIGH	Optional, Positional 50
<u>MSGREADAHD</u>	Message Read Ahead	<b>*SAME</b> , *DISABLED, *NO, *YES	Optional, Positional 51
<u>DFTPURRESP</u>	Default Put Response	<b>*SAME</b> , *SYNC, *ASYN	Optional, Positional 52
<u>PROPCTL</u>	Property Control	<b>*SAME</b> , *COMPAT, *NONE, *ALL, *FORCE, *V6COMPAT	Optional, Positional 53
<u>TARGETYPE</u>	Target Type	<b>*SAME</b> , *QUEUE, *TOPIC	Optional, Positional 54
<u>CUSTOM</u>	Custom attribute	Character value, *BLANK, <b>*SAME</b>	Optional, Positional 55
<u>CLCHNAME</u>	Cluster-sender channel name	Character value, *NONE, <b>*SAME</b>	Optional, Positional 56

Table 238. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>IMGRCOVQ</u>	Queue object attribute	<b>*SAME</b> , *NO, *YES, *QMGR	Optional, Positional 58

### From queue name (FROMQ)

Specifies the name of the existing queue definition, to provide values for the attributes not specified in this command.

The possible values are:

#### from-queue-name

Specify the name of the source queue.

### To queue name (TOQ)

Specifies the name of the new queue definition. The name can contain a maximum of 48 characters. Queue name and type combinations must be unique; if a queue definition already exists with the name and type of the new queue, REPLACE(\*YES) must be specified.

**Note:** The field length is 48 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

The possible values are:

#### to-queue-name

Specify the name of the queue being created.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

Specify the name of the queue manager.

### Queue type (QTYPE)

Specifies the type of queue that is to be copied.

The possible values are:

#### \*ALS

An alias queue.

#### \*LCL

A local queue.

#### \*RMT

A remote queue.

#### \*MDL

A model queue.

### Replace (REPLACE)

Specifies whether the new queue will replace an existing queue definition with the same name and type.

The possible values are:



**\*NO**

Do not replace the existing queue definition. The command fails if the named queue already exists.

**\*YES**

Replace the existing queue definition with the attributes of the FROMQ and the specified attributes.

The command fails if an application has the queue open or the USAGE attribute is changed.

**Note:** If the queue is a local queue, and a queue with the same name already exists, any messages already on that queue are retained.

**Text 'description' (TEXT)**

Specifies text that briefly describes the object.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Put enabled (PUTENBL)**

Specifies whether messages can be put on the queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Messages cannot be added to the queue.

**\*YES**

Messages can be added to the queue by authorized applications.

**Default message priority (DFTPTY)**

Specifies the default priority of messages put on the queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**priority-value**

Specify a value ranging from 0 through 9, where 9 is the highest priority.

**Default message persistence (DFTMSGPST)**

Specifies the default for message-persistence on the queue. Message persistence determines whether messages are preserved across restarts of the queue manager.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

By default, messages are lost across a restart of the queue manager.

**\*YES**

By default, messages are preserved across a restart of the queue manager.

**Process name (PRCNAME)**

Specifies the local name of the MQ process that identifies the application that should be started when a trigger event occurs.

The process does not have to be available when the queue is created, but it must be available for a trigger event to occur.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

The process name is blank.

**process-name**

Specify the name of the MQ process.

**Triggering enabled (TRGENBL)**

Specifies whether trigger messages are written to the initiation queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Triggering is not enabled. Trigger messages are not written to the initiation queue.

**\*YES**

Triggering is enabled. Trigger messages are written to the initiation queue.

**Get enabled (GETENBL)**

Specifies whether applications are to be permitted to get messages from this queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Applications cannot retrieve messages from the queue.

**\*YES**

Suitably authorized applications can retrieve messages from the queue.

**Sharing enabled (SHARE)**

Specifies whether multiple instances of applications can open this queue for input simultaneously.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Only a single application instance can open the queue for input.

**\*YES**

More than one application instance can open the queue for input.

**Default share option (DFTSHARE)**

Specifies the default share option for applications opening this queue for input.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

By default, the open request is for exclusive use of the queue for input.

**\*YES**

By default, the open request is for shared use of the queue for input.

**Message delivery sequence (MSGDLYSEQ)**

Specifies the message delivery sequence.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*PTY**

Messages are delivered in first-in-first-out (FIFO) order within priority.

**\*FIFO**

Messages are delivered in FIFO order regardless of priority.

**Harden backout count (HDNBKTCNT)**

Specifies whether the count of backed out messages is saved (hardened) across restarts of the message queue manager.

**Note:** On IBM MQ for IBM i the count is ALWAYS hardened, regardless of the setting of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

The backout count is not hardened.

**\*YES**

The backout count is hardened.

**Trigger type (TRGTYPE)**

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*FIRST**

When the number of messages on the queue goes from 0 to 1.

**\*ALL**

Every time a message arrives on the queue.

**\*DEPTH**

When the number of messages on the queue equals the value of the TRGDEPTH attribute.

**\*NONE**

No trigger messages are written.

**Trigger depth (TRGDEPTH)**

Specifies, for TRIGTYPE(\*DEPTH), the number of messages that initiate a trigger message to the initiation queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**depth-value**

Specify a value ranging from 1 through 999999999.

**Trigger message priority (TRGMSGPTY)**

Specifies the minimum priority that a message must have before it can produce, or be counted for, a trigger event.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**priority-value**

Specify a value ranging from 0 through 9, where 9 is the highest priority.

**Trigger data (TRGDATA)**

Specifies up to 64 characters of user data that the queue manager includes in the trigger message. This data is made available to the monitoring application that processes the initiation queue, and to the application started by the monitor.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No trigger data is specified.

**trigger-data**

Specify up to 64 characters enclosed in apostrophes. For a transmission queue you can use this parameter to specify the name of the channel to be started.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Retention interval (RTNITV)**

Specifies the retention interval. This interval is the number of hours for which the queue might be needed, based on the date and time when the queue was created.

This information is available to a housekeeping application or an operator and can be used to determine when a queue is no longer required.

**Note:** The message queue manager does not delete queues, nor does it prevent your queues from being deleted if their retention interval has not expired. It is your responsibility to take any required action.

The possible values are:

**\*SAME**

The attribute is unchanged.

**interval-value**

Specify a value ranging from 0 through 999999999.

### **Maximum queue depth (MAXDEPTH)**

Specifies the maximum number of messages allowed on the queue. However, other factors can cause the queue to be treated as full; for example, it appears to be full if there is no storage available for a message.

**Note:** If this value is subsequently reduced by using the CHGMQM command, any messages that are on the queue remain intact even if they cause the new maximum to be exceeded.

The possible values are:

**\*SAME**

The attribute is unchanged.

**depth-value**

Specify a value ranging from 0 through 999999999.

### **Maximum message length (MAXMSGLEN)**

Specifies the maximum length for messages on the queue.

**Note:** If this value is subsequently reduced by using the CHGMQM command, any messages that are on the queue remain intact even if they exceed the new maximum length.

Applications might use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore change the value only if you know this will not cause an application to operate incorrectly.

The possible values are:

**\*SAME**

The attribute is unchanged.

**length-value**

Specify a value ranging from 0 through 100 MB in bytes. The default is 4MB.

### **Backout threshold (BKTTHLD)**

Specifies the backout threshold.

Applications running inside of WebSphere Application Server and those that use the IBM MQ Application Server Facilities will use this attribute to determine if a message should be backed out. For all other applications, apart from allowing this attribute to be queried, the queue manager takes no action based on the value of the attribute.

The possible values are:

**\*SAME**

The attribute is unchanged.

**threshold-value**

Specify a value ranging from 0 through 999999999.

## Backout requeue name (BKTQNAME)

Specifies the backout-queue name.

Applications running inside of WebSphere Application Server and those that use the IBM MQ Application Server Facilities will use this attribute to determine where messages that have been backed out should go. For all other applications, apart from allowing this attribute to be queried, the queue manager takes no action based on the value of the attribute.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

No backout queue is specified.

### **backout-queue-name**

Specify the backout queue name.

## Initiation queue (INITQNAME)

Specifies the name of the initiation queue.

**Note:** The initiation queue must be on the same instance of a message queue manager.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NONE**

No initiation queue is specified.

### **initiation-queue-name**

Specify the initiation queue name.

## Usage (USAGE)

Specifies whether the queue is for normal usage, or for transmitting messages to a remote message queue manager.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*NORMAL**

Normal usage (the queue is not a transmission queue)

### **\*TMQ**

The queue is a transmission queue that is used to hold messages destined for a remote message queue manager. If the queue is intended for use in situations where a transmission queue name is not explicitly specified, the queue name must be the same as the name of the remote message queue manager. For further information, see IBM MQ Intercommunication.

## Definition type (DFNTYPE)

Specifies the type of dynamic queue definition that is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor.

**Note:** This parameter only applies to a model queue definition.

The possible values are:

### **\*SAME**

The attribute is unchanged.

**\*TEMPDYN**

A temporary dynamic queue is created. This value should not be specified with a DEFMSGPST value of \*YES.

**\*PERMDYN**

A permanent dynamic queue is created.

**Target object (TGTQNAME)**

Specifies the name of the object for which this queue is an alias.

The object can be a local or remote queue, a topic or a message queue manager.

**Note:** The target object does not need to exist at this time but it must exist when a process attempts to open the alias queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**target-object-name**

Specify the name of the target object.

**Remote queue (RMTQNAME)**

Specifies the name of the remote queue. That is, the local name of the remote queue as defined on the queue manager specified by RMTMQMNAME.

If this definition is used for a queue manager alias definition, RMTQNAME must be blank when the open occurs.

If this definition is used for a reply-to alias, this name is the name of the queue that is to be the reply-to queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No remote-queue name is specified (that is, the name is blank). This can be used if the definition is a queue manager alias definition.

**remote-queue-name**

Specify the name of the queue at the remote queue manager.

**Note:** The name is not checked to ensure that it contains only those characters normally allowed for queue names.

**Remote Message Queue Manager (RMTMQMNAME)**

Specifies the name of the remote queue manager on which the queue RMTQNAME is defined.

If an application opens the local definition of a remote queue, RMTMQMNAME must not be the name of the connected queue manager. If TMQNAME is blank there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a queue manager alias, RMTMQMNAME is the name of the queue manager, which can be the name of the connected queue manager. Otherwise, if TMQNAME is blank, when the queue is opened there must be a local queue of this name, with USAGE(\*TMQ) specified, which is to be used as the transmission queue.

If this definition is used for a reply-to alias, this name is the name of the queue manager that is to be the reply-to queue manager.

The possible values are:

**\*SAME**

The attribute is unchanged.

**remote-queue-manager-name**

Specify the name of the remote queue manager.

**Note:** Ensure this name contains only those characters normally allowed for queue manager names.

**Transmission queue (TMQNAME)**

Specifies the local name of the transmission queue to be used for messages destined for the remote queue, for either a remote queue or for a queue manager alias definition.

If TMQNAME is blank, a queue with the same name as RMTMQMNAME is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue manager alias and RMTMQMNAME is the name of the connected queue manager.

It is also ignored if the definition is used as a reply-to queue alias definition.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No specific transmission queue name is defined for this remote queue. The value of this attribute is set to all blanks.

**transmission-queue-name**

Specify the transmission queue name.

**Queue depth high threshold (HIGHTHLD)**

Specifies the threshold against which the queue depth is compared to generate a queue depth high event.

The possible values are:

**\*SAME**

The attribute is unchanged.

**threshold-value**

Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

**Queue depth low threshold (LOWTHLD)**

Specifies the threshold against which the queue depth is compared to generate a queue depth low event.

The possible values are:

**\*SAME**

The attribute is unchanged.

**threshold-value**

Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

**Queue full events enabled (FULLEVT)**

Specifies whether queue full events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.



**\*NO**

Queue full events are not generated.

**\*YES**

Queue full events are generated.

**Queue high events enabled (HIGHEVT)**

Specifies whether queue depth high events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Queue depth high events are not generated.

**\*YES**

Queue depth high events are generated.

**Queue low events enabled (LOWEVT)**

Specifies whether queue depth low events are generated.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

Queue depth low events are not generated.

**\*YES**

Queue depth low events are generated.

**Service interval (SRVITV)**

Specifies the service interval. This interval is used for comparison to generate service interval high and service interval OK events.

The possible values are:

**\*SAME**

The attribute is unchanged.

**interval-value**

Specify a value ranging from 0 through 999999999. The value is in units of milliseconds.

**Service interval events (SRVEVT)**

Specifies whether service interval high or service interval OK events are generated.

A service interval high event is generated when a check indicates that no messages have been retrieved from the queue for the time indicated by the SRVITV parameter as a minimum.

A service interval OK event is generated when a check indicates that messages have been retrieved from the queue within the time indicated by the SRVITV parameter.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*HIGH**

Service interval high events are generated.

**\*OK**

Service interval OK events are generated.

**\*NONE**

No service interval events are generated.

**Distribution list support (DISTLIST)**

Specifies whether the queue supports distribution lists.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NO**

The queue will not support distribution lists.

**\*YES**

The queue will support distribution lists.

**Cluster Name (CLUSTER)**

The name of the cluster to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are already open.

This parameter cannot be set for dynamic, transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx or SYSTEM.COMMAND.xx queues.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-name**

Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

**Cluster Name List (CLUSNL)**

The name of the namelist which specifies a list of clusters to which the queue belongs. Changes to this parameter do not affect instances of the queue that are already open.

This parameter cannot be set for dynamic, transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx or SYSTEM.COMMAND.xx queues.

The possible values are:

**\*SAME**

The attribute is unchanged.

**namelist-name**

Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

**Default Binding (DEFBIND)**

Specifies the binding to be used when the application specifies MQOO\_BIND\_AS\_Q\_DEF on the MQOPEN call and the queue is a cluster queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*OPEN**

The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

**\*NOTFIXED**

The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using MQPUT and to change that selection subsequently if necessary.

The MQPUT1 call always behaves as if NOTFIXED had been specified.

**\*GROUP**

When the queue is opened, the queue handle is bound to a specific instance of the cluster queue for as long as there are messages in a message group. All messages in a message group are allocated to the same destination instance.

**Cluster Workload Rank (CLWLRANK)**

Specifies the cluster workload rank of the queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-rank**

Specify a value ranging from 0 through 9.

**Cluster Workload Priority (CLWLPRTY)**

Specifies the cluster workload priority of the queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**cluster-workload-priority**

Specify a value ranging from 0 through 9.

**Cluster workload queue use (CLWLUSEQ)**

Specifies the behavior of an MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel then this attribute does not apply.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

The value is inherited from the Queue Manager CLWLUSEQ attribute.

**\*LOCAL**

The local queue will be the sole target of the MQPUT.

**\*ANY**

The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

**Queue Monitoring (MONQ)**

Controls the collection of Online Monitoring Data.

Online Monitoring Data is not collected when the queue manager attribute MONQ is set to \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

The collection of online monitoring data is inherited from the setting of the queue manager attribute MONQ.

**\*OFF**

Online monitoring data collection for this queue is disabled.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

**Queue Statistics (STATQ)**

Controls the collection of statistics data.

Online monitoring data is not collected when the queue manager attribute STATQ is set to \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

Statistics data collection is based upon the setting of the queue manager attribute STATQ.

**\*OFF**

Statistics data collection for this queue is disabled.

**\*ON**

Statistics data collection is enabled for this queue.

**Queue Accounting (ACCTQ)**

Controls the collection of accounting data.

Accounting data is not collected when the queue manager attribute ACCTQ is set to \*NONE.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QMGR**

Accounting data collection is based upon the setting of the queue manager attribute ACCTQ.

**\*OFF**

Accounting data collection for this queue is disabled.

**\*ON**

Accounting data collection is enabled for this queue.

**Non Persistent Message Class (NPMCLASS)**

Specifies the level of reliability for non-persistent messages put to this queue.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NORMAL**

Non-persistent messages put to this queue are only lost following a failure, or a queue manager shutdown. Non-persistent message put to this queue will be discarded in the event of a queue manager restart.

**\*HIGH**

Non-persistent messages put to this queue are not discarded in the event of a queue manager restart. Non-persistent messages put to this queue may still be lost in the event of a failure.

**Message Read Ahead (MSGREADAHD)**

Specifies whether non persistent messages are sent to the client ahead of an application requesting them.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*DISABLED**

Read ahead is disabled for this queue. Messages are not sent to the client ahead of an application requesting them regardless of whether read ahead is requested by the client application.

**\*NO**

Non-persistent messages are not sent to the client ahead of an application requesting them. A maximum of one non-persistent message can be lost if the client ends abnormally.

**\*YES**

Non-persistent messages are sent to the client ahead of an application requesting them. Non-persistent messages can be lost if the client ends abnormally or if the client application does not consume all the messages it is sent.

**Default Put Response (DFTPUTRESP)**

The default put response type (DFTPUTRESP) attribute specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO\_RESPONSE\_AS\_Q\_DEF option.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*SYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application. This is the default value supplied with IBM MQ, but your installation might have changed it.

**\*ASYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application; but an improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

**Property Control (PROPCTL)**

Specifies what happens to properties of messages that are retrieved from queues using the MQGET call when the MQGMO\_PROPERTIES\_AS\_Q\_DEF option is specified.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*COMPAT**

If the message contains a property with a prefix of `mcd.`, `jms.`, `usr.` or `mqext.` then all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

**\*NONE**

All properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

**\*ALL**

All properties of the message, except those contained in the message descriptor (or extension), are contained in one or more MQRFH2 headers in the message data.

**\*FORCE**

Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

**\*V6COMPAT**

When set, `*V6COMPAT` must be set both on one of the queue definitions resolved by MQPUT and one of the queue definitions resolved by MQGET. It must also be set on any other intervening transmission queues. It causes an MQRFH2 header to be passed unchanged from the sending application to the receiving application. It overrides other settings of **PROPCTL** found in a queue name resolution chain. If the property is set on a cluster queue, the setting is not cached locally on other queue managers. You must set `*V6COMPAT` on an alias queue that resolves to the cluster queue. Define the alias queue on the same queue manager that the putting application is connected to.

**Target Type (TARGTYPE)**

Specifies the type of object to which the alias resolves.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*QUEUE**

Queue object.

**\*TOPIC**

Topic object.

**Custom attribute (CUSTOM)**

This attribute is reserved for the configuration of new features before separate attributes have been introduced. This description will be updated when features using this attribute are introduced. At the moment there are no meaningful values for *CUSTOM*, so leave it empty.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**custom**

Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs must have the form `NAME (VALUE)` and be specified in uppercase. Single quotes must be escaped with another single quote.

**CLCHNAME**

This parameter is supported only on transmission queues.

**\*SAME**

The attribute is unchanged.

**\*NONE**

The attribute is removed.

**cluster-sender channel name**

`ClusterChannelName` is the generic name of the cluster-sender channels that use this queue as a transmission queue. The attribute specifies which cluster-sender channels send messages to a cluster-receiver channel from this cluster transmission queue.

By specifying asterisks, "\*", in **ClusterChannelName**, you can associate a transmission queue with a set of cluster-sender channels. The asterisks can be at the beginning, end, or any number of places in the middle of the channel name string. **ClusterChannelName** is limited to a length of 20 characters: `MQ_CHANNEL_NAME_LENGTH`.

**IMGRCOVQ**

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*YES**

These queue objects are recoverable.


**\*NO**

The [“RCDMQMIMG \(Record MQ Object Image\)” on page 1857](#) and [“RCRMQMOBJ \(Re-create MQ Object\)” on page 1859](#) commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

**\*QMGR**

If you specify \*QMGR, and the **IMGRCOVQ** attribute for the queue manager specifies \*YES, these queue objects are recoverable.

If you specify \*QMGR and the **IMGRCOVQ** attribute for the queue manager specifies \*NO, the [“RCDMQMIMG \(Record MQ Object Image\)” on page 1857](#) and [“RCRMQMOBJ \(Re-create MQ Object\)” on page 1859](#) commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.


**CPYMQMSUB (Copy MQ Subscription)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Copy MQ Subscription (CPYMQMSUB) command creates an MQ subscription of the same type and, for attributes not specified in the command, with the same attribute values as an existing subscription.

**Parameters**

Keyword	Description	Choices	Notes
<code>FROMSUBID</code>	From subscription identifier	<i>Character value, *SAME</i>	Optional, Key, Positional 3
<code>FROMSUB</code>	From subscription	<i>Character value, *SAME</i>	Optional, Key, Positional 2

Table 239. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>TOSUB</u>	To subscription	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Key, Positional 4
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 5
<u>TOPICSTR</u>	Topic string	Character value, *NONE, <b>*SAME</b>	Optional, Positional 6
<u>TOPICOBJ</u>	Topic object	Character value, *NONE, <b>*SAME</b>	Optional, Positional 7
<u>DEST</u>	Destination	Character value, *NONE, <b>*SAME</b>	Optional, Positional 8
<u>DESTMQM</u>	Destination Queue Manager	Character value, *NONE, <b>*SAME</b>	Optional, Positional 9
<u>DESTCRRLLID</u>	Destination Correlation Id	Character value, *NONE, <b>*SAME</b>	Optional, Positional 10
<u>PUBACCT</u>	Publish Accounting Token	Character value, *NONE, <b>*SAME</b>	Optional, Positional 11
<u>PUBAPPID</u>	Publish Application Id	Character value, *NONE, <b>*SAME</b>	Optional, Positional 12
<u>SUBUSER</u>	Subscription User Id	Character value, *CURRENT, <b>*SAME</b>	Optional, Positional 13
<u>USERDATA</u>	Subscription User Data	Character value, *NONE, <b>*SAME</b>	Optional, Positional 14
<u>SELECTOR</u>	Selector String	Character value, *NONE, <b>*SAME</b>	Optional, Positional 15
<u>PSPROP</u>	PubSub Property	<b>*SAME</b> , *NONE, *COMPAT, *RFH2, *MSGPROP	Optional, Positional 16
<u>DESTCLASS</u>	Destination Class	<b>*SAME</b> , *MANAGED, *PROVIDED	Optional, Positional 17
<u>SUBSCOPE</u>	Subscription Scope	<b>*SAME</b> , *ALL, *QMGR	Optional, Positional 18
<u>VARUSER</u>	Variable User	<b>*SAME</b> , *ANY, *FIXED	Optional, Positional 19
<u>REQONLY</u>	Request Publications	<b>*SAME</b> , *YES, *NO	Optional, Positional 20
<u>PUBPTY</u>	Publish Priority	0-9, <b>*SAME</b> , *AS PUB, *ASQDEF	Optional, Positional 21
<u>WSHEMA</u>	Wildcard Schema	<b>*SAME</b> , *CHAR, *TOPIC	Optional, Positional 22
<u>EXPIRY</u>	Expiry Time	0-999999999, <b>*SAME</b> , *UNLIMITED	Optional, Positional 23



## From subscription identifier (FROMSUBID)

Specifies the subscription identifier of the existing subscription to provide values for the attributes not specified in this command.

The possible values are:

### from-subscription-identifier

Specify the 48 character hexadecimal string representing the 24 byte subscription identifier.

## From subscription (FROMSUB)

Specifies the name of the existing subscription to provide values for the attributes not specified in this command.

The possible values are:

### from-subscription-name

Specify a maximum of 256 bytes for the subscription name.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

## To subscription (TOSUB)

The name of the new subscription to be created.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

If a subscription with this name already exists, REPLACE(\*YES) must be specified.

The possible values are:

### to-subscription-name

Specify a maximum of 256 bytes for name of the MQ subscription being created.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

## Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

### \*DFT

Use the default Queue Manager.

### queue-manager-name

The name of a Queue Manager.

## Replace (REPLACE)

Specifies whether the new subscription should replace an existing subscription with the same name.

The possible values are:

### \*NO

This subscription does not replace any existing subscription with the same name or subscription identifier. The command fails if the subscription already exists.

### \*YES

Replace the existing subscription. If there is no subscription with the same name or subscription identifier, a new subscription is created.

## Topic string (TOPICSTR)

Specifies the topic string associated with this subscription.

The possible values are:

**topic-string**

Specify a maximum of 256 bytes for the topic string.

**Note:** Topic strings of greater than 256 bytes can be specified using MQSC.

**Topic object (TOPICOBJ)**

Specifies the topic object associated with this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**topic-object**

Specify the name of the topic object.

**Destination (DEST)**

Specifies the destination queue for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**destination-queue**

Specify the name of the destination queue.

**Destination Queue Manager (DESTMQM)**

Specifies the destination queue manager for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No destination queue manager is specified.

**destination-queue**

Specify the name of the destination queue manager.

**Destination Correlation Id (DESTCRLID)**

Specifies the correlation identifier for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Messages are placed on the destination with a correlation identifier of MQCI\_NONE.

**correlation-identifier**

Specify the 48 character hexadecimal string representing the 24 byte correlation identifier.

**Publish Accounting Token (PUBACCT)**

Specifies the accounting token for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Messages are placed on the destination with an accounting token of MQACT\_NONE.

**publish-accounting-token**

Specify the 64 character hexadecimal string representing the 32 byte publish accounting token.

**Publish Application Id (PUBAPPID)**

Specifies the publish application identity for messages published to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No publish application identifier is specified.

**publish-application-identifier**

Specify the publish application identifier.

**Subscription User Id (SUBUSER)**

Specifies the user profile that owns this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*CURRENT**

The current user profile is the owner of the new subscription.

**user-profile**

Specify the user profile.

**Subscription User Data (USERDATA)**

Specifies the user data associated with the subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No user data is specified.

**user-data**

Specify a maximum of 256 bytes for user data.

**Note:** User data of greater than 256 bytes can be specified using MQSC.

**Selector String (SELECTOR)**

Specifies the SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

No selection string is specified.

**selection-string**

Specify a maximum of 256 bytes for selection string.

**Note:** Selection strings of greater than 256 bytes can be specified using MQSC.

**PubSub Property (PSPROP)**

Specifies the manner in which publish / subscribe related message properties are added to messages sent to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*NONE**

Publish / subscribe properties are not added to the message.

**\*COMPAT**

Publish / subscribe properties are added to the message to maintain compatibility with IBM MQ V6.0 Publish / Subscribe.

**\*RFH2**

Publish / subscribe properties are added to the message within an RFH 2 header.

**\*MSGPROP**

Publish / subscribe properties are added as message properties.

**Destination Class (DESTCLASS)**

Specifies whether this is a managed subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*MANAGED**

The destination is managed.

**\*PROVIDED**

The destination is a queue.

**Subscription Scope (SUBSCOPE)**

Specifies whether this subscription should be forwarded (as a proxy subscription) to other brokers, so that the subscriber will receive messages published at those other brokers.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ALL**

The subscription will be forwarded to all queue managers directly connected via a publish / subscribe collective or hierarchy.

**\*QMGR**

The subscription will only forward messages published on the topic within this queue manager.

**Variable User (VARUSER)**

Specifies whether user profiles other than the creator of the subscription can connect to it (subject to topic and destination authority checks).

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ANY**

Any user profiles can connect to the subscription.

**\*FIXED**

Only the user profile that created the subscription can connect to it.

**Request Publications (REQONLY)**

Specifies whether the subscriber will poll for updates via MQSUBRQ API, or whether all publications are delivered to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*YES**

Publications are only delivered to this subscription in response to an MQSUBRQ API.

**\*NO**

All publications on the topic are delivered to this subscription.

**Publish Priority (PUBPTY)**

Specifies the priority of the message sent to this subscription.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*AS PUB**

The priority of the message sent to this subscription is taken from that supplied in the published message.

**\*ASQDEF**

The priority of the message sent to this subscription is taken from the default priority of the queue defined as the destination.

**priority-value**

Specify a priority ranging from 0 through 9.

**Wildcard Schema (WSHEMA)**

Specifies the schema to be used when interpreting wildcard characters in the topic string.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*TOPIC**

Wildcard characters represent portions of the topic hierarchy.

**\*CHAR**

Wildcard characters represent portions of strings.

**Expiry Time (EXPIRY)**

Specifies the expiry time of the subscription. After a subscription's expiry time has elapsed, it becomes eligible to be discarded by the queue manager and will receive no further publications.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*UNLIMITED**

The subscription does not expire.

**expiry-time**

Specify an expiry time in tenths of a second ranging from 0 through 999999999.


**CPYMQMSVC (Copy MQ Service)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Copy MQ Service (CPYMQMSVC) command creates an MQ service definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing service definition.

**Parameters**

*Table 240. Command parameters*

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>FROMSVC</u>	From Service	Character value	Required, Key, Positional 1
<u>TOSVC</u>	To Service	Character value	Required, Key, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 3
<u>REPLACE</u>	Replace	<b>*NO</b> , <b>*YES</b>	Optional, Positional 4
<u>TEXT</u>	Text 'description'	<i>Character value</i> , <b>*BLANK</b> , <b>*SAME</b>	Optional, Positional 5
<u>STRCMD</u>	Start program	Single values: <b>*SAME</b> , <b>*NONE</b> Other values: <i>Qualified object name</i>	Optional, Positional 6
	Qualifier 1: Start program	Name	
	Qualifier 2: Library	Name	
<u>STRARG</u>	Start program arguments	<i>Character value</i> , <b>*BLANK</b> , <b>*SAME</b>	Optional, Positional 7
<u>ENDCMD</u>	End program	Single values: <b>*SAME</b> , <b>*NONE</b> Other values: <i>Qualified object name</i>	Optional, Positional 8
	Qualifier 1: End program	Name	
	Qualifier 2: Library	Name	
<u>ENDARG</u>	End program arguments	<i>Character value</i> , <b>*BLANK</b> , <b>*SAME</b>	Optional, Positional 9
<u>STDOUT</u>	Standard output	<i>Character value</i> , <b>*BLANK</b> , <b>*SAME</b>	Optional, Positional 10

Table 240. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>STDERR</u>	Standard error	Character value, *BLANK, *SAME	Optional, Positional 11
<u>TYPE</u>	Service type	*SAME, *CMD, *SVR	Optional, Positional 12
<u>CONTROL</u>	Service control	*SAME, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 13

### From Service (FROMSVC)

Specifies the name of the existing service definition to provide values for the attributes not specified in this command.

The possible values are:

#### from-service-name

Specify the name of the source service.

### To Service (TOSVC)

The name of the new service definition to be created. The name can contain a maximum of 48 characters.

If a service definition with this name already exists, REPLACE(\*YES) must be specified.

The possible values are:

#### to-service-name

Specify the name of the service being created.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.

### Replace (REPLACE)

Specifies whether the new service definition should replace an existing service definition with the same name.

The possible values are:

#### \*NO

This definition does not replace any existing service definition with the same name. The command fails if the named service definition already exists.

#### \*YES

Replace the existing service definition. If there is no definition with the same name, a new definition is created.

### Text 'description' (TEXT)

Specifies text that briefly describes the service definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

### **Start program (STRCMD)**

The name of the program to run.

The possible values are:

**\*SAME**

The attribute is unchanged.

**start-command**

The name of the start command executable.

### **Start program arguments (STRARG)**

The arguments passed to the program at startup.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

No arguments are passed to the start command.

**start-command-arguments**

The arguments passed to the start command.

### **End program (ENDCMD)**

The name of the executable to run when the service is requested to stop.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

No end command is executed.

**end-command**

The name of the end command executable.

### **End program arguments (ENDARG)**

The arguments passed to the end program when the service is requested to stop.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

No arguments are passed to the end command.



**end-command-arguments**

The arguments passed to the end command.

**Standard output (STDOUT)**

The path to a file to which the standard output of the service program is redirected.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The standard output is discarded.

**stdout-path**

The standard output path.

**Standard error (STDERR)**

The path to a file to which the standard error of the service program is redirected.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The standard error is discarded.

**stderr-path**

The standard error path.

**Service type (TYPE)**

Mode in which to run service.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*CMD**

When started the command is executed but no status is collected or displayed.

**\*SVR**

The status of the executable started will be monitored and displayed.

**Service control (CONTROL)**

Whether the service should be started automatically at queue manager start.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*MANUAL**

The service is automatically started or stopped.

**\*QMGR**

The service is started and stopped as the queue manager is started and stopped.

**\*STARTONLY**

The service is started as the queue manager is started, but will not be requested to stop when the queue manager is stopped.

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Copy MQ Topic (CPYMQMTOPTOP) command creates an MQ topic object of the same type and, for attributes not specified in the command, with the same attribute values as an existing topic object.

**Parameters**

<i>Table 241. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>FROMTOP</u>	From topic	Character value	Required, Key, Positional 1
<u>TOTOP</u>	To topic	Character value	Required, Key, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 3
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 4
<u>TEXT</u>	Text 'description'	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 5
<u>TOPICSTR</u>	Topic string	<i>Character value</i> , *BLANK, <b>*SAME</b>	Optional, Positional 6
<u>DURSUB</u>	Durable subscriptions	<b>*SAME</b> , *ASPARENT, *YES, *NO	Optional, Positional 7
<u>MGDDURMDL</u>	Durable model queue	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 8
<u>MGDNDURMDL</u>	Non-durable model queue	<i>Character value</i> , *NONE, <b>*SAME</b>	Optional, Positional 9
<u>PUBENBL</u>	Publish	<b>*SAME</b> , *ASPARENT, *YES, *NO	Optional, Positional 10
<u>SUBENBL</u>	Subscribe	<b>*SAME</b> , *ASPARENT, *YES, *NO	Optional, Positional 11
<u>DFTPTY</u>	Default message priority	0-9, <b>*SAME</b> , *ASPARENT	Optional, Positional 12
<u>DFTMSGPST</u>	Default message persistence	<b>*SAME</b> , *ASPARENT, *YES, *NO	Optional, Positional 13
<u>DFTPUTRESP</u>	Default Put Response	<b>*SAME</b> , *ASPARENT, *SYNC, *ASYNCR	Optional, Positional 14
<u>WILDCARD</u>	Wildcard behavior	<b>*SAME</b> , *PASSTHRU, *BLOCK	Optional, Positional 15
<u>PMSGDLV</u>	Persistent message delivery	<b>*SAME</b> , *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 16
<u>NPMSGDLV</u>	Non-persistent message deliver	<b>*SAME</b> , *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 17

Table 241. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>CUSTOM</u>	Custom attribute	Character value, *BLANK, *SAME	Optional, Positional 18

### From topic (FROMTOP)

Specifies the name of the existing topic object to provide values for the attributes not specified in this command.

The possible values are:

#### from-topic-name

Specify the name of the source MQ topic.

### To topic (TOTOP)

The name of the new topic object to be created. The name can contain a maximum of 48 characters.

If a topic object with this name already exists, REPLACE(\*YES) must be specified.

The possible values are:

#### to-topic-name

Specify the name of the MQ topic being created.

### Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

#### \*DFT

Use the default Queue Manager.

#### queue-manager-name

The name of a Queue Manager.

### Replace (REPLACE)

Specifies whether the new topic object should replace an existing topic object with the same name.

The possible values are:

#### \*NO

This object does not replace any existing topic object with the same name. The command fails if the named topic object already exists.

#### \*YES

Replace the existing topic object. If there is no object with the same name, a new object is created.

### Text 'description' (TEXT)

Specifies text that briefly describes the topic object.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

#### \*SAME

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**Topic string (TOPICSTR)**

Specifies the topic string represented by this topic object definition.

The possible values are:

**topic-string**

Specify a maximum of 256 bytes for the topic string.

**Note:** Topic strings of greater than 256 bytes can be specified using MQSC.

**Durable subscriptions (DURSUB)**

Specifies whether applications are permitted to make durable subscriptions on this topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

Whether durable subscriptions can be made on this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*YES**

Durable subscriptions can be made on this topic.

**\*NO**

Durable subscriptions cannot be made on this topic.

**Durable model queue (MGDDURMDL)**

Specifies the name of the model queue to be used for durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

**\*SAME**

The attribute is unchanged.

**durable-model-queue**

Specify the name of the model queue.

**Non-durable model queue (MGDNDURMDL)**

Specifies the name of the model queue to be used for non-durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

**\*SAME**

The attribute is unchanged.

**non-durable-model-queue**

Specify the name of the model queue.

**Publish (PUBENBL)**

Specifies whether messages can be published to the topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

Whether messages can be published to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*YES**

Messages can be published to the topic.

**\*NO**

Messages cannot be published to the topic.

**Subscribe (SUBENBL)**

Specifies whether applications are to be permitted to subscribe to this topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

Whether applications can subscribe to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*YES**

Subscriptions can be made to this topic.

**\*NO**

Applications cannot subscribe to this topic.

**Default message priority (DFTPTY)**

Specifies the default priority of messages published to the topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

The default priority is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**priority-value**

Specify a value ranging from 0 through 9.

**Default message persistence (DFTMSGPST)**

Specifies the message persistence to be used when applications specify the MQPER\_PERSISTENCE\_AS\_TOPIC\_DEF option.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

The default persistence is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*YES**

Messages on this queue survive a restart of the queue manager.

**\*NO**

Messages on this queue are lost across a restart of the queue manager.

## Default Put Response (DFTPURRESP)

Specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO\_RESPONSE\_AS\_Q\_DEF option.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*ASPARENT**

The default response type is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **\*SYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application.

### **\*ASYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application. An improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

## Wildcard behavior (WILDCARD)

Specifies the behavior of wildcard subscriptions with respect to this topic.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*PASSTHRU**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will receive publications made to this topic and to topic strings more specific than this topic.

### **\*BLOCK**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will not receive publications made to this topic or to topic strings more specific than this topic.

## Persistent message delivery (PMSGDLV)

Specifies the delivery mechanism for persistent messages published to this topic.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **\*ALL**

Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

### **\*ALLDUR**

Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

**\*ALLAVAIL**

Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**Non-persistent message delivery (NPMSGDLV)**

Specifies the delivery mechanism for non-persistent messages published to this topic.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*ALL**

Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**\*ALLDUR**

Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

**\*ALLAVAIL**

Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**Custom attribute (CUSTOM)**

This attribute is reserved for the configuration of new features before separate attributes have been introduced. This description will be updated when features using this attribute are introduced. At the moment there are no meaningful values for *CUSTOM*, so leave it empty.

The possible values are:

**\*SAME**

The attribute is unchanged.

**\*BLANK**

The text is set to a blank string.

**custom**

Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs must have the form NAME (VALUE) and be specified in uppercase. Single quotes must be escaped with another single quote.

 **CRTMQM (Create Message Queue Manager)****Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Create Message Queue Manager (CRTMQM) command creates a local queue manager that can be started with the Start Message Queue Manager (STRMQM) command.

## Parameters

Table 242. Command parameters			
Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1
<u>TEXT</u>	Text 'description'	<i>Character value</i> , <b>*BLANK</b>	Optional, Positional 2
<u>TRGITV</u>	Trigger interval	0-999999999, <b>999999999</b>	Optional, Positional 3
<u>UDLMSGQ</u>	Undelivered message queue	<i>Character value</i> , <b>*NONE</b>	Optional, Positional 4
<u>DFTTMQ</u>	Default transmission queue	<i>Character value</i> , <b>*NONE</b>	Optional, Positional 5
<u>MAXHDL</u>	Maximum handle limit	0-999999999, <b>256</b>	Optional, Positional 6
<u>MAXUMSG</u>	Maximum uncommitted messages	1-999999999, <b>10000</b>	Optional, Positional 7
<u>DFTQMGR</u>	Default Queue Manager	<b>*YES</b> , <b>*NO</b>	Optional, Positional 8
<u>MQMLIB</u>	Queue Manager Library	<i>Name</i> , <b>*AUTO</b>	Optional, Positional 9
<u>MQMDIRP</u>	Data Directory Prefix	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 10
<u>ASP</u>	ASP Number	1-32, <b>*SYSTEM</b> , <b>*ASPDEV</b>	Optional, Positional 11
<u>ASPDEV</u>	ASP device	<i>Character value</i> , <b>*ASP</b>	Optional, Positional 12
<u>THRESHOLD</u>	Journal receiver threshold	100000-1000000000, <b>*DFT</b> , <b>*MIN</b> , <b>*MAX</b>	Optional, Positional 13
<u>JRNBUFSIZ</u>	Journal buffer size	32000-15761440, <b>*DFT</b>	Optional, Positional 14

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

#### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### Text 'description' (TEXT)

Specifies text that briefly describes the queue manager definition.

The possible values are:

#### **\*BLANK**

No text is specified.

#### description

Specify no more than 64 characters enclosed in apostrophes.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).



## Trigger interval (TRGITV)

Specifies the trigger time interval, expressed in milliseconds, for use with queues that have TRGTYPE(\*FIRST) specified.

When the arrival of a message on a queue causes a trigger message to be put on the initiation queue, then any message that arrives on the same queue within the specified interval does not cause another trigger message to be put on the initiation queue.

The possible values are:

### **999999999**

The trigger time interval is 999999999 milliseconds.

### **interval-value**

Specify a value in milliseconds, in the range 0 through 999999999.

## Undelivered message queue (UDLMSGQ)

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination.

The possible values are:

### **\*NONE**

There is no undelivered-message queue. The attribute is set to a blank string.

### **undelivered-message-queue-name**

Specify the name of a local queue that is to be used as the undelivered-message queue.

## Default transmission queue (DFTTMQ)

Specifies the name of the local transmission queue that is to be used as the default transmission queue. Messages transmitted to a remote queue manager are put on the default transmission queue if there is no transmission queue defined for their destination.

The possible values are:

### **\*NONE**

There is no default transmission queue. The attribute is set to a blank string.

### **default-transmission-queue-name**

Specify the name of a local transmission queue that is to be used as the default transmission queue.

## Maximum handle limit (MAXHDL)

Specifies the maximum number of handles that any one job can have open at the same time.

The possible values are:

### **256**

The default number of open handles is 256.

### **maximum-handle-limit**

Specify a value in the range 0 through 999999999.

## Maximum uncommitted messages (MAXUMSG)

Specifies the maximum number of uncommitted messages. That is:

- The number of messages that can be retrieved, plus
- The number of messages that can be put on a queue, plus
- Any trigger messages generated within this unit of work,

under any one syncpoint. This limit does not apply to messages that are retrieved or put outside syncpoint.

The possible values are:

**10000**

The default value is 10000 uncommitted messages.

**maximum-uncommitted-messages**

Specify a value in the range 1 through 999999999.

### **Default Queue Manager (DFTQMGR)**

Specifies whether the queue manager being created is the default queue manager.

The possible values are:

**\*NO**

The queue manager is not to be the default queue manager.

**\*YES**

The queue manager is to be the default queue manager.

### **Queue Manager Library (MQMLIB)**

Specifies the library to be used by the queue manager.

The possible values are:

**\*AUTO**

The library to be used by the queue manager is chosen automatically.

**library name**

Specify the library to be used by the queue manager.

### **Data Directory Prefix (MQMDIRP)**

Specifies the data directory prefix to be used by the queue manager. The queue manager creates a directory here to store its data files, principally message data residing on queues.

The possible values are:

**\*DFT**

The default data directory prefix is '/QIBM/UserData/mqm'.

**directory-prefix**

Specify the data directory prefix to be used by the queue manager. This directory prefix may be located in a filesystem either in a local disk pool or in a networked filesystem e.g. NFS.

The queue manager directory can be placed into an independent auxiliary storage pool by setting the data directory prefix accordingly. For example specifying MQMDIRP('/MYASPDEV/QIBM/UserData/mqm/qmgrs') would store queue manager data in the MYASPDEV device.

The queue manager library, journals and journal receivers can be placed into an independent auxiliary storage pool by setting the ASP and ASPDEV parameters.

Independent auxiliary storage pools can be switched between systems to increase the availability of a queue manager. Refer to the IBM MQ documentation on configuring a queue manager for high availability.

### **ASP Number (ASP)**

Specifies the auxiliary storage pool from which the system allocates storage for the queue manager library, journal and journal receivers.

Note that the auxiliary storage pool identified in this parameter will not be used for the queue manager data files which are located in the integrated file system (IFS). To allocate queue manager data files in a specific auxiliary storage pool refer to the MQMDIRP parameter.

The possible values are:

**\*SYSTEM**

The system auxiliary storage pool (ASP 1) provides the storage for the queue manager library, journal and journal receivers.

**\*ASPDEV**

Storage for the queue manager library, journal and journal receivers is allocated from the primary or secondary ASP specified for the ASPDEV parameter.

**auxiliary-storage-pool-number**

Specify a value in the range 1 through 32 to specify the number of the system or basic user ASP to provide storage for the queue manager library, journal and journal receivers.

Independent auxiliary storage pools can be switched between systems to increase the availability of a queue manager. Refer to the IBM MQ documentation on configuring a queue manager for high availability.

### **ASP device (ASPDEV)**

Specifies the auxiliary storage pool (ASP) device name where storage is allocated for the queue manager library, journal and journal receivers.

Note that the auxiliary storage pool device name identified in this parameter will not be used for the queue manager data files which are located in the integrated file system (IFS). To allocate queue manager data files in a specific auxiliary storage pool refer to the MQMDIRP parameter.

The possible values are:

**\*ASP**

The storage for the queue manager library, journal and journal receivers is allocated from the system or basic user ASP specified for the ASP parameter.

**device-name**

Specify the name of a primary or secondary ASP device. The storage for the queue manager library, journal and journal receivers is allocated from the primary or secondary ASP. The primary or secondary ASP must have already been activated (by varying on the ASP device) and have a status of 'Available'.

Independent auxiliary storage pools can be switched between systems to increase the availability of a queue manager. Refer to the IBM MQ documentation on configuring a queue manager for high availability.

### **Journal receiver threshold (THRESHOLD)**

Specifies the threshold in kilobytes for the queue managers journal receivers.

The possible values are:

**\*DFT**

Use the default threshold of 100000 KB.

**threshold-value**

Specify a value in the range 100000 through 1000000000 in kilobytes (KB) of storage. Each 1000 KB specifies 1024000 bytes of storage space. When the size of the space for the journal receiver is larger than the size specified by this value, a message is sent to the identified message queue if appropriate, and journaling continues.

### **Journal buffer size (JRNBUFSIZ)**

Specifies the journal buffer size in bytes

The possible values are:

**\*DFT**

Use the default journal buffer size of 32000 bytes.

## journal-buffer-size

Specify a value in bytes, in the range 32000 through 15761440.

## IBM i CRTMQMAUTI (Create MQ AuthInfo object)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Create MQ AuthInfo object (CRTMQMAUTI) command creates a new authentication information object, specifying those attributes that are different from the system default.

## Parameters

Keyword	Description	Choices	Notes
<u>AINAME</u>	AuthInfo name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Required, Key, Positional 2
<u>AUTHTYPE</u>	AuthInfo type	*CRLLDAP, *OCSP, *IDPWOS, *IDPWLDAP	Required, Key, Positional 3
<u>CONNNAME</u>	Connection name	Character value, *SYSDFTAI	Optional, Positional 4
<u>REPLACE</u>	Replace	*NO, *YES	Optional, Positional 5
<u>TEXT</u>	Text 'description'	Character value, *SYSDFTAI, *NONE	Optional, Positional 6
<u>USERNAME</u>	User name	Character value, *SYSDFTAI, *NONE	Optional, Positional 7
<u>PASSWORD</u>	User password	Character value, *SYSDFTAI, *NONE	Optional, Positional 8
<u>OCSPURL</u>	OCSP Responder URL	Character value, *SAME	Optional, Positional 9
<u>CHCKCLNT</u>	Authentication checks required	*ASQMGR, *REQUIRED, *REQADM	Optional, Positional 10
<u>CHCKLOCL</u>	Authentication checks required	*NONE, *OPTIONAL, *REQUIRED, *REQADM	Optional, Positional 11
<u>FAILDELAY</u>	Failure delay	Integer value	Optional, Positional 12
<u>BASEDNU</u>	Base user DN	Character value, *SAME	Optional, Positional 13
<u>ADOPTCTX</u>	Context adoption	Integer value	Optional, Positional 14
<u>CLASSUSR</u>	LDAP object class	Character value, *SAME	Optional, Positional 15
<u>SHORTUSR</u>	Short user name	Character value, *SAME	Optional, Positional 16
<u>USRFIELD</u>	User field	Character value, *SAME	Optional, Positional 17
<u>SECCOMM</u>	LDAP communications	Character value, *SAME	Optional, Positional 18

Table 243. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>AUTHORMD</u>	Authorization method	Character value, <b>*OS</b> , *SEARCHGRP, *SEARCHUSR, *SRCHGRPSN	Optional, Positional 19
<u>BASEDNG</u>	Base DN for groups	Character value, <b>*SAME</b>	Optional, Positional 20
<u>CLASSGRP</u>	Object class for group	Character value, <b>*SAME</b>	Optional, Positional 21
<u>FINDGRP</u>	Attribute to find group membership	Character value, <b>*SAME</b>	Optional, Positional 22
<u>GRPFIELD</u>	Simple name for group	Character value, <b>*SAME</b>	Optional, Positional 23
<u>NESTGRP</u>	Group nesting	<b>*NO</b> *YES	Optional, Positional 24
<u>AUTHENMD</u>	Authentication method	<b>*OS</b> Cannot be changed	Optional, Positional 25

### AuthInfo name (AINAME)

The name of the new authentication information object to create.

The possible values are:

#### authentication-information-name

Specify the name of the authentication information object. The maximum string length is 48 characters.

### Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of an existing message queue manager. The maximum string length is 48 characters.

### Adopt context (ADOPTCTX)

Whether to use the presented credentials as the context for this application. This means that they are used for authorization checks, shown on administrative displays, and appear in messages.

#### YES

The user ID presented in the MQCSP structure, which has been successfully validated by password, is adopted as the context to use for this application. Therefore, this user ID will be the credentials checked for authorization to use IBM MQ resources.

If the user ID presented is an LDAP user ID, and authorization checks are done using operating system user IDs, the SHORTUSR associated with the user entry in LDAP will be adopted as the credentials for authorization checks to be done against.

#### NO

Authentication will be performed on the user ID and password presented in the MQCSP structure, but then the credentials will not be adopted for further use. Authorization will be performed using the user ID the application is running under.

This attribute is only valid for an **AUTHTYPE** of \*IDPWOS and \*IDPWLDP.

## Authentication method (AUTHENMD)

The authentication method used for this application.

### **\*OS**

Use operating system groups to determine permissions associated with a user.

You can use only **\*OS** to set the authentication method.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWOS*.

## Authorization method (AUTHORMD)

The authorization method used for this application.

### **\*OS**

Use operating system groups to determine permissions associated with a user.

This is how IBM MQ has previously worked, and is the default value.

### **\*SEARCHGRP**

A group entry in the LDAP repository contains an attribute listing the Distinguished Name of all the users belonging to that group. Membership is indicated by the attribute defined in FINDGRP. This value is typically *member* or *uniqueMember*.

### **\*SEARCHUSR**

A user entry in the LDAP repository contains an attribute listing the Distinguished Name of all the groups to which the specified user belongs. The attribute to query is defined by the FINDGRP value, typically *memberOf*.

### **\*SRCHGRPSN**

A group entry in the LDAP repository contains an attribute listing the short user name of all the users belonging to that group. The attribute in the user record that contains the short user name is specified by SHORTUSR.

Membership is indicated by the attribute defined in FINDGRP. This value is typically *memberUid*.

**Note:** This authorization method should only be used if all user short names are distinct.

Many LDAP servers use an attribute of the group object to determine group membership and you should, therefore, set this value to *SEARCHGRP*.

Microsoft Active Directory typically stores group memberships as a user attribute. The IBM Tivoli Directory Server supports both methods.

In general, retrieving memberships through a user attribute will be faster than searching for groups that list the user as a member.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

## AuthInfo type (AUTHTYPE)

The type of the authentication information object. There is no default value

The possible values are:

### **\*CRLLDAP**

The type of the authentication information object is CRLLDAP.

### **\*OCSP**

The type of the authentication information objects is OCSPURL.

### **\*IDPWOS**

Connection authentication user ID and password checking is done using the operating system.

### **\*IDPWLDAP**

Connection authentication user ID and password checking is done using an LDAP server.

## Base DN for groups (BASEDNG)

In order to be able to find group names, this parameter must be set with the base DN to search for groups in the LDAP server.

This attribute is valid only for **AUTHTYPE** of *\*IDPWLDAP*.

## Base user DN (BASEDNU)

In order to be able to find the short user name attribute (see [SHORTUSR](#)) this parameter must be set with the base DN to search for users within the LDAP server.

This attribute is valid only for **AUTHTYPE** of *\*IDPWLDAP*.

## Check client (CHCKCLNT)

Whether connection authentication checks are required by all locally bound connections, or only checked when a user ID and password are provided in the MQCSP structure.

These attributes are valid only for an **AUTHTYPE** of *\*IDPWOS* or *\*IDPWLDAP*. The possible values are:

### **\*ASQMGR**

In order for the connection to be allowed in, it must meet the connection authentication requirements defined on the queue manager. If the CONNAUTH field provides an authentication information object, and the value of CHCKCLNT is *\*REQUIRED*, the connection will not be successful unless a valid user ID and password are supplied. If the CONNAUTH field does not provide an authentication information object, or the value of CHCKCLNT is not *\*REQUIRED*, then the user ID and password are not required.

### **\*REQUIRED**

Requires that all applications provide a valid user ID and password.

### **\*REQDADM**

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the *\*OPTIONAL* setting.

## Check local (CHCKLOCL)

Whether connection authentication checks are required by all locally bound connections, or only checked when a user ID and password are provided in the MQCSP structure.

These attributes are valid only for an **AUTHTYPE** of *\*IDPWOS* or *\*IDPWLDAP*. The possible values are:

### **\*NONE**

Switches off checking.

### **\*OPTIONAL**

Ensures that if a user ID and password are provided by an application, they are a valid pair, but that it is not mandatory to provide them. This option might be useful during migration, for example.

### **\*REQUIRED**

Requires that all applications provide a valid user ID and password.

### **\*REQDADM**

Privileged users must supply a valid user ID and password, but non-privileged users are treated as with the *\*OPTIONAL* setting.

## Class group (CLASSGRP)

The LDAP object class used for group records in the LDAP repository.

If the value is blank, **groupOfNames** is used.

Other commonly used values include *groupOfUniqueNames* or *group*.

This attribute is valid only for **AUTHTYPE** of *\*IDPWLDAP*.

## Class user (CLASSUSR)

The LDAP object class used for user records in the LDAP repository.

If blank, the value defaults to *inetOrgPerson*, which is generally the value needed.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

## Connection name (CONNAME)

The DNS name or IP address of the host on which the LDAP server is running, together with an optional port number. The default port number is 389. No default is provided for the DNS name or IP address.

This field is only valid for *\*CRLLDAP* or *\*IDPWLDAP* authentication information objects, when it is required.

When used with *IDPWLDAP* authentication information objects, this can be a comma separated list of connection names.

The possible values are:

### **\*SYSDFTAI**

The connection name is set to the system default value in SYSTEM.DEFAULT.AUTHINFO.CRLLDAP.

### **connection-name**

Specify the fully qualified DNS name or IP address of the host together with an optional port number. The maximum string length is 264 characters.

## Failure delay (FAILDELAY)

When a user ID and password are provided for connection authentication, and the authentication fails due to the user ID or password being incorrect, this is the delay, in seconds, before the failure is returned to the application.

This can aid in avoiding busy loops from an application that simply retries, continuously, after receiving a failure.

The value must be in the range 0 - 60 seconds. The default value is 1.

This attribute is only valid for an **AUTHTYPE** of *\*IDPWOS* and *\*IDPWLDAP*.

## Group membership attribute (FINDGRP)

Name of the attribute used within an LDAP entry to determine group membership.

When AUTHORMD = *\*SEARCHGRP*, this attribute is typically set to *member* or *uniqueMember*.

When AUTHORMD = *\*SEARCHUSR*, this attribute is typically set to *memberOf*.

When AUTHORMD = *\*SRCHGRPSN*, this attribute is typically set to *memberUid*.

When left blank, if:

- AUTHORMD = *\*SEARCHGRP*, this attribute defaults to *memberOf*
- AUTHORMD = *\*SEARCHUSR*, this attribute defaults to *member*
- AUTHORMD = *\*SRCHGRPSN*, this attribute defaults to *memberUid*

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

## Simple name for group (GRPFIELD)

If the value is blank, commands like setmqaut must use a qualified name for the group. The value can either be a full DN, or a single attribute.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.



## Group nesting (NESTGRP)

The possible values are:

### **\*NO**

Only the initially discovered groups are considered for authorization.

### **\*YES**

The group list is searched recursively to enumerate all the groups to which a user belongs.

The group's Distinguished Name is used when searching the group list recursively, regardless of the authorization method selected in `AUTHORMD`.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

## OCSP Responder URL (OCSPURL)

The URL of the OCSP Responder used to check for certificate revocation. This must be an HTTP URL containing the host name and port number of the OCSP Responder. If the OCSP Responder is using port 80, which is the default for HTTP, then the port number may be omitted.

This field is only valid for OCSP authentication information objects.

The possible values are:

### **\*SYSDFTAI**

The OCSP Responder URL is set to the system default value in `SYSTEM.DEFAULT.AUTHINFO.OCSP`.

### **OCSP-Responder-URL**

The OCSP Responder URL. The maximum string length is 256 characters.

## Replace (REPLACE)

If an authentication information object with the same name already exists, this specifies whether it is replaced.

The possible values are:

### **\*NO**

This definition does not replace any existing authentication information object with the same name. The command fails if the named authentication information object already exists.

### **\*YES**

Replace an existing authentication information object. A new object is created if the named authentication information object does not exist.

## Secure comms (SECCOMM)

Whether connectivity to the LDAP server should be done securely using TLS

### **YES**

Connectivity to the LDAP server is made securely using TLS.

The certificate used is the default certificate for the queue manager, named in `CERTLABL` on the queue manager object, or if that is blank, the one described in [Digital certificate labels, understanding the requirements](#).

The certificate is located in the key repository specified in `SSLKEYR` on the queue manager object. A cipherspec will be negotiated that is supported by both IBM MQ and the LDAP server.

If the queue manager is configured to use `SSLFIPS(YES)` or `SUITEB` cipher specs, then this is taken account of in the connection to the LDAP server as well.

### **ANON**

Connectivity to the LDAP server is made securely using TLS just as for `SECCOMM(YES)` with one difference.

No certificate is sent to the LDAP server; the connection will be made anonymously. To use this setting, ensure that the key repository specified in `SSLKEYR`, on the queue manager object, does not contain a certificate marked as the default.

## **NO**

Connectivity to the LDAP server does not use TLS.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*

## **Short user (SHORTUSR)**

A field in the user record to be used as a short user name in IBM MQ.

This field must contain values of 12 characters or less. This short user name is used for the following purposes:

- If LDAP authentication is enabled, but LDAP authorization is not enabled, this is used as an operating system user ID for authorization checks. In this case, the attribute must represent an operating system user ID.
- If LDAP authentication and authorization are both enabled, this is used as the user ID carried with the message in order for the LDAP user name to be rediscovered when the user ID inside the message needs to be used.

For example, on another queue manager, or when writing report messages. In this case, the attribute does not need to represent an operating system user ID, but must be a unique string. An employee serial number is an example of a good attribute for this purpose.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP* and is mandatory.

## **Text 'description' (TEXT)**

A short text description of the authentication information object.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

### **\*SYSDFTAI**

The text string is set to the system default value in `SYSTEM.DEFAULT.AUTHINFO.CRLLDAP`.

### **\*NONE**

The text is set to a blank string.

### **description**

The string length can be up to 64 characters enclosed in apostrophes.

## **User field (USRFIELD)**

If the user ID provided by an application for authentication does not contain a qualifier for the field in the LDAP user record, that is, it does not contain an '=' sign, this attribute identifies the field in the LDAP user record that is used to interpret the provided user ID.

This field can be blank. If this is the case, any unqualified user IDs use the [SHORTUSR](#) parameter to interpret the provided user ID.

The contents of this field will be concatenated with an '=' sign, together with the value provided by the application, to form the full user ID to be located in an LDAP user record. For example, the application provides a user of `fred` and this field has the value `cn`, then the LDAP repository will be searched for `cn=fred`.

This attribute is valid only for an **AUTHTYPE** of *\*IDPWLDAP*.

## User name (USERNAME)

The distinguished name of the user that is binding to the directory. The default user name is blank.

This field is only valid for \*CRLLDAP or \*IDPWLDAP authentication information objects.

The possible values are:

### \*SYSDFTAI

The user name is set to the system default value in SYSTEM.DEFAULT.AUTHINFO.CRLLDAP.

### \*NONE

The user name is blank.

### LDAP-user-name

Specify the Distinguished name of the LDAP user. The maximum string length is 1024 characters.

## User password (PASSWORD)

The password for the LDAP user.

This field is only valid for \*CRLLDAP or \*IDPWLDAP authentication information objects.

The possible values are:

### \*SYSDFTAI

The password is set to the system default value in SYSTEM.DEFAULT.AUTHINFO.CRLLDAP.

### \*NONE

The password is blank.

### LDAP-password

The LDAP user password. The maximum string length is 32 characters.

## CRTMQMCHL (Create MQ Channel)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Create MQ Channel (CRTMQMCHL) command creates a new MQ channel definition, specifying those attributes that are to be different from the default values.

## Parameters

Keyword	Description	Choices	Notes
<u>CHLNAME</u>	Channel name	Character value	Required, Key, Positional 1
<u>CHLTYPE</u>	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Required, Key, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Key, Positional 3
<u>REPLACE</u>	Replace	*NO, *YES	Optional, Positional 4
<u>TRPTYPE</u>	Transport type	*LU62, *TCP, *SYSDFTCHL	Optional, Positional 5

Table 244. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>TEXT</u>	Text 'description'	Character value, *BLANK, <b>*SYSDFTCHL</b>	Optional, Positional 6
<u>TGTMQMNAME</u>	Target Queue Manager	Character value, *NONE, <b>*SYSDFTCHL</b>	Optional, Positional 7
<u>CONNNAME</u>	Connection name	Character value, *NONE, <b>*SYSDFTCHL</b>	Optional, Positional 8
<u>TPNAME</u>	Transaction Program Name	Character value, *BLANK, <b>*SYSDFTCHL</b>	Optional, Positional 9
<u>MODENAME</u>	Mode Name	Character value, *BLANK, <b>*SYSDFTCHL</b>	Optional, Positional 10
<u>TMQNAME</u>	Transmission queue	Character value, <b>*SYSDFTCHL</b>	Optional, Positional 11
<u>MCANAME</u>	Message channel agent	Single values: <b>*SYSDFTCHL</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 12
	Qualifier 1: Message channel agent	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	
<u>MCAUSRID</u>	Message channel agent user ID	Character value, *NONE, *PUBLIC, <b>*SYSDFTCHL</b>	Optional, Positional 13
<u>MCATYPE</u>	Message channel agent Type	*PROCESS, *THREAD, <b>*SYSDFTCHL</b>	Optional, Positional 14
<u>BATCHINT</u>	Batch Interval	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 15
<u>BATCHSIZE</u>	Batch size	1-9999, <b>*SYSDFTCHL</b>	Optional, Positional 16
<u>DSCITV</u>	Disconnect interval	0-999999, <b>*SYSDFTCHL</b>	Optional, Positional 17
<u>SHORTTMR</u>	Short retry interval	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 18
<u>SHORTRTY</u>	Short retry count	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 19
<u>LONGTMR</u>	Long retry interval	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 20
<u>LONGRTY</u>	Long retry count	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 21
<u>SCYEXIT</u>	Security exit	Single values: <b>*SYSDFTCHL</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 22
	Qualifier 1: Security exit	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	

Table 244. Command parameters (continued)

Keyword	Description	Choices	Notes
<a href="#">CSCYEXIT</a>	Security exit	Character value, <b>*SYSDFTCHL</b> , *NONE	Optional, Positional 23
<a href="#">SCYUSRDATA</a>	Security exit user data	Character value, <b>*SYSDFTCHL</b> , *NONE	Optional, Positional 24
<a href="#">SNDEXIT</a>	Send exit	Single values: <b>*SYSDFTCHL</b> , *NONE Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 25
	Qualifier 1: Send exit	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	
<a href="#">CSNDEXIT</a>	Send exit	Single values: <b>*SYSDFTCHL</b> , *NONE Other values (up to 10 repetitions): <i>Character value</i>	Optional, Positional 26
<a href="#">SNDUSRDATA</a>	Send exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SYSDFTCHL</b> , *NONE	Optional, Positional 27
<a href="#">RCVEXIT</a>	Receive exit	Single values: <b>*SYSDFTCHL</b> , *NONE Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 28
	Qualifier 1: Receive exit	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	
<a href="#">CRCVEXIT</a>	Receive exit	Single values: <b>*SYSDFTCHL</b> , *NONE Other values (up to 10 repetitions): <i>Character value</i>	Optional, Positional 29
<a href="#">RCVUSRDATA</a>	Receive exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SYSDFTCHL</b> , *NONE	Optional, Positional 30
<a href="#">MSGEXIT</a>	Message exit	Single values: <b>*SYSDFTCHL</b> , *NONE Other values (up to 10 repetitions): <i>Qualified object name</i>	Optional, Positional 31
	Qualifier 1: Message exit	Name	
	Qualifier 2: Library	Name, <b>*CURLIB</b>	

Table 244. Command parameters (continued)


Keyword	Description	Choices	Notes
<a href="#">MSGUSRDATA</a>	Message exit user data	Values (up to 10 repetitions): <i>Character value</i> , <b>*SYSDFTCHL</b> , <b>*NONE</b>	Optional, Positional 32
<a href="#">MSGRTYEXIT</a>	Message retry exit	Single values: <b>*SYSDFTCHL</b> , <b>*NONE</b> Other values: <i>Qualified object name</i>	Optional, Positional 33
	Qualifier 1: Message retry exit	Name	
	Qualifier 2: Library	<i>Name</i> , <b>*CURLIB</b>	
<a href="#">MSGRTYDATA</a>	Message retry exit data	<i>Character value</i> , <b>*SYSDFTCHL</b> , <b>*NONE</b>	Optional, Positional 34
<a href="#">MSGRTYNBR</a>	Number of message retries	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 35
<a href="#">MSGRTYITV</a>	Message retry interval	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 36
<a href="#">CVTMSG</a>	Convert message	<b>*YES</b> , <b>*NO</b> , <b>*SYSDFTCHL</b>	Optional, Positional 37
<a href="#">PUTAUT</a>	Put authority	<b>*DFT</b> , <b>*CTX</b> , <b>*SYSDFTCHL</b>	Optional, Positional 38
<a href="#">SEQNUMWRAP</a>	Sequence number wrap	100-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 39
<a href="#">MAXMSGLEN</a>	Maximum message length	0-104857600, <b>*SYSDFTCHL</b>	Optional, Positional 40
<a href="#">HRTBTINTVL</a>	Heartbeat interval	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 41
<a href="#">NPMSPEED</a>	Non Persistent Message Speed	<b>*FAST</b> , <b>*NORMAL</b> , <b>*SYSDFTCHL</b>	Optional, Positional 42
<a href="#">CLUSTER</a>	Cluster Name	<i>Character value</i> , <b>*NONE</b> , <b>*SYSDFTCHL</b>	Optional, Positional 43
<a href="#">CLUSNL</a>	Cluster Name List	<i>Character value</i> , <b>*NONE</b> , <b>*SYSDFTCHL</b>	Optional, Positional 44
<a href="#">NETPRTY</a>	Network Connection Priority	0-9, <b>*SYSDFTCHL</b>	Optional, Positional 45
<a href="#">SSLCIPH</a>	TLS CipherSpec	Supported CipherSpecs are listed here: <a href="#">CipherSpecs you can use with IBM MQ TLS support</a> .   <b>Deprecated</b> CipherSpecs that you can re-enable if necessary are listed here: <a href="#">Deprecated CipherSpecs</a> .	Optional, Positional 46

Table 244. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<a href="#">SSLCAUTH</a>	TLS Client Authentication	*REQUIRED, *OPTIONAL, <b>*SYSDFTCHL</b>	Optional, Positional 47
<a href="#">SSLPEER</a>	TLS Peer name	<i>Character value</i> , *NONE, <b>*SYSDFTCHL</b>	Optional, Positional 48
<a href="#">LOCLADDR</a>	Local communication address	<i>Character value</i> , *NONE, <b>*SYSDFTCHL</b>	Optional, Positional 49
<a href="#">BATCHHB</a>	Batch Heartbeat Interval	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 50
<a href="#">USERID</a>	Task user identifier	<i>Character value</i> , *NONE, <b>*SYSDFTCHL</b>	Optional, Positional 51
<a href="#">PASSWORD</a>	Password	<i>Character value</i> , *NONE, <b>*SYSDFTCHL</b>	Optional, Positional 52
<a href="#">KAINT</a>	Keep Alive Interval	<i>Integer</i> , *AUTO, <b>*SYSDFTCHL</b>	Optional, Positional 53
<a href="#">COMPHDR</a>	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, <b>*SYSDFTCHL</b>	Optional, Positional 54
<a href="#">COMPMSG</a>	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST,, <b>*SYSDFTCHL</b>	Optional, Positional 55
<a href="#">MONCHL</a>	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, <b>*SYSDFTCHL</b>	Optional, Positional 56
<a href="#">STATCHL</a>	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, <b>*SYSDFTCHL</b>	Optional, Positional 57
<a href="#">CLWLRANK</a>	Cluster Workload Rank	0-9, <b>*SYSDFTCHL</b>	Optional, Positional 58
<a href="#">CLWLPRTY</a>	Cluster Workload Priority	0-9, <b>*SYSDFTCHL</b>	Optional, Positional 59
<a href="#">CLWLWGHT</a>	Cluster Channel Weight	1-99, <b>*SYSDFTCHL</b>	Optional, Positional 60
<a href="#">SHARECNV</a>	Sharing Conversations	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 61
<a href="#">PROPCTL</a>	Property Control	*COMPAT, *NONE, *ALL, <b>*SYSDFTCHL</b>	Optional, Positional 62
<a href="#">MAXINST</a>	Maximum Instances	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 63
<a href="#">MAXINSTC</a>	Maximum Instances Per Client	0-999999999, <b>*SYSDFTCHL</b>	Optional, Positional 64
<a href="#">CLNTWGHT</a>	Client Channel Weight	0-99, <b>*SYSDFTCHL</b>	Optional, Positional 65
<a href="#">AFFINITY</a>	Connection Affinity	*PREFERRED, *NONE, <b>*SYSDFTCHL</b>	Optional, Positional 66

Table 244. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>BATCHLIM</u>	Batch Data Limit	0-999999, <b>*SYSDFTCHL</b>	Optional, Positional 67
<u>DFTRECON</u>	Default client reconnection	*NO, *YES, *QMGR, *DISABLED, <b>*SYSDFTCHL</b>	Optional, Positional 68

### Channel name (CHLNAME)

Specifies the name of the new channel definition; the name can contain a maximum of 20 characters. Channel names must be unique. If a channel definition with this name already exists, REPLACE(\*YES) must be specified.

### Channel type (CHLTYPE)

Specifies the type of the channel being defined.

The possible values are:

**\*SDR**

Sender channel

**\*SVR**

Server channel

**\*RCVR**

Receiver channel

**\*RQSTR**

Requester channel

**\*SVRCN**

Server-connection channel

**\*CLUSSDR**

Cluster-sender channel

**\*CLUSRCVR**

Cluster-receiver channel

**\*CLTCN**

Client-connection channel

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

**message-queue-manager-name**

The name of a message queue manager.

### Replace (REPLACE)

Specifies whether the new channel definition should replace an existing channel definition with the same name.

The possible values are:



**\*NO**

Do not replace the existing channel definition. The command fails if the named channel definition already exists.

**\*YES**

Replace the existing channel definition. If there is no definition with the same name, a new definition is created.

**Transport type (TRPTYPE)**

Specifies the transmission protocol.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*LU62**

SNA LU 6.2.

**\*TCP**

Transmission Control Protocol / Internet Protocol (TCP/IP).

**Text 'description' (TEXT)**

Specifies text that briefly describes the channel definition.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Target Queue Manager (TGTMQMNAME)**

Specifies the name of the target queue manager.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The name of the target queue manager for a client connection channel (CHLTYPE) \*CLTCN is unspecified.

**message-queue-manager-name**

The name of the target message queue manager for a client connection channel (CHLTYPE) \*CLTCN.

For other channel types this parameter must not be specified.

**Connection name (CONNAME)**

Specifies the name of the machine to connect.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The connection name is blank.

**connection-name**

Specify the connection name as required by the transmission protocol:

- For \*LU62, specify the name of the CSI object.
- For \*TCP, specify either the host name, or the network address of the remote machine (or the local machine for cluster-receiver channels). This can be followed by an optional port number enclosed in parentheses.



On Multiplatforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

```
(1415)
```

The generated **CONNNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

Where a port is not specified the default port 1414 is assumed.

For cluster-receiver channels the connection name relates to the local queue manager, and for other channels it relates to the target queue manager.

This parameter is required for channels with channel type (CHLTYPE) of \*SDR, \*RQSTR, \*CLTCN and \*CLUSDR. It is optional for \*SVR and \*CLUSRCVR channels, and is not valid for \*RCVR or \*SVRCN channels.

**Transaction Program Name (TPNAME)**

This parameter is valid for channels with a TRPTYPE defined as LU 6.2 only.

This parameter must be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it must be set to blanks. The name is taken instead from the CPI-C Communications Side Object.

This parameter is not valid for channels with a CHLTYPE defined as \*RCVR.

The possible values are:

**\*SAME**

The value of this attribute does not change.

**\*NONE**

No transaction program name is specified.

**\*BLANK**

The transaction program name is taken from CPI-C Communications Side Object. The side object name must be specified in the CONNAME parameter.

**Transaction Program Name**

Specify the SNA transaction program name.

**Mode Name (MODENAME)**

This parameter is valid for channels with a TRPTYPE defined as LU 6.2. If TRPTYPE is not defined as LU 6.2 the data is ignored and no error message is issued.

If specified, the value must be set to the SNA mode name, unless the CONNAME contains a side-object name, in which case it must be set to blanks. The name is then taken from the CPI-C Communications Side Object.

This parameter is not valid for channels with CHLTYPE defined as \*RCVR or \*SVRCONN.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*BLANK**

Name will be taken from the CPI-C Communications Side Object. This must be specified in the CONNAME parameter.

**\*NONE**

No mode name is specified.

**SNA-mode-name**

Specify the SNA Mode Name

## Transmission queue (TMQNAME)

Specifies the name of the transmission queue.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**transmission-queue-name**

Specify the name of the transmission queue.

A transmission queue name is required if the channel type (CHLTYPE) is \*SDR or \*SVR. For other channel types, the parameter must not be specified.

## Message channel agent (MCANAME)

This parameter is reserved and should not be used.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The MCA program name is blank.

This parameter cannot be specified for a channel type (CHLTYPE) of \*RCVR, \*SVRCN, or \*CLTCN.

## Message channel agent user ID (MCAUSRID)

Specifies the message channel agent user identifier which is to be used by the message channel agent for authorization to access MQ resources, including (if PUTAUT is \*DFT) authorization to put the message to the destination queue for receiver or requester channels.

The possible values are:

**\*SYSDFTCHL**

The value is taken from the system default channel for the type of the channel being created.

**\*NONE**

The message channel agent uses its default user identifier.

**\*PUBLIC**

Uses the public authority.

**mca-user-identifier**

Specify the user identifier to be used.

This parameter cannot be specified for a channel type (CHLTYPE) of \*CLTCN.

## Message channel agent Type (MCATYPE)

Specifies whether the message-channel-agent program should run as a thread or a process.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **\*PROCESS**

The message channel agent runs as a separate process.

### **\*THREAD**

The message channel agent runs as a separate thread.

This parameter can only be specified for a channel type (CHLTYPE) of \*SDR, \*SVR, \*RQSTR, \*CLUSSDR or \*CLUSRCVR.

## Batch Interval (BATCHINT)

The minimum amount of time, in milliseconds, that a channel will keep a batch open.

The batch is terminated by which ever of the following occurs first: BATCHSZ messages have been sent, BATCHLIM bytes have been sent, or the transmission queue is empty and BATCHINT is exceeded.

The default value is 0, which means that the batch is terminated as soon as the transmission queue becomes empty (or the BATCHSZ limit is reached).

The value must be in the range 0 through 999999999.

This parameter is valid for channels with CHLTYPE defined as \*SDR, \*SVR, \*CLUSSDR, or \*CLUSRCVR.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **batch-interval**

Specify a value in the range 0 through 999999999. A value of 0 indicates the batch will be terminated as soon as the transmission queue is empty,

## Batch size (BATCHSIZE)

Specifies the maximum number of messages that should be sent down a channel before a checkpoint is taken.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **batch-size**

Specify a value in the range 1 through 9999

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

## Disconnect interval (DSCITV)

Specifies the disconnect interval, which defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before closing the channel.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **disconnect-interval**

Specify a value in the range 0 through 999999. A value of 0 indicates an indefinite wait.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR or \*CLTCN.

### Short retry interval (SHORTTMR)

Specifies the short retry wait interval for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the interval between attempts to establish a connection to the remote machine.

The possible values are:

#### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

#### **short-retry-interval**

Specify a value in the range 0 through 999999999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this are treated as 999999.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

### Short retry count (SHORTRTY)

Specifies the short retry count for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of attempts that are made to establish a connection to the remote machine, at intervals specified by SHORTTMR, before the (normally longer) LONGRTY and LONGTMR are used.

The possible values are:

#### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

#### **short-retry-count**

Specify a value in the range 0 through 999999999. A value of 0 means that no retries are allowed.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

### Long retry interval (LONGTMR)

Specifies the long retry wait interval for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

The possible values are:

#### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

#### **long-retry-interval**

Specify a value in the range 0 through 999999999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this are treated as 999999.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

### Long retry count (LONGRTY)

Specifies the long retry count for a sender, server or cluster channel (\*SDR, \*SVR, \*CLUSSDR or \*CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by LONGTMR, after the count specified by SHORTRTY has been exhausted. An error message is logged if the connection is not established after the defined number of attempts.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**long-retry-count**

Specify a value in the range 0 through 999999999. A value of 0 means that no retries are allowed.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

### **Security exit (SCYEXIT)**

Specifies the name of the program to be called as the security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.

- On receipt of a response to a security message flow.

Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The security exit program is not invoked.

**security-exit-name**

Specify the name of the security exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

### **Security exit (CSCYEXIT)**

Specifies the name of the program to be called as the client security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.

- On receipt of a response to a security message flow.

Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the SYSTEM.DEF.CLNTCONN channel.

**\*NONE**

The client security exit program is not invoked.

**security-exit-name**

Specify the name of the client security exit program.

### **Security exit user data (SCYUSRDATA)**

Specifies a maximum of 32 characters of user data that is passed to the channel security exit program.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The user data for the security exit is not specified.

**security-exit-user-data**

Specify the user data for the security exit program.

**Send exit (SNDEXIT)**

Specifies the entry point of the program to be called as the send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The send exit is not invoked.

**send-exit-name**

Specify the name of the send exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

**Send exit (CSNDEXIT)**

Specifies the entry point of the program to be called as the client send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the SYSTEM.DEF.CLNTCONN channel.

**\*NONE**

The client send exit is not invoked.

**send-exit-name**

Specify the name of the client send exit program.

**Send exit user data (SNDUSRDATA)**

Specifies a maximum of 32 characters of user data that is passed to the send exit program.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The user data for the send exit program is not specified.

**send-exit-user-data**

Specify a maximum of 32 characters of user data for the send exit program.

## Receive exit (RCVEXIT)

Specifies the entry point of the program to be called as the receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **\*NONE**

The receive exit program is not invoked.

### **receive-exit-name**

Specify the name of the receive exit program.

### **library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

## Receive exit (RCVEXIT)

Specifies the entry point of the program to be called as the client receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the SYSTEM.DEF.CLNTCONN channel.

### **\*NONE**

The client receive exit program is not invoked.

### **receive-exit-name**

Specify the name of the client receive exit program.

## Receive exit user data (RCVUSRDATA)

Specifies user data that is passed to the receive exit.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **\*NONE**

The user data for the receive exit program is not specified.

### **receive-exit-user-data**

Specify a maximum of 32 characters of user data for the receive exit program.

## Message exit (MSGEXIT)

Specifies the entry point of the program to be called as the message exit. If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **\*NONE**

The message exit program is not invoked.



**message-exit-name**

Specify the name of the message exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Message exit user data (MSGUSRDATA)**

Specifies user data that is passed to the message exit program.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The user data for the message exit program is not specified.

**message-exit-user-data**

Specify a maximum of 32 characters of user data for the message exit program.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Message retry exit (MSGRTYEXIT)**

Specifies the entry point of the program to be called as the message retry exit.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The message retry exit program is not invoked.

**message-retry-exit-name**

Specify the name of the message retry exit program.

**library-name**

Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

**Message retry exit data (MSGRTYDATA)**

Specifies user data that is passed to the message retry exit program.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

The user data for the message retry exit program is not specified.

**message-retry-exit-user-data**

Specify a maximum of 32 characters of user data for the message retry exit program.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

## Number of message retries (MSGRTYNBR)

Specifies the number of times the channel will retry before it decides it cannot deliver the message. This attribute controls the action of the MCA only if the message-retry exit name is blank, the value of MSGRTYNBR is passed to the exit for the exit's use, but the number of retries performed is controlled by the exit, and not by this attribute.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **message-retry-number**

Specify a value in the range 0 through 999999999. A value of 0 signifies no retries will be performed.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

## Message retry interval (MSGRTYITV)

Specifies the minimum interval of time that must pass before the channel can retry the MQPUT operation. This time is in milliseconds.

This attribute controls the action of the MCA only if the message-retry exit name is blank, the value of MSGRTYITV is passed to the exit for the exit's use, but the retry interval is controlled by the exit, and not by this attribute.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **message-retry-number**

Specify a value in the range 0 through 999999999. A value of 0 signifies that the retry will be performed as soon as possible.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

## Convert message (CVTMSG)

Specifies whether the application data in the message should be converted before the message is transmitted.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel for the type of channel being created.

### **\*YES**

The application data in the message is converted before sending.

### **\*NO**

The application data in the message is not converted before sending.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

## Put authority (PUTAUT)

Specifies whether the user identifier in the context information associated with a message should be used to establish authority to put the message on the destination queue. This applies only to receiver and requester (\*CLUSRCVR, \*RCVR and \*RQSTR) channels.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*DFT**

No authority check is made before the message is put on the destination queue.

**\*CTX**

The user identifier in the message context information is used to establish authority to put the message.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLTCN, \*SVRCN or \*CLUSSDR.

**Sequence number wrap (SEQNUMWRAP)**

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

**Note:** The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**sequence-number-wrap-value**

Specify a value in the range 100 through 999999999.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Maximum message length (MAXMSGLEN)**

Specifies the maximum message length that can be transmitted on the channel. This is compared with the value for the remote channel and the actual maximum is the lower of the two values.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**maximum-message-length**

Specify a value in the range 0 through 104857600. A value of 0 signifies that the maximum length is unlimited.

**Heartbeat interval (HRTBTINTVL)**

Specifies The time, in seconds, between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. The heartbeat exchange gives the receiving MCA the opportunity to quiesce the channel.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**heart-beat-interval**

Specify a value in the range 0 through 999999999. A value of 0 means that no heartbeat exchanges are to take place.

**Note:** For implementation reasons, the maximum heartbeat interval that can be used is 999999; values exceeding this are treated as 999999.

**Non Persistent Message Speed (NPMSPEED)**

Specifies whether the channel supports Fast Non Persistent Messages.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute does not change.

**\*FAST**

The channel supports fast non persistent messages.

**\*NORMAL**

The channel does not support fast non persistent messages.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

### **Cluster Name (CLUSTER)**

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

This parameter is valid only for \*CLUSDR and \*CLUSRCVR channels. If the CLUSNL parameter is non-blank, this parameter must be blank.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

No cluster name is specified.

**cluster-name**

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

### **Cluster Name List (CLUSNL)**

The name of the namelist that specifies a list of clusters to which the channel belongs

This parameter is valid only for \*CLUSDR and \*CLUSRCVR channels. If the CLUSTER parameter is non-blank, this parameter must be blank.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*NONE**

No cluster namelist is specified.

**cluster-name-list**

The name of the namelist specifying a list of clusters to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

### **Network Connection Priority (NETPRTY)**

The priority for the network connection. Distributed queuing chooses the path with the highest priority if there are multiple paths available. The value must be in the range between 0 and 9 where 0 is the lowest priority.

This parameter is valid only for \*CLUSRCVR channels.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**network-connection-priority**

Specify a value in the range 0 through 9; 0 is the lowest priority.

## TLS CipherSpec (SSLCIPH)


SSLCIPH specifies the CipherSpec used in TLS channel negotiation. The possible values are:

### \*SYSDFTCHL

The value of this attribute is taken from the system default channel of the specified type.

### cipherspec

The name of the CipherSpec.

**Note:**  From IBM MQ 8.0.0 Fix Pack 2, the SSLv3 protocol and the use of some IBM MQ CipherSpecs is deprecated. For more information, see [Deprecated CipherSpecs](#).

## TLS Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel should carry out client authentication over TLS. The parameter is used only for channels with SSLCIPH specified.

The possible values are:

### \*SYSDFTCHL

The value of this attribute is taken from the system default channel of the specified type.

### \*REQUIRED

Client authentication is required.

### \*OPTIONAL

Client authentication is optional.

This parameter cannot be specified for channel types (CHLTYPE) \*SDR, \*CLTCN or \*CLUSSDR.

## TLS Peer name (SSLPEER)

SSLPEER specifies the X500 peer name used in TLS channel negotiation. The possible values are:

### \*SYSDFTCHL

The value of this attribute is taken from the system default channel of the specified type.

### x500peername

The X500 peer name to use.

**Note:** An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both SSLPEER on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns in order to connect. For more information, see [Channel authentication records](#).

## Local communication address (LOCLADDR)

Specifies the local communication address for the channel.

This parameter is only valid for \*SDR, \*SVR, \*RQSTR, \*CLUSSDR, \*CLUSRCVR and \*CLTCN channels.

The possible values are:

### \*SAME

The attribute is unchanged.

### \*NONE

The connection is blank.

### local-address

Only valid for transport type TCP/IP. Specify the optional IP address and optional port or port range used for outbound TCP/IP communications. The format is:

```
LOCLADDR([ip-addr][(low-port[,high-port])][, [ip-addr][(low-port[,high-port])]])
```

## Batch Heartbeat Interval (BATCHEB)

The time in milliseconds used to determine whether batch heartbeating occurs on this channel. Batch heartbeating allows sender-type channels to determine whether the remote channel instance is still active before going in-doubt. A batch heartbeat will occur if a sender-type channel has not communicated with the remote channel within the specified time.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **batch-heartbeat-interval**

Specify a value in the range 0 through 999999999. A value of 0 indicates that batch heartbeating is not to be used.

**Note:** For implementation reasons, the maximum batch heartbeat interval that can be used is 999999; values exceeding this are treated as 999999.

This parameter cannot be specified for channel types (CHLTYPE) \*RCVR, \*RQSTR, \*CLTCN or \*SVRCN.

## Task user identifier (USERID)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

This parameter is valid only for channels with a channel type (CHLTYPE) of \*SDR, \*SVR, \*RQSTR, \*CLTCN or \*CLUSSDR.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **\*NONE**

No user identifier is specified.

### **user-identifier**

Specify the task user identifier.

## Password (PASSWORD)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

This parameter is valid only for channels with a channel type (CHLTYPE) of \*SDR, \*SVR, \*RQSTR, \*CLTCN or \*CLUSSDR.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **\*NONE**

No password is specified.

### **Password**

Specify the password.

## Keep Alive Interval (KAINT)

Specifies the Keep Alive timing interval for this channel.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel for the type of channel being created.

### **\*AUTO**

The Keep Alive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated HBINT is greater than 0, Keep Alive interval is set to that value plus 60 seconds.
- If the negotiated HBINT is 0, the value used is that specified by the KEEPALIVEOPTIONS statement in the TCP profile configuration data set.

### **keep-alive-interval**

Specify a value in the range 0 through 99999.

## Header Compression (COMPHDR)

The list of header data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (\*SDR, \*SVR, \*CLUSSDR, \*CLUSRCVR and \*CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **\*NONE**

No header data compression is performed.

### **\*SYSTEM**

Header data compression is performed.

## Message Compression (COMPMSG)

The list of message data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (\*SDR, \*SVR, \*CLUSSDR, \*CLUSRCVR and \*CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

### **\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

### **\*NONE**

No message data compression is performed.

### **\*RLE**

Message data compression is performed using run-length encoding.

### **\*ZLIBFAST**

Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

### **\*ZLIBHIGH**

Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

**\*ANY**

Any compression technique supported by the queue manager can be used. Only valid for channel types Receiver, Requester and Server-Connection.

**Channel Monitoring (MONCHL)**

Controls the collection of online monitoring data.

Online monitoring data is not collected when the queue manager attribute MONCHL is set to \*NONE.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*QMGR**

The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONCHL.

**\*NONE**

Online Monitoring Data collection for this channel is disabled.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for a channel type (CHLTYPE) of \*CLTCN.

**Channel Statistics (STATCHL)**

Controls the collection of statistics data.

Statistics data is not collected when the queue manager attribute STATCHL is set to \*NONE.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*QMGR**

Statistics data collection is based upon the setting of the queue manager attribute STATCHL.

**\*NONE**

Statistics data collection for this channel is disabled.

**\*LOW**

Statistics data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Statistics data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Statistics data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for channel types (CHLTYPE) \*CLTCN or \*SVRCN.

**Cluster Workload Rank (CLWLRANK)**

Specifies the cluster workload rank of the channel.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.



**cluster-workload-rank**

The cluster workload rank of the channel in the range 0 through 9.

**Cluster Workload Priority (CLWLPRTY)**

Specifies the cluster workload priority of the channel.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**cluster-workload-rank**

The cluster workload priority of the channel in the range 0 through 9.

**Cluster Channel Weight (CLWLWGHT)**

Specifies the cluster workload weight of the channel.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**cluster-workload-rank**

The cluster workload weight of the channel in the range 1 through 99.

**Sharing Conversations (SHARECNV)**

Specifies the maximum the number of conversations which can be shared over a particular TCP/IP client channel instance (socket).

This parameter is valid for channels with CHLTYPE defined as \*CLTCN or \*SVRCN.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**0**

Specifies no sharing of conversations over a TCP/IP socket. The channel instance runs in a mode prior to that of IBM WebSphere MQ 7.0, with regard to:

- Administrator stop-quiesce
- Heartbeating
- Read ahead

**1**

Specifies no sharing of conversations over a TCP/IP socket. Client heartbeating and read ahead are available, whether in an MQGET call or not, and channel quiescing is more controllable.

**shared-conversations**

The number of shared conversations in the range 2 through 999999999.

**Note:** If the client-connection SHARECNV value does not match the server-connection SHARECNV value, the lower of the two values is used.

**Property Control (PROPCTL)**

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor).

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**\*COMPAT**

If the message contains a property with a prefix of "mcd.", "jms.", "usr." or "mqext." then all optional message properties, except those in the message descriptor (or extension) will be placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.

**\*NONE**

All properties of the message, except those in the message descriptor (or extension), will be removed from the message before the message is sent to the remote queue manager.

**\*ALL**

All properties of the message will be included with the message when it is sent to the remote queue manager. The properties, except those in the message descriptor (or extension), will be placed in one or more MQRFH2 headers in the message data.

**Maximum Instances (MAXINST)**

Specifies the maximum number of clients that can simultaneously connect to the queue manager via this server-connection channel object.

This attribute is valid only for server-connection channels.

The possible values are:

**\*SYSDFT**

The value of this attribute is taken from the system default channel of the specified type.

**maximum-instances**

The maximum number of simultaneous instances of the channel in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

**Maximum Instances Per Client (MAXINSTC)**

Specifies the maximum number of simultaneous instances of an individual server-connection channel which can be started from a single client.

In this context, multiple client connections originating from the same remote network address are considered to be a single client.

This attribute is valid only for server-connection channels.

The possible values are:

**\*SYSDFT**

The value of this attribute is taken from the system default channel of the specified type.

**maximum-instances-per-client**

The maximum number of simultaneous instances of the channel which can be in the started from a single client in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running from individual clients, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

**Client Channel Weight (CLNTWGHT)**

The client channel weighting attribute is used so client channel definitions can be selected at random based on their weighting when more than one suitable definition is available.

The possible values are:

**\*SYSDFT**

The value of this attribute is taken from the system default channel of the specified type.

**client-channel-weight**

The client channel weight in the range 0 through 99.

**Connection Affinity (AFFINITY)**

The channel affinity attribute is used so client applications that connect multiple times using the same queue manager name can choose whether to use the same client channel definition for each connection.

The possible values are:

**\*SYSDFT**

The value of this attribute is taken from the system default channel of the specified type.

**\*PREFERRED**

The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions based on the weighting with any applicable CLNTWGHT(0) definitions first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non CLNTWGHT(0) definitions are moved to the end of the list. CLNTWGHT(0) definitions remain at the start of the list and are selected first for each connection.

**\*NONE**

The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable CLNTWGHT(0) definitions selected first in alphabetical order.

**Batch Data Limit (BATCHLIM)**

The limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point. A sync point is taken after the message that caused the limit to be reached has flowed across the channel. A value of zero in this attribute means that no data limit is applied to batches over this channel.

The batch is terminated when one of the following conditions is met:

- **BATCHSZ** messages have been sent.
- **BATCHLIM** bytes have been sent.
- The transmission queue is empty and **BATCHINT** is exceeded.

This parameter is valid only for channels with a channel type (**CHLTYPE**) of SDR, SVR, CLUSSDR, or CLUSRCVR.

The value must be in the range 0 - 999999. The default value is 5000.

The **BATCHLIM** parameter is supported on all platforms.

The possible values are:

**\*SYSDFTCHL**

The value of this attribute is taken from the system default channel of the specified type.

**batch-data-limit**

Specify a value in the range 0 through 999999.

This parameter can only be specified for channel types (CHLTYPE) \*SDR, \*SVR, \*CLUSSDR, or \*CLUSRCVR.

**Pending Reset Sequence Number (RESETSEQ)**

Pending reset sequence number.

This is the sequence number from an outstanding request and it indicates a user RESET CHANNEL command request is outstanding.

The possible value is:

### pending-reset-sequence-number

A value of zero indicates that there is no outstanding RESET CHANNEL. The value can be in the range 1 - 999999999.

### Default client reconnection (DFTRECON)

Specifies whether a client connection automatically reconnects a client application if its connection breaks.

#### \*SYSDFTCHL

The value of this attribute is taken from the system default channel of the specified type.

#### \*NO

Unless overridden by **MQCONN**, the client is not reconnected automatically.

#### \*YES

Unless overridden by **MQCONN**, the client reconnects automatically.

#### \*QMGR

Unless overridden by **MQCONN**, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as MQCNO\_RECONNECT\_Q\_MGR.

#### \*DISABLED

Reconnection is disabled, even if requested by the client program using the **MQCONN** MQI call.

This parameter is specified for a client connection channel, (CHLTYPE) \*CLTCN

## IBM i CRTMQMLSR (Create MQ Listener)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Create MQ Listener (CRTMQMLSR) command creates a new MQ listener definition, specifying those attributes that are to be different from the default.

### Parameters

Keyword	Description	Choices	Notes
<u>LSRNAME</u>	Listener name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Key, Positional 2
<u>REPLACE</u>	Replace	*NO, *YES	Optional, Positional 3
<u>TEXT</u>	Text 'description'	Character value, *BLANK, *SYSDFTLSR	Optional, Positional 4
<u>CONTROL</u>	Listener control	*SYSDFTLSR, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 5
<u>PORT</u>	Port number	0-65535, *SYSDFTLSR	Optional, Positional 6
<u>IPADDR</u>	IP Address	Character value, *BLANK, *SYSDFTLSR	Optional, Positional 7
<u>BACKLOG</u>	Listener backlog	0-999999999, *SYSDFTLSR	Optional, Positional 8

## Listener name (LSRNAME)

The name of the new MQ listener definition to be created.

The possible values are:

### **listener-name**

Specify the name of the listener definition. The maximum length of the string is 48 bytes.

## Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.

### **queue-manager-name**

The name of a message queue manager.

## Replace (REPLACE)

If a listener definition with the same name already exists, this specifies whether it is to be replaced.

The possible values are:

### **\*NO**

This definition does not replace any existing listener definition with the same name. The command fails if the named listener definition already exists.

### **\*YES**

Replace the existing listener definition. If there is no definition with the same name, a new definition is created.

## Text 'description' (TEXT)

Specifies text that briefly describes the listener definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

### **\*SYSDFTLSR**

The value of this attribute is taken from the system default listener.

### **\*BLANK**

The text is set to a blank string.

### **description**

Specify the new descriptive information.

## Listener control (CONTROL)

Whether the listener starts automatically when the queue manager is started.

The possible values are:

### **\*SYSDFTLSR**

The value for this attribute is taken from the system default listener.

### **\*MANUAL**

The listener is not automatically started or stopped.

### **\*QMGR**

The listener is started and stopped as the queue manager is started and stopped.

**\*STARTONLY**

The listener is started as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**Port number (PORT)**

The port number to be used by the listener.

The possible values are:

**\*SYSDFTLSR**

The value for this attribute is taken from the system default listener.

**port-number**

The port number to be used.

**IP Address (IPADDR)**

The IP address to be used by the listener.

The possible values are:

**\*SYSDFTLSR**

The value for this attribute is taken from the system default listener.

**ip-addr**

The IP address to be used.

**Listener backlog (BACKLOG)**

The number of concurrent connection requests the listener supports.

The possible values are:

**\*SYSDFTLSR**

The value for this attribute is taken from the system default listener.

**backlog**

The number of concurrent connection requests supported.

 **CRTMQMNL (Create MQ Namelist)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Create MQ Namelist (CRTMQMNL) command creates a new MQ namelist. A namelist is an MQ object that contains a list of other MQ objects. Typically, namelists are used by applications, for example trigger monitors, where they are used to identify a group of queues. A namelist is maintained independently of applications, therefore you can update it without stopping any of the applications that use it.

**Parameters**

<i>Table 246. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>NAMELIST</u>	Namelist	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 2

Table 246. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>REPLACE</u>	Replace	<b>*NO</b> , <b>*YES</b>	Optional, Positional 3
<u>TEXT</u>	Text 'description'	<i>Character value</i> , <b>*BLANK</b> , <b>*SYSDFTNL</b>	Optional, Positional 4
<u>NAMES</u>	List of Names	Values (up to 256 repetitions): <i>Character value</i> , <b>*BLANKS</b> , <b>*SYSDFTNL</b> , <b>*NONE</b>	Optional, Positional 5

### Namelist (NAMELIST)

The name of the namelist to be created.

#### **namelist**

Specify the name of the namelist. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### **\*DFT**

The default queue manager is used.

#### **message-queue-manager-name**

Specify the name of the queue manager.

### Replace (REPLACE)

Specifies whether the new namelist should replace an existing namelist with the same name.

#### **\*NO**

Do not replace the existing namelist. The command fails if the named namelist already exists.

#### **\*YES**

Replace the existing namelist. If there is no namelist with the same name, a new namelist is created.

### Text 'description' (TEXT)

Specifies text that briefly describes the namelist.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

#### **\*SYSDFTNL**

The value of the attribute is taken from the system default namelist.

#### **description**

Specify no more than 64 characters enclosed in apostrophes.

### List of Names (NAMES)

List of names. This is the list of names to be created. The names can be of any type, but must conform to the rules for naming MQ objects.

#### **\*SYSDFTNL**

The value of the attribute is taken from the system default namelist.

## namelist

The list to create. An empty list is valid.

## IBM i CRTMQMPRC (Create MQ Process)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Create MQ Process (CRTMQMPRC) command creates a new MQ process definition, specifying those attributes that are different from the default.

## Parameters

Keyword	Description	Choices	Notes
<u>PRCNAME</u>	Process name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Key, Positional 2
<u>REPLACE</u>	Replace	*NO, *YES	Optional, Positional 3
<u>TEXT</u>	Text 'description'	Character value, *BLANK, *SYSDFTPRC	Optional, Positional 4
<u>APPTYPE</u>	Application type	Integer, *DEF, *CICS, *UNIX, *OS400, *WINDOWS, *WINDOWS_NT,	Optional, Positional 5
<u>APPID</u>	Application identifier	Character value, *SYSDFTPRC	Optional, Positional 6
<u>USRDATA</u>	User data	Character value, *SYSDFTPRC, *NONE	Optional, Positional 7
<u>ENVDATA</u>	Environment data	Character value, *SYSDFTPRC, *NONE	Optional, Positional 8

### Process name (PRCNAME)

The name of the new MQ process definition to be created.

The possible values are:

#### process-name

Specify the name of the new MQ process definition. The name can contain up to 48 characters.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.



## Replace (REPLACE)

If a process definition with the same name already exists, this specifies whether it is replaced.

The possible values are:

### **\*NO**

This definition does not replace any existing process definition with the same name. The command fails if the named process definition already exists.

### **\*YES**

Replace the existing process definition. If there is no definition with the same name, a new definition is created.

## Text 'description' (TEXT)

Specifies text that briefly describes the process definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

### **\*SYSDFTPRC**

The value of this attribute is taken from the system default process.

### **\*BLANK**

The text is set to a blank string.

### **description**

Specify the new descriptive information.

## Application type (APPTYPE)

The type of application started.

The possible values are:

### **\*DEF**

Specifying DEF causes the default application type for the platform at which the command is interpreted to be stored in the process definition. This default cannot be changed by the installation. If the platform supports clients, the default is interpreted as the default application type of the server.

### **\*CICS**

Represents a CICS/400 application.

### **\*UNIX**

Represents a UNIX or Linux application.

### **\*OS400**

Represents an IBM i application.

### **\*WINDOWS**

Represents a Windows application.

### **\*WINDOWS\_NT**

Represents a Windows NT application.

### **integer**

User-defined application type in the range 65536 through 999999999.

## Application identifier (APPID)

Application identifier. This is the name of the application to be started, on the platform for which the command is processing. It is typically a program name and library name.

The possible values are:

**\*SYSDFTPRC**

The value for this attribute is taken from the system default process.

**application-id**

The maximum length is 256 characters.

**User data (USRDATA)**

A character string that contains user information pertaining to the application, as defined by APPID, to start.

The possible values are:

**\*SYSDFTPRC**

The value for this attribute is taken from the system default process.

**\*NONE**

The user data is blank.

**user-data**

Specify up to 128 characters of user data.

**Environment data (ENVDATA)**

A character string that contains environment information pertaining to the application, as defined by APPID, to start.

The possible values are:

**\*SYSDFTPRC**

The value for this attribute is taken from the system default process.

**\*NONE**

The environment data is blank.

**environment-data**

The maximum length is 128 characters.

 **CRTMQMQ (Create MQ Queue)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Create MQ Queue (CRTMQMQ) command creates a queue definition with the specified attributes. All attributes that are not specified are set to the default value for the type of queue that is created.

**Parameters**

*Table 248. Command parameters*

Keyword	Description	Choices	Notes
<u>QNAME</u>	Queue name	Character value	Required, Key, Positional 1
<u>QTYPE</u>	Queue type	*ALS, *LCL, *MDL, *RMT	Required, Key, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Key, Positional 3
<u>REPLACE</u>	Replace	*NO, *YES	Optional, Positional 4

Table 248. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>TEXT</u>	Text 'description'	<i>Character value</i> , *BLANK, <b>*SYSDFTQ</b>	Optional, Positional 5
PUTENBL	Put enabled	<b>*SYSDFTQ</b> , *NO, *YES	Optional, Positional 6
<u>DFTPTY</u>	Default message priority	0-9, <b>*SYSDFTQ</b>	Optional, Positional 7
<u>DFTMSGPST</u>	Default message persistence	<b>*SYSDFTQ</b> , *NO, *YES	Optional, Positional 8
<u>PRCNAME</u>	Process name	<i>Character value</i> , *NONE, <b>*SYSDFTQ</b>	Optional, Positional 9
<u>TRGENBL</u>	Triggering enabled	<b>*SYSDFTQ</b> , *NO, *YES	Optional, Positional 10
<u>GETENBL</u>	Get enabled	<b>*SYSDFTQ</b> , *NO, *YES	Optional, Positional 11
SHARE	Sharing enabled	<b>*SYSDFTQ</b> , *NO, *YES	Optional, Positional 12
<u>DFTSHARE</u>	Default share option	<b>*SYSDFTQ</b> , *NO, *YES	Optional, Positional 13
<u>MSGDLYSEQ</u>	Message delivery sequence	<b>*SYSDFTQ</b> , *PTY, *FIFO	Optional, Positional 14
<u>HDNBKT CNT</u>	Harden backout count	<b>*SYSDFTQ</b> , *NO, *YES	Optional, Positional 15
<u>TRGTYPE</u>	Trigger type	<b>*SYSDFTQ</b> , *FIRST, *ALL, *DEPTH, *NONE	Optional, Positional 16
<u>TRGDEPTH</u>	Trigger depth	1-999999999, <b>*SYSDFTQ</b>	Optional, Positional 17
<u>TRGMSGPTY</u>	Trigger message priority	0-9, <b>*SYSDFTQ</b>	Optional, Positional 18
<u>TRGDATA</u>	Trigger data	<i>Character value</i> , *NONE, <b>*SYSDFTQ</b>	Optional, Positional 19
<u>RTNITV</u>	Retention interval	0-999999999, <b>*SYSDFTQ</b>	Optional, Positional 20
<u>MAXDEPTH</u>	Maximum queue depth	0-999999999, <b>*SYSDFTQ</b>	Optional, Positional 21
<u>MAXMSGLEN</u>	Maximum message length	0-104857600, <b>*SYSDFTQ</b>	Optional, Positional 22
<u>BKTTHLD</u>	Backout threshold	0-999999999, <b>*SYSDFTQ</b>	Optional, Positional 23
<u>BKTQNAME</u>	Backout requeue name	<i>Character value</i> , *NONE, <b>*SYSDFTQ</b>	Optional, Positional 24
<u>INITQNAME</u>	Initiation queue	<i>Character value</i> , *NONE, <b>*SYSDFTQ</b>	Optional, Positional 25
<u>USAGE</u>	Usage	<b>*SYSDFTQ</b> , *NORMAL, *TMQ	Optional, Positional 26
<u>DFNTYPE</u>	Definition type	<b>*SYSDFTQ</b> , *TEMPDYN, *PERMDYN	Optional, Positional 27
<u>TGTQNAME</u>	Target object	<i>Character value</i> , <b>*SYSDFTQ</b>	Optional, Positional 28
<u>RMTQNAME</u>	Remote queue	<i>Character value</i> , <b>*SYSDFTQ</b> , *NONE	Optional, Positional 29
<u>RMTMQMNAME</u>	Remote Message Queue Manager	<i>Character value</i> , <b>*SYSDFTQ</b>	Optional, Positional 30

Table 248. Command parameters (continued)

<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>TMQNAME</u>	Transmission queue	Character value, *NONE, *SYSDFTQ	Optional, Positional 31
<u>HIGHTHLD</u>	Queue depth high threshold	0-100, *SYSDFTQ	Optional, Positional 32
<u>LOWTHLD</u>	Queue depth low threshold	0-100, *SYSDFTQ	Optional, Positional 33
<u>FULLEVT</u>	Queue full events enabled	*SYSDFTQ, *NO, *YES	Optional, Positional 34
<u>HIGHEVT</u>	Queue high events enabled	*SYSDFTQ, *NO, *YES	Optional, Positional 35
<u>LOWEVT</u>	Queue low events enabled	*SYSDFTQ, *NO, *YES	Optional, Positional 36
<u>SRVITV</u>	Service interval	0-999999999, *SYSDFTQ	Optional, Positional 37
<u>SRVEVT</u>	Service interval events	*SYSDFTQ, *HIGH, *OK, *NONE	Optional, Positional 38
<u>DISTLIST</u>	Distribution list support	*SYSDFTQ, *NO, *YES	Optional, Positional 39
<u>CLUSTER</u>	Cluster Name	Character value, *SYSDFTQ, *NONE	Optional, Positional 40
<u>CLUSNL</u>	Cluster Name List	Character value, *NONE, *SYSDFTQ	Optional, Positional 41
<u>DEFBIND</u>	Default Binding	*SYSDFTQ, *OPEN, *NOTFIXED, *GROUP	Optional, Positional 42
<u>CLWLRANK</u>	Cluster Workload Rank	0-9, *SYSDFTQ	Optional, Positional 43
<u>CLWLPRTY</u>	Cluster Workload Priority	0-9, *SYSDFTQ	Optional, Positional 44
<u>CLWLUSEQ</u>	Cluster workload queue use	*SYSDFTQ, *QMGR, *LOCAL, *ANY	Optional, Positional 45
<u>MONQ</u>	Queue Monitoring	*SYSDFTQ, *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 46
<u>STATQ</u>	Queue Statistics	*SYSDFTQ, *QMGR, *OFF, *ON	Optional, Positional 47
<u>ACCTQ</u>	Queue Accounting	*SYSDFTQ, *QMGR, *OFF, *ON	Optional, Positional 48
<u>NPMCLASS</u>	Non Persistent Message Class	*SYSDFTQ, *NORMAL, *HIGH	Optional, Positional 49
<u>MSGREADAHD</u>	Message Read Ahead	*SYSDFTQ, *DISABLED, *NO, *YES	Optional, Positional 50
<u>DFTPUTRESP</u>	Default Put Response	*SYSDFTQ, *SYNC, *ASYNC	Optional, Positional 51
<u>PROPCTL</u>	Property Control	*SYSDFTQ, *COMPAT, *NONE, *ALL, *FORCE, *V6COMPAT	Optional, Positional 52
<u>TARGETYPE</u>	Target Type	*SYSDFTQ, *QUEUE, *TOPIC	Optional, Positional 53

Table 248. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>CUSTOM</u>	Custom attribute	Character value, *BLANK, *SYSDFTQ	Optional, Positional 54
<u>CLCHNAME</u>	Cluster-sender channel name	Character value, *NONE, *SYSDFTQ	Optional, Positional 55
<u>IMGRCOVQ</u>	Queue object attribute	*SAME, *NO, *YES, *QMGR	Optional, Positional 57

### Queue name (QNAME)

Specifies the name of the queue definition. Queue names must be unique. If a queue definition with this name already exists, you must specify REPLACE(\*YES).

The name can contain up to 48 characters.

**Note:** The field length is 48 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

The possible values are:

#### queue-name

Specify the name of the new queue.

### Queue type (QTYPE)

Specifies the type of queue that is to be created.

If the queue already exists, REPLACE(\*YES) must be specified, and the value specified by QTYPE must be the type of the existing queue.

The possible values are:

#### \*ALS

An alias queue.

#### \*LCL

A local queue.

#### \*RMT

A remote queue.

#### \*MDL

A model queue.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

Specify the name of the queue manager.

### Replace (REPLACE)

Specifies whether the new queue will replace an existing queue definition with the same name and type.

The possible values are:

**\*NO**

Do not replace the existing queue. The command fails if the named queue already exists.

**\*YES**

Replace the existing queue definition with the attributes of the FROMQ and the specified attributes.

The command will fail if an application has the Queue open or the USAGE attribute is changed.

**Note:** If the queue is a local queue, and a queue with the same name already exists, any messages already on that queue are retained.

**Text 'description' (TEXT)**

Specifies text that briefly describes the queue definition.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*BLANK**

The text is set to a blank string.

**description**

Specify no more than 64 characters enclosed in apostrophes.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Put enabled (PUTENBL)**

Specifies whether messages can be put on the queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

Messages cannot be added to the queue.

**\*YES**

Messages can be added to the queue by authorized applications.

**Default message priority (DFTPTY)**

Specifies the default priority of messages put on the queue.

The possible values are:

**\*SYSDFTQ**

The value of this attribute taken from the system default queue of the specified type.

**priority-value**

Specify a value ranging from 0 through 9.

**Default message persistence (DFTMSGPST)**

Specifies the default for message-persistence on the queue. Message persistence determines whether messages are preserved across restarts of the queue manager.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

By default, messages are lost across a restart of the queue manager.

**\*YES**

By default, messages are preserved across a restart of the queue manager.

**Process name (PRCNAME)**

Specifies the local name of the MQ process that identifies the application that should be started when a trigger event occurs.

The process does not have to be available when the queue is created, but it must be available for a trigger event to occur.

The possible values are:

**\*SYSDFTQ**

The value of this attribute taken from the system default queue of the specified type.

**\*NONE**

No process is specified.

**process-name**

Specify the name of the process.

**Triggering enabled (TRGENBL)**

Specifies whether trigger messages are written to the initiation queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

Do not write trigger messages to the initiation queue.

**\*YES**

Triggering is active; trigger messages are written to the initiation queue.

**Get enabled (GETENBL)**

Specifies whether applications are to be permitted to get messages from this queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

Applications cannot retrieve messages from the queue.

**\*YES**

Suitably authorized applications can retrieve messages from the queue.

**Sharing enabled (SHARE)**

Specifies whether multiple instances of applications can open this queue for input.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is from the system default queue of the specified type.

**\*NO**

Only a single application instance can open the queue for input.

**\*YES**

More than one application instance can open the queue for input.

**Default share option (DFTSHARE)**

Specifies the default share option for applications opening this queue for input.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

The open request is for exclusive use of the queue for input.

**\*YES**

The open request is for shared use of the queue for input.

**Message delivery sequence (MSGDLYSEQ)**

Specifies the message delivery sequence.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*PTY**

Messages are delivered in first-in-first-out (FIFO) order within priority.

**\*FIFO**

Messages are delivered in FIFO order regardless of priority.

**Harden backout count (HDNBKTCNT)**

Specifies whether the count of backed out messages should be saved (hardened) across restarts of the message queue manager.

**Note:** On IBM MQ for IBM i the count is ALWAYS hardened, regardless of the setting of this attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

The backout count is not hardened.

**\*YES**

The backout count is hardened.

**Trigger type (TRGTYPE)**

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*FIRST**

When the number of messages on the queue goes from zero to one.



**\*ALL**

Every time a message arrives on the queue.

**\*DEPTH**

When the number of messages on the queue equals the value of the TRGDEPTH attribute.

**\*NONE**

No trigger messages are written.

**Trigger depth (TRGDEPTH)**

Specifies, for TRIGTYPE(\*DEPTH), the number of messages that initiate a trigger message to the initiation queue.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**depth-value**

Specify a value ranging from 1 through 999999999.

**Trigger message priority (TRGMSGPTY)**

Specifies the minimum priority that a message must have before it can produce, or be counted for, a trigger event.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**priority-value**

Specify a value ranging from 0 through 9.

**Trigger data (TRGDATA)**

Specifies up to 64 characters of user data that the queue manager includes in the trigger message. This data is made available to the monitoring application that processes the initiation queue and to the application started by the monitor.

**Note:** An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NONE**

No trigger data is specified.

**trigger-data**

Specify up to 64 characters enclosed in apostrophes. For a transmission queue you can use this parameter to specify the name of the channel to be started.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Retention interval (RTNITV)**

Specifies the retention interval. This interval is the number of hours for which the queue might be needed, based on the date and time when the queue was created.

This information is available to a housekeeping application or an operator and can be used to determine when a queue is no longer required.

**Note:** The message queue manager does not delete queues, nor does it prevent your queues from being deleted if their retention interval has not expired. It is your responsibility to take any required action.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**interval-value**

Specify a value ranging from 0 through 999999999.

### **Maximum queue depth (MAXDEPTH)**

Specifies the maximum number of messages allowed on the queue. However, other factors can cause the queue to be treated as full; for example, it appears to be full if there is no storage available for a message.

**Note:** If this value is subsequently reduced by using the CHGMQMQ command, any messages that are on the queue remain intact even if they cause the new maximum to be exceeded.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**depth-value**

Specify a value ranging from 0 through 999999999.

### **Maximum message length (MAXMSGLEN)**

Specifies the maximum length for messages on the queue.

**Note:** If this value is subsequently reduced by using the CHGMQMQ command, any messages that are on the queue remain intact even if they exceed the new maximum length.

Applications might use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore change the value only if you know this will not cause an application to operate incorrectly.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified queue type.

**length-value**

Specify a value ranging from 0 through 104 857 600.

### **Backout threshold (BKTTHLD)**

Specifies the backout threshold.

Applications running inside of WebSphere Application Server and those that use the IBM MQ Application Server Facilities will use this attribute to determine if a message should be backed out. For all other applications, apart from allowing this attribute to be queried, the queue manager takes no action based on the value of the attribute.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified queue type.

**threshold-value**

Specify a value ranging from 0 through 999999999.

## Backout requeue name (BKTQNAME)

Specifies the backout-queue name.

Applications running inside of WebSphere Application Server and those that use the IBM MQ Application Server Facilities will use this attribute to determine where messages that have been backed out should go. For all other applications, apart from allowing this attribute to be queried, the queue manager takes no action based on the value of the attribute.

The possible values are:

### **\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified queue type.

### **\*NONE**

No backout queue is specified.

### **backout-queue-name**

Specify the backout queue name.

## Initiation queue (INITQNAME)

Specifies the name of the initiation queue.

**Note:** The initiation queue must be on the same instance of a message queue manager.

The possible values are:

### **\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified queue type.

### **\*NONE**

No initiation queue is specified.

### **initiation-queue-name**

Specify the initiation queue name.

## Usage (USAGE)

Specifies whether the queue is for normal usage, or for transmitting messages to a remote message queue manager.

The possible values are:

### **\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified queue type.

### **\*NORMAL**

Normal usage (the queue is not a transmission queue)

### **\*TMQ**

The queue is a transmission queue that is used to hold messages destined for a remote message queue manager. If the queue is intended for use in situations where a transmission queue name is not explicitly specified, the queue name must be the same as the name of the remote message queue manager. For further information, see the IBM MQ Intercommunication.

## Definition type (DFNTYPE)

Specifies the type of dynamic queue definition that is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor.

**Note:** This parameter only applies to a model queue definition.

The possible values are:

### **\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*TEMPDYN**

Creates a temporary dynamic queue. Do not specify with a DEFMSGPST value of \*YES.

**\*PERMDYN**

Creates a permanent dynamic queue.

**Target object (TGTQNAME)**

Specifies the name of the target object for which this queue is an alias.

The object can be a local or remote queue, a topic or a message queue manager.

Do not leave this field blank. If you do so, it is possible that you will create an alias queue, that has to be subsequently modified, by the addition of a TGTNAME.

When a message queue manager name is specified, it identifies the message queue manager that handles the messages posted to the alias queue. You can specify either the local message queue manager or a transmission queue name.

**Note:** The target object does not need to exist at this time but it must exist when a process attempts to open the alias queue.

The possible values are:

**\*SYSDFTQ**

The name of the target object is taken from the SYSTEM.DEFAULT.ALIAS.QUEUE.

**target-object-name**

Specify the name of the target object.

**Remote queue (RMTQNAME)**

Specifies the name of the remote queue. That is, the local name of the remote queue as defined on the queue manager specified by RMTMQMNAME.

If this definition is used for a queue manager alias definition, RMTQNAME must be blank when the open occurs.

If this definition is used for a reply-to alias, this name is the name of the queue that is to be the reply-to queue.

The possible values are:

**\*SYSDFTQ**

The name of the remote queue is taken from the SYSTEM.DEFAULT.REMOTE.QUEUE.

**\*NONE**

No remote-queue name is specified (that is, the name is blank). This can be used if the definition is a queue manager alias definition.

**remote-queue-name**

Specify the name of the queue at the remote queue manager.

**Note:** The name is not checked to ensure that it contains only those characters normally allowed for queue names

**Remote Message Queue Manager (RMTMQMNAME)**

Specifies the name of the remote queue manager on which the queue RMTQNAME is defined.

If an application opens the local definition of a remote queue, RMTMQMNAME must not be the name of the connected queue manager. If TMQNAME is blank there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a queue manager alias, RMTMQMNAME is the name of the queue manager, which can be the name of the connected queue manager. Otherwise, if TMQNAME is blank, when the

queue is opened there must be a local queue of this name, with USAGE(\*TMQ) specified, which is to be used as the transmission queue.

If this definition is used for a reply-to alias, this name is the name of the queue manager that is to be the reply-to queue manager.

The possible values are:

**\*SYSDFTQ**

The name of the remote queue manager is taken from the SYSTEM.DEFAULT.REMOTE.QUEUE.

**remote-queue-manager-name**

Specify the name of the remote queue manager.

**Note:** Ensure this name contains only those characters normally allowed for queue manager names.

## Transmission queue (TMQNAME)

Specifies the local name of the transmission queue to be used for messages destined for the remote queue, for either a remote queue or for a queue manager alias definition.

If TMQNAME is blank, a queue with the same name as RMTMQMNAME is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue manager alias and RMTMQMNAME is the name of the connected queue manager.

It is also ignored if the definition is used as a reply-to queue alias definition.

The possible values are:

**\*SYSDFTQ**

The transmission queue name is taken from the SYSTEM.DEFAULT.REMOTE.QUEUE.

**\*NONE**

No specific transmission queue name is defined for this remote queue. The value of this attribute is set to all blanks.

**transmission-queue-name**

Specify the transmission queue name.

## Queue depth high threshold (HIGHTHLD)

Specifies the threshold against which the queue depth is compared to generate a queue depth high event.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**threshold-value**

Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

## Queue depth low threshold (LOWTHLD)

Specifies the threshold against which the queue depth is compared to generate a queue depth low event.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**threshold-value**

Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

### **Queue full events enabled (FULLEVT)**

Specifies whether queue full events are generated.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

Queue Full events are not generated.

**\*YES**

Queue Full events are generated.

### **Queue high events enabled (HIGHEVT)**

Specifies whether queue depth high events are generated.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

Queue Depth High events are not generated.

**\*YES**

Queue Depth High events are generated.

### **Queue low events enabled (LOWEVT)**

Specifies whether queue depth low events are generated.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

Queue Depth Low events are not generated.

**\*YES**

Queue Depth Low events are generated.

### **Service interval (SRVITV)**

Specifies the service interval. This interval is used for comparison to generate service interval high and service interval OK events.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**interval-value**

Specify a value ranging from 0 through 999999999. The value is in units of milliseconds.

### **Service interval events (SRVEVT)**

Specifies whether service interval high or service interval OK events are generated.

A service interval high event is generated when a check indicates that no messages have been retrieved from the queue for the time indicated by the SRVITV parameter as a minimum.

A service interval OK event is generated when a check indicates that messages have been retrieved from the queue within the time indicated by the SRVITV parameter.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*HIGH**

Service Interval High events are generated.

**\*OK**

Service Interval OK events are generated.

**\*NONE**

No service interval events are generated.

### **Distribution list support (DISTLIST)**

Specifies whether the queue supports distribution lists.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NO**

Distribution Lists are not supported.

**\*YES**

Distribution Lists are supported.

### **Cluster Name (CLUSTER)**

The name of the cluster to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are already open.

This parameter cannot be set for dynamic, transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx or SYSTEM.COMMAND.xx queues.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**cluster-name**

Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

### **Cluster Name List (CLUSNL)**

The name of the namelist which specifies a list of clusters to which the queue belongs. Changes to this parameter do not affect instances of the queue that are already open.

This parameter cannot be set for dynamic, transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx or SYSTEM.COMMAND.xx queues.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**namelist-name**

The name of the namelist that specifies a list of clusters to which the queue belongs.

### **Default Binding (DEFBIND)**

Specifies the binding to be used when the application specifies MQOO\_BIND\_AS\_Q\_DEF on the MQOPEN call and the queue is a cluster queue.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*OPEN**

The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

**\*NOTFIXED**

The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using MQPUT and to change that selection subsequently if necessary.

The MQPUT1 call always behaves as if NOTFIXED had been specified.

**\*GROUP**

When the queue is opened, the queue handle is bound to a specific instance of the cluster queue for as long as there are messages in a message group. All messages in a message group are allocated to the same destination instance.

### **Cluster Workload Rank (CLWLRANK)**

Specifies the cluster workload rank of the queue.

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**cluster-workload-rank**

Specify a value ranging from 0 through 9.

### **Cluster Workload Priority (CLWLPRTY)**

Specifies the cluster workload priority of the queue.

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**cluster-workload-priority**

Specify a value ranging from 0 through 9.

### **Cluster workload queue use (CLWLUSEQ)**

Specifies the behavior of an MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel then this attribute does not apply.

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*QMGR**

The value is inherited from the Queue Manager CLWLUSEQ attribute.

**\*LOCAL**

The local queue will be the sole target of the MQPUT.

**\*ANY**

The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

### **Queue Monitoring (MONQ)**

Controls the collection of Online Monitoring Data.

Online Monitoring Data is not collected when the queue manager attribute MONQ is set to \*NONE.

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.



**\*QMGR**

The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONQ.

**\*OFF**

Online Monitoring Data collection for this queue is disabled.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

**Queue Statistics (STATQ)**

Controls the collection of statistics data.

Online monitoring data is not collected when the queue manager attribute STATQ is set to \*NONE.

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*QMGR**

Statistics data collection is based upon the setting of the queue manager attribute STATQ.

**\*OFF**

Statistics data collection for this queue is disabled.

**\*ON**

Statistics data collection is enabled for this queue.

**Queue Accounting (ACCTQ)**

Controls the collection of accounting data.

Accounting data is not collected when the queue manager attribute ACCTQ is set to \*NONE.

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*QMGR**

Accounting data collection is based upon the setting of the queue manager attribute ACCTQ.

**\*OFF**

Accounting data collection for this queue is disabled.

**\*ON**

Accounting data collection is enabled for this queue.

**Non Persistent Message Class (NPMCLASS)**

Specifies the level of reliability for non-persistent messages put to this queue.

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NORMAL**

Non-persistent messages put to this queue are only lost following a failure, or a queue manager shutdown. Non-persistent message put to this queue are discarded in the event of a queue manager restart.

**\*HIGH**

Non-persistent messages put to this queue are not discarded in the event of a queue manager restart. Non-persistent messages put to this queue may still be lost in the event of a failure.

## Message Read Ahead (MSGREADAHD)

Specifies whether nonpersistent messages are sent to the client ahead of an application requesting them.

The possible values are:

### **\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

### **\*DISABLED**

Read ahead is disabled for this queue. Messages are not sent to the client ahead of an application requesting them regardless of whether read ahead is requested by the client application.

### **\*NO**

Non-persistent messages are not sent to the client ahead of an application requesting them. A maximum of one non-persistent message can be lost if the client ends abnormally.

### **\*YES**

Non-persistent messages are sent to the client ahead of an application requesting them. Non-persistent messages can be lost if the client ends abnormally or if the client application does not consume all the messages it is sent.

## Default Put Response (DFTPUTRESP)

The default put response type (DFTPUTRESP) attribute specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO\_RESPONSE\_AS\_Q\_DEF option.

The possible values are:

### **\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

### **\*SYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application. This is the default value supplied with IBM MQ, but your installation might have changed it.

### **\*ASYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application; but an improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

## Property Control (PROPCTL)

Specifies what happens to properties of messages that are retrieved from queues using the MQGET call when the MQGMO\_PROPERTIES\_AS\_Q\_DEF option is specified.

The possible values are:

### **\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

### **\*COMPAT**

If the message contains a property with a prefix of mcd . , jms . , usr . or mqext . then all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

### **\*NONE**

All properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

**\*ALL**

All properties of the message, except those contained in the message descriptor (or extension), are contained in one or more MQRFH2 headers in the message data.

**\*FORCE**

Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

**\*V6COMPAT**

When set, \*V6COMPAT must be set both on one of the queue definitions resolved by MQPUT and one of the queue definitions resolved by MQGET. It must also be set on any other intervening transmission queues. It causes an MQRFH2 header to be passed unchanged from the sending application to the receiving application. It overrides other settings of **PROPCTL** found in a queue name resolution chain. If the property is set on a cluster queue, the setting is not cached locally on other queue managers. You must set \*V6COMPAT on an alias queue that resolves to the cluster queue. Define the alias queue on the same queue manager that the putting application is connected to.

**Target Type (TARGTYPE)**

Specifies the type of object to which the alias resolves.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*QUEUE**

Queue object.

**\*TOPIC**

Topic object.

**Custom attribute (CUSTOM)**

This attribute is reserved for the configuration of new features before separate attributes have been introduced. This description will be updated when features using this attribute are introduced. At the moment there are no meaningful values for *CUSTOM*, so leave it empty.

The possible values are:

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*BLANK**

The text is set to a blank string.

**custom**

Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs must have the form NAME (VALUE) and be specified in uppercase. Single quotes must be escaped with another single quote.

**CLCHNAME**

This parameter is supported only on transmission queues.

**\*SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

**\*NONE**

The attribute is removed.

**custom**

Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs must have the form NAME (VALUE) and be specified in uppercase. Single quotes must be escaped with another single quote.

By specifying asterisks, "\*", in **ClusterChannelName**, you can associate a transmission queue with a set of cluster-sender channels. The asterisks can be at the beginning, end, or any number of places in the middle of the channel name string. **ClusterChannelName** is limited to a length of 20 characters: MQ\_CHANNEL\_NAME\_LENGTH.

## IMGRCOVQ

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used.

The possible values are:

### \*SAME

The attribute is unchanged.

### \*YES

These queue objects are recoverable.

### \*NO

The "RCDMQMIMG (Record MQ Object Image)" on page 1857 and "RCRMQMOBJ (Re-create MQ Object)" on page 1859 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

### \*QMGR

If you specify \*QMGR, and the **IMGRCOVQ** attribute for the queue manager specifies \*YES, these queue objects are recoverable.

If you specify \*QMGR and the **IMGRCOVQ** attribute for the queue manager specifies \*NO, the "RCDMQMIMG (Record MQ Object Image)" on page 1857 and "RCRMQMOBJ (Re-create MQ Object)" on page 1859 commands are not permitted for these objects, and automatic media images, if enabled, are not written for these objects.

## CRTMQMSUB (Create MQ Subscription)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Create MQ Subscription (CRTMQMSUB) command creates a new MQ subscription, specifying those attributes that are different from the default.

## Parameters

Keyword	Description	Choices	Notes
<u>SUBNAME</u>	Subscription name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Key, Positional 2
<u>REPLACE</u>	Replace	*NO, *YES	Optional, Key, Positional 3
<u>TOPICSTR</u>	Topic string	Character value, *NONE, *SYSDFTSUB	Optional, Positional 4
<u>TOPICOBJ</u>	Topic object	Character value, *NONE, *SYSDFTSUB	Optional, Positional 5
<u>DEST</u>	Destination	Character value, *SYSDFTSUB	Optional, Positional 6

Table 249. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>DESTMQM</u>	Destination Queue Manager	Character value, *NONE, <b>*SYSDFTSUB</b>	Optional, Positional 7
<u>DESTCRLID</u>	Destination Correlation Id	Character value, *NONE, <b>*SYSDFTSUB</b>	Optional, Positional 8
<u>PUBACCT</u>	Publish Accounting Token	Character value, *CURRENT, <b>*SYSDFTSUB</b>	Optional, Positional 9
<u>PUBAPPID</u>	Publish Application Id	Character value, *NONE, <b>*SYSDFTSUB</b>	Optional, Positional 10
<u>SUBUSER</u>	Subscription User Id	Character value, *CURRENT, <b>*SYSDFTSUB</b>	Optional, Positional 11
<u>USERDATA</u>	Subscription User Data	Character value, *NONE, <b>*SYSDFTSUB</b>	Optional, Positional 12
<u>SELECTOR</u>	Selector String	Character value, *NONE, <b>*SYSDFTSUB</b>	Optional, Positional 13
<u>PSPROP</u>	PubSub Property	<b>*SYSDFTSUB</b> , *NONE, *COMPAT, *RFH2, *MSGPROP	Optional, Positional 14
<u>DESTCLASS</u>	Destination Class	<b>*SYSDFTSUB</b> , *MANAGED, *PROVIDED	Optional, Positional 15
<u>SUBSCOPE</u>	Subscription Scope	<b>*SYSDFTSUB</b> , *ALL, *QMGR	Optional, Positional 16
<u>VARUSER</u>	Variable User	<b>*SYSDFTSUB</b> , *ANY, *FIXED	Optional, Positional 17
<u>REQONLY</u>	Request Publications	<b>*SYSDFTSUB</b> , *YES, *NO	Optional, Positional 18
<u>PUBPTY</u>	Publish Priority	0-9, <b>*SYSDFTSUB</b> , *ASPUB, *ASQDEF	Optional, Positional 19
<u>WSHEMA</u>	Wildcard Schema	<b>*SYSDFTSUB</b> , *TOPIC, *CHAR	Optional, Positional 20
<u>EXPIRY</u>	Expiry Time	0-999999999, <b>*SYSDFTSUB</b> , *UNLIMITED	Optional, Positional 21

### Subscription name (SUBNAME)

The name of the new MQ subscription to be created.

The possible values are:

#### subscription-name

Specify a maximum of 256 bytes for the subscription name.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

### Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

**\*DFT**

Use the default Queue Manager.

**queue-manager-name**

The name of a Queue Manager.

**Replace (REPLACE)**

If a subscription with the same name already exists, this specifies whether it is replaced.

The possible values are:

**\*NO**

This subscription does not replace any existing subscription with the same name or subscription identifier. The command fails if the subscription already exists.

**\*YES**

Replace the existing subscription. If there is no subscription with the same name or subscription identifier, a new subscription is created.

**Topic string (TOPICSTR)**

Specifies the topic string associated with this subscription.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**topic-string**

Specify a maximum of 256 bytes for the topic string.

**Note:** Topic strings of greater than 256 bytes can be specified using MQSC.

**Topic object (TOPICOBJ)**

Specifies the topic object associated with this subscription.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**topic-object**

Specify the name of the topic object.

**Destination (DEST)**

Specifies the destination queue for messages published to this subscription.

The possible values are:

**destination-queue**

Specify the name of the destination queue.

**Destination Queue Manager (DESTMQM)**

Specifies the destination queue manager for messages published to this subscription.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**destination-queue-manager**

Specify the name of the destination queue manager.

## **Destination Correlation Id (DESTCRLID)**

Specifies the correlation identifier for messages published to this subscription.

The possible values are:

### **\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

### **destination-correlation-identifier**

Specify the 48 character hexadecimal string representing the 24 byte correlation identifier.

## **Publish Accounting Token (PUBACCT)**

Specifies the accounting token for messages published to this subscription.

The possible values are:

### **\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

### **\*NONE**

Messages are placed on the destination with an accounting token of MQACT\_NONE.

### **publish-accounting-token**

Specify the 64 character hexadecimal string representing the 32 byte publish accounting token.

## **Publish Application Id (PUBAPPID)**

Specifies the publish application identity for messages published to this subscription.

The possible values are:

### **\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

### **\*NONE**

No publish application identifier is specified.

### **publish-application-identifier**

Specify the publish application identifier.

## **Subscription User Id (SUBUSER)**

Specifies the user profile that owns this subscription.

The possible values are:

### **\*SAME**

The attribute is unchanged.

### **\*CURRENT**

The current user profile is the owner of the new subscription.

### **user-profile**

Specify the user profile.

## **Subscription User Data (USERDATA)**

Specifies the user data associated with the subscription.

The possible values are:

### **\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

### **\*NONE**

No user data is specified.

**user-data**

Specify a maximum of 256 bytes for user data.

**Note:** User data of greater than 256 bytes can be specified using MQSC.

**Selector String (SELECTOR)**

Specifies the SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*NONE**

No selection string is specified.

**selection-string**

Specify a maximum of 256 bytes for selection string.

**Note:** Selection strings of greater than 256 bytes can be specified using MQSC.

**PubSub Property (PSPROP)**

Specifies the manner in which publish / subscribe related message properties are added to messages sent to this subscription.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*NONE**

Publish / subscribe properties are not added to the message.

**\*COMPAT**

Publish / subscribe properties are added to the message to maintain compatibility with V6 Publish / Subscribe.

**\*RFH2**

Publish / subscribe properties are added to the message within an RFH 2 header.

**\*MSGPROP**

Publish / subscribe properties are added as message properties.

**Destination Class (DESTCLASS)**

Specifies whether this is a managed subscription.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*MANAGED**

The destination is managed.

**\*PROVIDED**

The destination is a queue.

**Subscription Scope (SUBSCOPE)**

Specifies whether this subscription should be forwarded (as a proxy subscription) to other brokers, so that the subscriber will receive messages published at those other brokers.

The possible values are:



**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*ALL**

The subscription will be forwarded to all queue managers directly connected via a publish / subscribe collective or hierarchy.

**\*QMGR**

The subscription will only forward messages published on the topic within this queue manager.

**Variable User (VARUSER)**

Specifies whether user profiles other than the creator of the subscription can connect to it (subject to topic and destination authority checks).

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*ANY**

Any user profiles can connect to the subscription.

**\*FIXED**

Only the user profile that created the subscription can connect to it.

**Request Publications (REQONLY)**

Specifies whether the subscriber will poll for updates via MQSUBRQ API, or whether all publications are delivered to this subscription.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*YES**

Publications are only delivered to this subscription in response to an MQSUBRQ API.

**\*NO**

All publications on the topic are delivered to this subscription.

**Publish Priority (PUBPTY)**

Specifies the priority of the message sent to this subscription.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*ASPUB**

The priority of the message sent to this subscription is taken from that supplied in the published message.

**\*ASQDEF**

The priority of the message sent to this subscription is taken from the default priority of the queue defined as the destination.

**priority-value**

Specify a priority ranging from 0 through 9.

**Wildcard Schema (WSHEMA)**

Specifies the schema to be used when interpreting wildcard characters in the topic string.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*TOPIC**

Wildcard characters represent portions of the topic hierarchy.

**\*CHAR**

Wildcard characters represent portions of strings.

**Expiry Time (EXPIRY)**

Specifies the expiry time of the subscription. After a subscription's expiry time has elapsed, it becomes eligible to be discarded by the queue manager and will receive no further publications.

The possible values are:

**\*SYSDFTSUB**

The value of this attribute is taken from the system default subscription.

**\*UNLIMITED**

The subscription does not expire.

**expiry-time**

Specify an expiry time in tenths of a second ranging from 0 through 999999999.


**CRTMQMSVC (Create MQ Service)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Create MQ Service (CRTMQMSVC) command creates a new MQ service definition, specifying those attributes that are to be different from the default.

**Parameters**

<i>Table 250. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>SVCNAME</u>	Service name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 2
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 3
<u>TEXT</u>	Text 'description'	<i>Character value</i> , *BLANK, <b>*SYSDFTSVC</b>	Optional, Positional 4
<u>STRCMD</u>	Start program	Single values: <b>*SYSDFTSVC</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 5
	Qualifier 1: Start program	Name	
	Qualifier 2: Library	Name	
<u>STRARG</u>	Start program arguments	<i>Character value</i> , *BLANK, <b>*SYSDFTSVC</b>	Optional, Positional 6

Table 250. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>ENDCMD</u>	End program	Single values: <b>*SYSDFTSVC</b> , *NONE Other values: <i>Qualified object name</i>	Optional, Positional 7
	Qualifier 1: End program	Name	
	Qualifier 2: Library	Name	
<u>ENDARG</u>	End program arguments	<i>Character value</i> , *BLANK, <b>*SYSDFTSVC</b>	Optional, Positional 8
<u>STDOUT</u>	Standard output	<i>Character value</i> , *BLANK, <b>*SYSDFTSVC</b>	Optional, Positional 9
<u>STDERR</u>	Standard error	<i>Character value</i> , *BLANK, <b>*SYSDFTSVC</b>	Optional, Positional 10
<u>TYPE</u>	Service type	<b>*SYSDFTSVC</b> , *CMD, *SVR	Optional, Positional 11
<u>CONTROL</u>	Service control	<b>*SYSDFTSVC</b> , *MANUAL, *QMGR, *STARTONLY	Optional, Positional 12

### Service name (SVCNAME)

The name of the new MQ service definition.

The possible values are:

#### **service-name**

Specify the name of the service definition. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

#### **queue-manager-name**

The name of a message queue manager.

### Replace (REPLACE)

If a service definition with the same name already exists, this specifies whether it is replaced.

The possible values are:

#### **\*NO**

This definition does not replace any existing service definition with the same name. The command fails if the named service definition already exists.

#### **\*YES**

Replace the existing service definition. If there is no definition with the same name, a new definition is created.

## Text 'description' (TEXT)

Specifies text that briefly describes the service definition.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

### **\*SYSDFTSVC**

The value of this attribute is taken from the system default service.

### **\*BLANK**

The text is set to a blank string.

### **description**

Specify the new descriptive information.

## Start program (STRCMD)

The name of the program to run.

The possible values are:

### **\*SYSDFTSVC**

The value of this attribute is taken from the system default service.

### **start-command**

The name of the start command executable.

## Start program arguments (STRARG)

The arguments passed to the program at startup.

The possible values are:

### **\*SYSDFTSVC**

The value of this attribute is taken from the system default service.

### **\*BLANK**

No arguments are passed to the start command.

### **start-command-arguments**

The arguments passed to the start command.

## End program (ENDCMD)

The name of the executable to run when the service is requested to stop.

The possible values are:

### **\*SYSDFTSVC**

The value of this attribute is taken from the system default service.

### **\*BLANK**

No end command is executed.

### **end-command**

The name of the end command executable.

## End program arguments (ENDARG)

The arguments passed to the end program when the service is requested to stop.

The possible values are:

### **\*SYSDFTSVC**

The value of this attribute is taken from the system default service.

**\*BLANK**

No arguments are passed to the end command.

**end-command-arguments**

The arguments passed to the end command.

**Standard output (STDOUT)**

The path to a file to which the standard output of the service program is redirected.

The possible values are:

**\*SYSDFTSVC**

The value of this attribute is taken from the system default service.

**\*BLANK**

The standard output is discarded.

**stdout-path**

The standard output path.

**Standard error (STDERR)**

The path to a file to which the standard error of the service program is redirected.

The possible values are:

**\*SYSDFTSVC**

The value of this attribute is taken from the system default service.

**\*BLANK**

The standard error is discarded.

**stderr-path**

The standard error path.

**Service type (TYPE)**

Mode in which to run service.

The possible values are:

**\*SYSDFTSVC**

The value for this attribute is taken from the system default service.

**\*CMD**

When started the command is executed but no status is collected or displayed.

**\*SVR**

The status of the executable started will be monitored and displayed.

**Service control (CONTROL)**

Whether the service should be started automatically at queue manager start.

The possible values are:

**\*SYSDFTSVC**

The value for this attribute is taken from the system default service.

**\*MANUAL**

The service will not be automatically started or stopped.

**\*QMGR**

The service will be started and stopped as the queue manager is started and stopped.

**\*STARTONLY**

The service will be started as the queue manager is started, but will not be requested to stop when the queue manager is stopped.


**CRTMQMTOPT (Create MQ Topic)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Create MQ Topic (CRTMQMTOPT) command creates a new MQ topic object, specifying those attributes that are different from the default.

**Parameters**

<i>Table 251. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>TOPNAME</u>	Topic name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Key, Positional 2
<u>REPLACE</u>	Replace	<b>*NO</b> , *YES	Optional, Positional 3
<u>TEXT</u>	Text 'description'	<i>Character value, *BLANK, *SYSDFTTOP</i>	Optional, Positional 4
<u>TOPICSTR</u>	Topic string	<i>Character value, *BLANK, *SYSDFTTOP</i>	Optional, Positional 5
<u>DURSUB</u>	Durable subscriptions	<b>*SYSDFTTOP</b> , *ASPARENT, *YES, *NO	Optional, Positional 6
<u>MGDDURMDL</u>	Durable model queue	<i>Character value, *NONE, *SYSDFTTOP</i>	Optional, Positional 7
<u>MGDNDURMDL</u>	Non-durable model queue	<i>Character value, *NONE, *SYSDFTTOP</i>	Optional, Positional 8
<u>PUBENBL</u>	Publish	<b>*SYSDFTTOP</b> , *ASPARENT, *YES, *NO	Optional, Positional 9
<u>SUBENBL</u>	Subscribe	<b>*SYSDFTTOP</b> , *ASPARENT, *YES, *NO	Optional, Positional 10
<u>DFTPTY</u>	Default message priority	0-9, <b>*SYSDFTTOP</b> , *ASPARENT	Optional, Positional 11
<u>DFTMSGPST</u>	Default message persistence	<b>*SYSDFTTOP</b> , *ASPARENT, *YES, *NO	Optional, Positional 12
<u>DFTPUTRESP</u>	Default Put Response	<b>*SYSDFTTOP</b> , *ASPARENT, *SYNC, *ASYN	Optional, Positional 13
<u>WILDCARD</u>	Wildcard behavior	<b>*SYSDFTTOP</b> , *PASSTHRU, *BLOCK	Optional, Positional 14

Table 251. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>PMSGDLV</u>	Persistent message delivery	<b>*SYSDFTTOP</b> , *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 15
<u>NPMSGDLV</u>	Non-persistent message deliver	<b>*SYSDFTTOP</b> , *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 16
<u>CUSTOM</u>	Custom attribute	<i>Character value</i> , *BLANK, <b>*SYSDFTTOP</b>	Optional, Positional 17

### Topic name (TOPNAME)

The name of the new MQ topic object to be created.

The possible values are:

#### topic-name

Specify the name of the new MQ topic object. The name can contain up to 48 characters.

### Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

#### \*DFT

Use the default Queue Manager.

#### queue-manager-name

The name of a Queue Manager.

### Replace (REPLACE)

If a topic object with the same name already exists, this specifies whether it is replaced.

The possible values are:

#### \*NO

This object does not replace any existing topic object with the same name. The command fails if the named topic object already exists.

#### \*YES

Replace the existing topic object. If there is no object with the same name, a new object is created.

### Text 'description' (TEXT)

Specifies text that briefly describes the topic object.

**Note:** The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

#### \*SYSDFTTOP

The value of this attribute is taken from the system default topic.

#### \*BLANK

The text is set to a blank string.

#### description

Specify the new descriptive information.

## Topic string (TOPICSTR)

Specifies the topic string represented by this topic object definition.

The possible values are:

### **topic-string**

Specify a maximum of 256 bytes for the topic string.

**Note:** Topic strings of greater than 256 bytes can be specified using MQSC.

## Durable subscriptions (DURSUB)

Specifies whether applications are permitted to make durable subscriptions on this topic.

The possible values are:

### **\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

### **\*ASPARENT**

Whether durable subscriptions can be made on this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

### **\*YES**

Durable subscriptions can be made on this topic.

### **\*NO**

Durable subscriptions cannot be made on this topic.

## Durable model queue (MGDDURMDL)

Specifies the name of the model queue to be used for durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

### **\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

### **durable-model-queue**

Specify the name of the model queue.

## Non-durable model queue (MGDNDURMDL)

Specifies the name of the model queue to be used for non-durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

### **\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

### **non-durable-model-queue**

Specify the name of the model queue.

## Publish (PUBENBL)

Specifies whether messages can be published to the topic.

The possible values are:

### **\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

### **\*ASPARENT**

Whether messages can be published to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.



**\*YES**

Messages can be published to the topic.

**\*NO**

Messages cannot be published to the topic.

**Subscribe (SUBENBL)**

Specifies whether applications are to be permitted to subscribe to this topic.

The possible values are:

**\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

**\*ASPARENT**

Whether applications can subscribe to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*YES**

Subscriptions can be made to this topic.

**\*NO**

Applications cannot subscribe to this topic.

**Default message priority (DFTPTY)**

Specifies the default priority of messages published to the topic.

The possible values are:

**\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

**\*ASPARENT**

The default priority is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**priority-value**

Specify a value ranging from 0 through 9.

**Default message persistence (DFTMSGPST)**

Specifies the message persistence to be used when applications specify the MQPER\_PERSISTENCE\_AS\_TOPIC\_DEF option.

The possible values are:

**\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

**\*ASPARENT**

The default persistence is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*YES**

Messages on this queue survive a restart of the queue manager.

**\*NO**

Messages on this queue are lost across a restart of the queue manager.

**Default Put Response (DFTPUTRESP)**

Specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO\_RESPONSE\_AS\_Q\_DEF option.

The possible values are:

**\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

**\*ASPARENT**

The default response type is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*SYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application.

**\*ASYNC**

Specifying this value ensures that the put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application. An improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

**Wildcard behavior (WILDCARD)**

Specifies the behavior of wildcard subscriptions with respect to this topic.

The possible values are:

**\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

**\*PASSTHRU**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will receive publications made to this topic and to topic strings more specific than this topic.

**\*BLOCK**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will not receive publications made to this topic or to topic strings more specific than this topic.

**Persistent message delivery (PMSGDLV)**

Specifies the delivery mechanism for persistent messages published to this topic.

The possible values are:

**\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

**\*ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*ALL**

Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**\*ALLDUR**

Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

**\*ALLAVAIL**

Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

**Non-persistent message delivery (NPMSGDLV)**

Specifies the delivery mechanism for non-persistent messages published to this topic.

The possible values are:

**\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

**\*ASPARENT**

The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*ALL**

Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

**\*ALLDUR**

Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

**\*ALLAVAIL**

Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

### Custom attribute (CUSTOM)

This attribute is reserved for the configuration of new features before separate attributes have been introduced. This description will be updated when features using this attribute are introduced. At the moment there are no meaningful values for *CUSTOM*, so leave it empty.

The possible values are:

**\*SYSDFTTOP**

The value of this attribute is taken from the system default topic.

**\*BLANK**

The text is set to a blank string.

**custom**

Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name-value pairs must have the form NAME (VALUE) and be specified in uppercase. Single quotes must be escaped with another single quote.

## CVTMQMDTA (Convert MQ Data Type)

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Convert MQ Data Type (CVTMQMDTA) command produces a fragment of code to perform data conversion on data type structures, for use by the data-conversion exit program.

For information on how to use the data-conversion exit, see the IBM MQ Application Programming Guide.

Support is provided for the C programming language only.

## Parameters

Table 252. Command parameters			
Keyword	Description	Choices	Notes
<u>FROMFILE</u>	Input file	Qualified object name	Required, Positional 1
	Qualifier 1: Input file	Name	
	Qualifier 2: Library	<i>Name</i> , *LIBL, *CURLIB	
<u>FROMMBR</u>	Member containing input	Name	Required, Positional 2
<u>TOFILE</u>	File to receive output	Qualified object name	Required, Positional 3
	Qualifier 1: File to receive output	Name	
	Qualifier 2: Library	<i>Name</i> , *LIBL, *CURLIB	
<u>TOMBR</u>	Member to receive output	<i>Name</i> , *FROMMBR	Optional, Positional 4
<u>RPLTOMBR</u>	Replace to member	*YES, *NO	Optional, Positional 5

### Input file (FROMFILE)

Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the data to convert.

The possible values are:

#### \*LIBL

The library list is searched for the file name.

#### \*CURLIB

The current library is used.

#### from-library-name

Specify the name of the library to be used.

#### from-file-name

Specify the name of the file containing the data to convert.

### Member containing input (FROMMBR)

Specifies the name of the member containing the data to be converted.

The possible values are:

#### from-member-name

Specifies the name of the member containing the data to convert.

### File to receive output (TOFILE)

Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the converted data.

The possible values are:

#### \*LIBL

The library list is searched for the file name.

#### \*CURLIB

The current library is used.

#### to-library-name

Specify the name of the library to be used.

#### to-file-name

Specify the name of the file to contain the converted data.

## Member to receive output (TOMBR)

Specifies the name of the member containing the converted data.

The possible values are:

### \*FROMMBR

The from-member name is used.

### to-member-name

Specify the name of the member containing the converted data.

## Replace to member (RPLTOMBR)

Specifies whether the converted data replaces the existing member.

The possible values are:

### \*YES

The converted data replaces the existing member.

### \*NO

The converted data does not replace the existing member.

## DLTMQM (Delete Message Queue Manager)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Delete Message Queue Manager (DLTMQM) command deletes the specified local queue manager.

## Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

## DLTMQMAUTI (Delete MQ AuthInfo object)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Delete MQ AuthInfo object (DLTMQMAUTI) command deletes an existing MQ authentication information object.

## Parameters

Keyword	Description	Choices	Notes
<u>AINAME</u>	AuthInfo name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2

### AuthInfo name (AINAME)

The name of the authentication information object to delete.

If an application has this open, the command fails.

The possible values are:

#### authentication-information-name

Specify the name of the authentication information object. The maximum string length is 48 characters.

### Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of an existing message queue manager. The maximum string length is 48 characters.

## DLTMQMBRK (Delete MQ Pub/Sub Broker)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The delete IBM MQ broker command (DLTMQMBRK) is used to delete the broker. The broker must be stopped when this command is issued, and the queue manager must be running. If the broker is already started, you must issue ENDMQMBRK before issuing this command. To delete more than one broker in the hierarchy, it is essential that you stop (using the ENDMQMBRK command) and delete each broker one at a time. You should not attempt to stop all the brokers in the hierarchy that you want to delete first and then try to delete them.

## Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

**queue-manager-name**

Specify the name of the queue manager.


**DLTMQMCHL (Delete MQ Channel)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Delete MQ Channel (DLTMQMCHL) command deletes the specified channel definition.

**Parameters**

<i>Table 256. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>CHLNAME</u>	Channel name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 2
<u>CHLTYPE</u>	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, <b>*NONCLT</b> , *CLTCN	Optional, Positional 3

**Channel name (CHLNAME)**

Specifies the name of the channel definition.

The possible values are:

**channel-name**

Specify the channel name.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

**message-queue-manager-name**

The name of a message queue manager.

**Channel type**

Specifies the type of the channel to delete.

The possible values are:

**\*NONCLT**

Any channel type, that is not a client-connection channel, that matches the channel name.

**\*SDR**

Sender channel

**\*SVR**

Server channel

**\*RCVR**

Receiver channel

**\*RQSTR**

Requester channel

**\*SVRCN**

Server-connection channel

**\*CLUSSDR**

Cluster-sender channel

**\*CLUSRCVR**

Cluster-receiver channel

**\*CLTCN**

Client-connection channel

**IBM i DLTMQMLSR (Delete MQ Listener)****Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Delete MQ Listener object (DSPMQMLSR) command deletes an existing MQ listener object.

**Parameters**

<i>Table 257. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>LSRNAME</u>	Listener name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 2

**Listener name (LSRNAME)**

The name of the listener object to delete.

The possible values are:

**listener-name**

Specify the name of the listener definition. The maximum length of the string is 48 bytes.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of a message queue manager.

**IBM i DLTMQMNL (Delete MQ Namelist)****Where allowed to run**

All environments (\*ALL)



## Threadsafe

Yes

The Delete MQ Namelist (DLTMQMNL) command deletes the specified namelist on the selected local queue manager.

## Parameters

Table 258. Command parameters

Keyword	Description	Choices	Notes
<u>NAMELIST</u>	Namelist	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2

### Namelist (NAMELIST)

The name of the namelist to delete.

#### namelist

Specify the name of the namelist. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### **\*DFT**

The default queue manager is used.

#### message-queue-manager-name

Specify the name of the queue manager.

## DLTMQMPRC (Delete MQ Process)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Delete MQ Process (DLTMQMPRC) command deletes an existing MQ process definition.

## Parameters

Table 259. Command parameters

Keyword	Description	Choices	Notes
<u>PRCNAME</u>	Process name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2

### Process name (PRCNAME)

The name of the process definition to delete. If an application has this process open, the command fails.

The possible values are:

**process-name**

Specify the name of the process definition. The maximum length of the string is 48 bytes.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of a message queue manager.


**DLTMQMQ (Delete MQ Queue)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Delete MQ Queue (DLTMQMQ) command deletes an MQ queue.

If the queue is a local queue, it must be empty for the command to succeed. CLRMQMQ can be used to clear all of the messages from a local queue.

The command fails if an application has:

- This queue open
- A queue that resolves to this queue open
- A queue open that resolves through this definition as a queue manager alias.

An application using the definition as a reply-to queue alias, however, does not cause this command to fail.

**Parameters**

<i>Table 260. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>QNAME</u>	Queue name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 2

**Queue name (QNAME)**

The name of the queue.

The possible values are:

**queue-name**

Specify the name of the queue.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.


**DLTMQMSUB (Delete MQ Subscription)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Delete MQ Subscription (DLTMQMSUB) command deletes an existing MQ subscription.

**Parameters**

<i>Table 261. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>SUBID</u>	Subscription identifier	<i>Character value, *NONE</i>	Optional, Positional 1
<u>SUBNAME</u>	Subscription name	<i>Character value, *NONE</i>	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 3

**Subscription identifier (SUBID)**

The subscription identifier of the subscription to delete.

The possible values are:

**subscription-name**

Specify a maximum of 256 bytes for the subscription name.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

**Subscription name (SUBNAME)**

The name of the subscription to delete.

The possible values are:

**subscription-name**

Specify a maximum of 256 bytes for the subscription name.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the Queue Manager.

The possible values are:

**\*DFT**

Use the default Queue Manager.

**queue-manager-name**

The name of a Queue Manager.


**DLTMQMSVC (Delete MQ Service)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Delete MQ Service object (DLTMQMSVC) command deletes an existing MQ service object.

**Parameters**

<i>Table 262. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>SVCNAME</u>	Service name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 2

**Service name (SVCNAME)**

The name of the service object to delete.

The possible values are:

**service-name**

Specify the name of the service definition. The maximum length of the string is 48 bytes.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of a message queue manager.

 **DLTMQMTOP (Delete MQ Topic)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Delete MQ Topic (DLTMQMTOP) command deletes an existing MQ topic object.

**Parameters**

<i>Table 263. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>TOPNAME</u>	Topic name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 2

**Topic name (TOPNAME)**

The name of the topic object to delete. If an application has this topic open, the command fails.

The possible values are:

**topic-name**

Specify the name of the topic object. The maximum length of the string is 48 bytes.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the Queue Manager.

The possible values are:

**\*DFT**

Use the default Queue Manager.

**queue-manager-name**

The name of a Queue Manager.

 **DMPMQMCFG (Dump MQ Configuration)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Dump MQ Configuration (DMPMQMCFG) command is used to dump the configuration objects and authorities for a queue manager.

**Parameters**

<i>Table 264. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *ALL</i>	Optional, Positional 1
<u>OBJ</u>	Object name	<i>Character value, *ALL</i>	Optional, Positional 2
<u>OBJTYPE</u>	Object type	<b>*ALL, *AUTHINFO, *CHL, *CLTCN, *COMMINFO, *LSR, *NMLIST, *PRC, *Q, *MQM, *SVC, *SUB, *TOPIC</b>	Optional, Positional 3
<u>EXPTYPE</u>	Export type	<b>*ALL, *OBJECT, *AUTHREC, *CHLAUTH</b>	Optional, Positional 4
<u>EXPATTR</u>	Export attributes	<b>*NONDEF, *ALL</b>	Optional, Positional 5
<u>WARN</u>	Warnings	<b>*NO, *YES</b>	Optional, Positional 6
<u>OUTPUT</u>	Output	<b>*MQSC, *ONELINE, *SETMQAUT, *GRTMQMAUT</b>	Optional, Positional 7
<u>CLIENT</u>	Client connection	<b>*NO, *YES, *CHL</b>	Optional, Positional 8
<u>CLIENTCHL</u>	MQSC Channel Definition	<i>Character value, *NONE</i>	Optional, Positional 9
<u>MSGSEQNUM</u>	Message sequence number	1-999999999, <b>*NORESET</b>	Optional, Positional 10
<u>RPLYQ</u>	Reply Queue	<i>Character value, 'SYSTEM.DEFAULT.MODEL.QUEUE'</i>	Optional, Positional 11
<u>RMTMQMNAM</u> <u>E</u>	Remote Message Queue Manager	<i>Character value, *NONE</i>	Optional, Positional 12

Table 264. Command parameters (continued)

Keyword	Description	Choices	Notes
TOFILE	File to receive output	<i>Qualified object name</i>	Optional, Positional 13
	Qualifier 1: File to receive output	<i>Name</i>	
	Qualifier 2: Library	<i>Name, *LIBL</i>	
TOMBR	Member to receive output	<i>Name</i>	Optional, Positional 14

### Message Queue Manager name (MQMNAME)

Specifies the name of the IBM MQ queue manager for which object information is to displayed.

The possible values are:

**\*DFT**

**queue-manager-name**

The name of an existing message queue manager. The maximum string length is 48 characters.

### Object name (OBJ)

Specifies the name of the objects to dump. It is a 48-character MQ object or generic object name.

The possible values are:

**\*ALL**

All objects of the specified type (OBJTYPE) are dumped.

**generic-object-name**

Specify the generic name of the objects. A generic name is a character string followed by an asterisk (\*). For example, ABC\*. It selects all objects having names that start with the selected character string.

Specifying the required name within quotation marks ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**object-name**

The name of an object for which the corresponding name and type is to be displayed.

### Object type (OBJTYPE)

Specifies the type of the objects to be dumped.

The possible values are:

**\*ALL**

All MQ Objects with names specified by OBJ.

**\*AUTHINFO**

All MQ authentication information objects with names specified by OBJ.

**\*CHL**

All MQ channel objects with names specified by OBJ.

**\*CLTCN**

All MQ client connection objects with names specified by OBJ.

**\*COMMINFO**

All MQ communication information objects with names specified by OBJ.

**\*LSR**

All MQ listener objects with names specified by OBJ.

**\*NMLIST**

All MQ namelist objects with names specified by OBJ.

**\*PRC**

All MQ process objects with names specified by OBJ.

**\*Q**

All MQ queue objects with names specified by OBJ.

**\*MQM**

The queue manager object.

**\*SVC**

All MQ service objects with names specified by OBJ.

**\*TOPIC**

All MQ topic objects with names specified by OBJ.

**Export type (EXPTYPE)**

Specifies the type of the export.

The possible values are:

**\*ALL**

All MQ object, authority and subscription configuration information is dumped.

**\*OBJECT**

Only MQ object information is dumped.

**\*AUTHREC**

Only MQ authority information is dumped.

**\*CHLAUTH**

Only MQ channel authority records are dumped.

**\*SUB**

Only MQ durable subscription information is dumped.

**Export attributes (EXPATTR)**

Specifies the attributes to export.

The possible values are:

**\*NONDEF**

Only non-default attribute values are dumped.

**\*ALL**

All attribute values are dumped.

**Warnings (WARN)**

Specifies whether warnings should be generated during the dump, for example if the command is issued against a newer queue manager or encounters a damaged object.

The possible values are:

**\*NO**

No warnings messages will be issued during the dump.

**\*YES**

Warning messages may be issued during the dump.

## Output (OUTPUT)

Specifies the output format from the dump.

The possible values are:

### **\*MQSC**

The output format is in the form of MQSC commands that could be used as input to the RUNMQSC or STRMQMMQSC commands.

### **\*ONELINE**

The output format is in the form of MQSC commands formatted into single line records, suitable for use with line comparison tools.

### **\*SETMQAUT**

The output format is in the form of setmqaut commands, suitable for use with AIX, Linux, and Windows.

### **\*GRTMQMAUT**

The output format is in the form of GRTMQMAUT commands, suitable for use generating a CL program on the IBM i platform.

## Client connection (CLIENT)

Specifies whether to use a client connection to the queue manager.

The possible values are:

### **\*NO**

The command will first attempt a server bindings connection, if this connection fails a client connection will be attempted.

### **\*YES**

The command will attempt to connect via a client connection using the default client connection process. If the MQSERVER environment variable is set it will override use of a client connection channel table.

### **\*CHL**

The command will attempt to connect to the queue manager using a temporary channel definition defined by the MQSC string specified in the CLIENTCHL parameter.

## MQSC Channel Definition (CLIENTCHL)

Specifies, via MQSC syntax, a temporary client channel definition to use in connecting to the queue manager.

The possible values are:

### **\*NONE**

Do not use a temporary client channel definition when connecting to the queue manager.

### **mqsc-define-channel-string**

The command will attempt to construct a temporary client channel definition from the using the MQSC command supplied on this parameter. The MQSC command must define all required attributes for a client connection channel, for example:

```
"DEFINE CHANNEL(MY.CHL) CHLTYPE(CLNTCONN) CONNAME(MYHOST.MYCORP.COM(1414))"
```

## Message sequence number (MSGSEQNUM)

Specifies whether to generate reset channel commands for sender, server and cluster sender channel types when dumping channel objects.

The possible values are:

### **\*NORESET**

Do not include any reset channel commands in the dumped output.



## 1 - 999999999

Specify a message sequence number for the reset channel commands included in the dump.

### Reply Queue (RPLYQ)

Specifies the name of the queue to use for receiving PCF replies when inquiring configuration information.

The possible values are:

#### **SYSTEM.DEFAULT.MODEL.QUEUE**

The default model queue, a dynamic queue will be generated to receive replies.

#### **reply-to-queue-name**

Specify the name of the reply to queue.

### Remote Message Queue Manager (RMTMQMNAME)

Specifies the name of a remote MQ queue manager for which object information is to be displayed.

The possible values are:

#### **\*NONE**

The configuration information is collected from the queue manager specified in the MQMNAME parameter.

#### **remote-queue-manager-name**

Specify the name of the remote queue manager. PCF inquiry commands are issued to the queue manager specified in RMTMQMNAME via the queue manager specified in MQMNAME, this is known as queued mode. \

### File to receive output (TOFILE)

Specifies the qualified name of the file, in the form LIBRARY/FILE, that will be used to store the dumped configuration data. The FILE should have been created with a record length of 240, otherwise the configuration information might be truncated.

The possible values are:

#### **\*LIBL**

The library list is searched for the file name.

#### **\*CURLIB**

The current library is used.

#### **to-library-name**

Specify the name of the library to be used.

#### **to-file-name**

Specify the name of the file to contain the configuration data.

### Member to receive output (TOMBR)

Specifies the name of the member to store the dumped configuration data.

The possible values are:

#### **to-member-name**

Specify the name of the member to contain the configuration data.

### Examples

To make these examples work you need to ensure that your system is set up for remote MQSC operation. See [Configuring queue managers for remote administration](#).

```
DMPMQMCFG MQMNAME('MYQMGR') CLIENT(*YES) CLIENTCHL(''DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN)
CHLTYPE(CLNTCONN) CONNAME('myhost.mycorp.com(1414)')')
```

dumps all the configuration information from remote queue manager *MYQMGR* in MQSC format and creates an ad-hoc client connection to the queue manager using a client channel called *SYSTEM.ADMIN.SVRCONN*.


**Note:** You need to ensure that a server-connection channel with the same name exists.

```
DMPMQMCFG MQMNAME('LOCALQM') RMTMQMNAME('MYQMGR')
```

dumps all configuration information from remote queue manager *MYQMGR*, in MQSC format, connects initially to local queue manager *LOCALQM*, and sends inquiry messages through this local queue manager.

**Note:** You need to ensure that the local queue manager has a transmission queue named *MYQMGR*, with channel pairings defined in both directions, to send and receive replies between queue managers.

### Related tasks

 [Backing up queue manager configuration](#)

 [Restoring queue manager configuration](#)

## DSCMQM (Disconnect MQ)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Disconnect Message Queue Manager (DSCMQM) command does not perform any function and is provided only for compatibility with previous releases of IBM MQ and MQSeries.

### Parameters

None

## DSPMQM (Display Message Queue Manager)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Display Message Queue Manager (DSPMQM) command displays the attributes of the specified local queue manager.

### Parameters

<i>Table 265. Command parameters</i>			
Keyword	Description	Choices	Notes
<a href="#">OUTPUT</a>	Output	*, *PRINT	Optional, Positional 1
<a href="#">MQMNAME</a>	Message Queue Manager name	Character value, *DFT	Optional, Positional 2

### Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

\*

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**IBM i DSPMQMAUT (Display MQ Object Authority)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display MQ Authority (DSPMQMAUT) command shows, for the specified object, the current authorizations to the object. If a user ID is a member of more than one group, this command displays the combined authorizations of all of the groups.

- The 48-character MQ object name
- The MQ object type
- Authorizations for object, context and MQI calls

**Parameters**

<i>Table 266. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>OBJ</u>	Object name	Character value	Required, Positional 1
<u>OBJTYPE</u>	Object type	*Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC, *RMTMQMNAME	Required, Positional 2
<u>USER</u>	User name	Name, *PUBLIC	Optional, Positional 3
<u>OUTPUT</u>	Output	*, *PRINT	Optional, Positional 4
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 5
<u>SRVCOMP</u>	Service Component name	Character value, *DFT	Optional, Positional 6

**Object name (OBJ)**

Specifies the name of the MQ object for which the authorizations are displayed.

## Object type (OBJTYPE)

Specifies the type of the object for which the authorizations are displayed.

### **\*Q**

All queue object types.

### **\*ALSQ**

Alias queue.

### **\*LCLQ**

Local queue.

### **\*MDLQ**

Model queue.

### **\*RMTQ**

Remote queue.

### **\*AUTHINFO**

Authentication Information object.

### **\*MQM**

Message Queue Manager.

### **\*NMLIST**

Namelist object.

### **\*PRC**

Process definition.

### **\*CHL**

Channel object.

### **\*CLTCN**

Client Connection Channel object.

### **\*LSR**

Listener object.

### **\*SVC**

Service object.

### **\*TOPIC**

Topic object.

### **\*RMTMQMNAME**

Remote queue manager name.

## User name (USER)

Specifies the name of the user for whom authorities for the named object are displayed.

The possible values are:

### **\*PUBLIC**

All users of the system.

### **user-profile-name**

Specify the name of the user.

## Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

\*

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Service Component name (SRVCOMP)**

Specifies the name of the installed authorization service in which to search for the authority to display.

The possible values are:

**\*DFT**

All installed authorization components are searched for the specified object name, object type and user.

**Authorization-service-component-name**

The component name of the required authorization service as specified in the Queue manager's qm.ini file.

 **DSPMQMAUTI (Display MQ AuthInfo object)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display MQ AuthInfo object (DSPMQMAUTI) command displays the attributes of an existing MQ authentication information object.

**Parameters**

Table 267. Command parameters

Keyword	Description	Choices	Notes
<u>AINAME</u>	AuthInfo name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 2
<u>OUTPUT</u>	Output	Character value, *, *PRINT	Optional, Positional 3

**AuthInfo name (AINAME)**

The name of the authentication information object to display.

The possible values are:

**authentication-information-name**

Specify the name of the authentication information object. The maximum string length is 48 characters.

**Message Queue Manager name (MQMNAME)**

The name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of an existing message queue manager. The maximum string length is 48 characters.

**Output (OUTPUT)**

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

 **DSPMQMBRK (Display MQ Pub/Sub Broker)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display IBM MQ broker (DSPMQMBRK) command does not perform any function and is only provided for compatibility with previous releases of IBM MQ.

**Parameters**

<i>Table 268. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1

**Message Queue Manager name (MQMNAME)**

The name of the queue manager.

The value is:

**queue-manager-name**

The name of an existing message queue manager. The maximum string length is 48 characters.

 **DSPMQMCHL (Display MQ Channel)**
**Where allowed to run**

All environments (\*ALL)

## Threadsafe

Yes

The Display MQ Channel (DSPMQMCHL) command displays the attributes of an existing MQ channel definition.

## Parameters

Table 269. Command parameters

Keyword	Description	Choices	Notes
<u>CHLNAME</u>	Channel name	Character value	Required, Positional 1
<u>OUTPUT</u>	Output	<b>*</b> , *PRINT	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 3
<u>CHLTYPE</u>	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, <b>*NONCLT</b> , *CLTCN	Optional, Positional 4

### Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

#### channel-name

Specify the channel name.

### Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

#### \*PRINT

The output is printed with the job's spooled output.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

#### message-queue-manager-name

The name of a message queue manager.

### Channel type (CHLTYPE)

Specifies the type of the channel to be displayed.

The possible values are:

**\*NONCLT**

Any channel type, that is not a client-connection channel, that matches the channel name.

**\*SDR**

Sender channel

**\*SVR**

Server channel

**\*RCVR**

Receiver channel

**\*RQSTR**

Requester channel

**\*SVRCN**

Server-connection channel

**\*CLUSSDR**

Cluster-sender channel

**\*CLUSRCVR**

Cluster-receiver channel

**\*CLTCN**

Client-connection channel

## **DSPMQMCSVR (Display MQ Command Server)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display MQ Command Server (DSPMQMCSVR) command displays the status of the MQ command server.

The status of the command server can be one of the following:

**Enabled**

Available to process messages

**Disabled**

Not available to process messages

**Starting**

STRMQMCSVR command in progress

**Stopping**

ENDMQMCSVR command in progress

**Stopped**

ENDMQMCSVR command completed

**Running**

Processing a message

**Waiting**

Waiting for a message

**Parameters**

<i>Table 270. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 1



## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.

### **queue-manager-name**

Specify the name of the queue manager.

## DSPMQMLSR (Display MQ Listener)

### **Where allowed to run**

All environments (\*ALL)

### **Threadsafe**

Yes

The Display MQ Listener object (DSPMQMLSR) command displays the attributes of an existing MQ listener object.

## Parameters

Keyword	Description	Choices	Notes
<u>LSRNAME</u>	Listener name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Key, Positional 2
<u>OUTPUT</u>	Output	<b>*</b> , *PRINT	Optional, Positional 3

## Listener name (LSRNAME)

The name of the listener object to display.

The possible values are:

### **listener-name**

Specify the name of the listener definition. The maximum length of the string is 48 bytes.

## Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.

### **queue-manager-name**

The name of a message queue manager.

## Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

\*

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

 **DSPMQMNL (Display MQ Namelist)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display MQ Namelist (DSPMQMNL) command displays an MQ namelist.

**Parameters**

Keyword	Description	Choices	Notes
<u>NAMELIST</u>	Namelist	Character value	Required, Positional 1
<u>OUTPUT</u>	Output	<b>*</b> , *PRINT	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 3

**Namelist (NAMELIST)**

The name of the namelist to be displayed.

**namelist**

Specify the name of the namelist. The maximum length of the string is 48 bytes.

**Output (OUTPUT)**

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

\*

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used.

**message-queue-manager-name**

Specify the name of the queue manager.

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display MQ Object Names (DSPMQMOBJN) command is used to provide the name, type, and fully-qualified file name for a specified MQ object.

**Parameters**

Keyword	Description	Choices	Notes
<u>OBJ</u>	Object name	<i>Character value</i> , *ALL	Required, Positional 1
<u>OBJTYPE</u>	Object type	<b>*ALLMQM</b> , *Q, *ALSO, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *CHL, *CLTCN, *SVC, *MQM, *NMLIST, *PRC, *LSR, *TOPIC	Optional, Positional 2
<u>OUTPUT</u>	Output	<b>*</b> , *PRINT	Optional, Positional 3
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 4

**Object name (OBJ)**

Specifies the name of the objects for which the corresponding name, type and file name to display. It is a 48-character MQ object or generic object name.

The possible values are:

**\*ALL**

All objects of the specified type (OBJTYPE) are displayed.

**generic-object-name**

Specify the generic name of the objects. A generic name is a character string followed by an asterisk (\*). For example, ABC\*. It selects all objects having names that start with the selected character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**object-name**

The name of an object for which the corresponding name and type is to be displayed.

**Object type (OBJTYPE)**

Specifies the type of the objects to be displayed.

The possible values are:

**\*ALLMQM**

All MQ Objects with names specified by OBJ.

**\*Q**

All MQ queues with names specified by OBJ.

**\*ALSQ**

All MQ alias queues with names specified by OBJ.

**\*LCLQ**

All MQ local queues with names specified by OBJ.

**\*MDLQ**

All MQ model queues with names specified by OBJ.

**\*RMTQ**

All MQ remote queues with names specified by OBJ.

**\*AUTHINFO**

All MQ authentication information objects with names specified by OBJ.

**\*CHL**

All MQ channel objects with names specified by OBJ.

**\*CLTCN**

All MQ MQI client connection channel objects with names specified by OBJ.

**\*SVC**

All MQ service objects with names specified by OBJ.

**\*LSR**

All MQ listener objects with names specified by OBJ.

**\*CTLG**

The MQ queue manager catalog object with name specified by OBJ. This has the same name as the queue manager object.

**\*MQM**

The Message Queue Manager object with name specified by OBJ.

**\*NMLIST**

All MQ namelists with names specified by OBJ.

**\*PRC**

All MQ process definitions with names specified by OBJ.

**\*LOBJ**

All MQ listener objects with names specified by OBJ.

**\*TOPIC**

All MQ topic objects with names specified by OBJ.

**Output (OUTPUT)**

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the MQ queue manager for which object information is to be displayed.

The possible values are:

**\*DFT**

The default queue manager.

**queue-manager-name**

Specify the name of the queue manager.


**DSPMQMPCRC (Display MQ Process)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display MQ Process (DSPMQMPCRC) command displays the attributes of an existing MQ process definition.

**Parameters**

<i>Table 274. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>PRCNAME</u>	Process name	Character value	Required, Positional 1
<u>OUTPUT</u>	Output	<b>*</b> , *PRINT	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 3

**Process name (PRCNAME)**

The name of the process definition to be displayed.

The possible values are:

**process-name**

Specify the name of the process definition. The maximum length of the string is 48 bytes.

**Output (OUTPUT)**

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of a message queue manager.

## IBM i DSPMQMQ (Display MQ Queue)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Display MQ Queue (DSPMQMQ) command displays the attributes of an existing MQ queue definition.

### Parameters

Keyword	Description	Choices	Notes
<u>QNAME</u>	Queue name	Character value	Required, Positional 1
<u>OUTPUT</u>	Output	<b>*</b> , *PRINT	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 3

### Queue name (QNAME)

The name of the queue.

The possible values are:

#### queue-name

Specify the name of the queue.

### Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

#### \*PRINT

The output is printed with the job's spooled output.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

Specify the name of the queue manager.

## IBM i DSPMQMRTE (Display MQ Route Information)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The DSPMQMRTE command generates a trace route message based on user specified parameters and puts it to a specified queue. One or more reports about the route the message takes to its final destination might be generated, as well as a reply. These will be got from a specified reply queue and the information contained within them will be written to the job's spooled output when it is received.

## Parameters

<i>Table 276. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>QNAME</u>	Target object	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 2
<u>CRRLID</u>	Correlation Identifier	<i>Character value</i> , <b>*NONE</b>	Optional, Positional 3
<u>MSGPST</u>	Message Persistence	<b>*YES</b> , <b>*NO</b> , <b>*QUEUE</b>	Optional, Positional 4
<u>MSGPRTY</u>	Message Priority	0-9, <b>*QUEUE</b>	Optional, Positional 5
<u>OPTION</u>	Report Option	Single values: <b>*DFT</b> , <b>*NONE</b> Other values (up to 6 repetitions): <b>*ACTIVITY</b> , <b>*COA</b> , <b>*COD</b> , <b>*DISCARD</b> , <b>*EXCEPTION</b> , <b>*EXPIRATION</b>	Optional, Positional 6
<u>RPLYQ</u>	Reply Queue	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 7
<u>RPLYMQM</u>	Reply Queue Manager	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 8
<u>EXPIRY</u>	Message Expiry	0-999999999, <b>*DFT</b>	Optional, Positional 9
<u>EXPRPT</u>	Pass Expiry	<b>*YES</b> , <b>*NO</b>	Optional, Positional 10
<u>RTEINF</u>	Route Accumulation	<b>*YES</b> , <b>*NO</b>	Optional, Positional 11
<u>RPLYMSG</u>	Reply Message	<b>*YES</b> , <b>*NO</b>	Optional, Positional 12
<u>DLVRMSG</u>	Deliver Message	<b>*YES</b> , <b>*NO</b>	Optional, Positional 13
<u>FWDMSG</u>	Forward Message	<b>*SUPPORT</b> , <b>*ALL</b>	Optional, Positional 14
<u>MAXACTS</u>	Maximum Activities	1-999999999, <b>*NOMAX</b>	Optional, Positional 15
<u>DETAIL</u>	Route Detail	<b>*LOW</b> , <b>*MEDIUM</b> , <b>*HIGH</b>	Optional, Positional 16
<u>BROWSE</u>	Browse Only	<b>*YES</b> , <b>*NO</b>	Optional, Positional 17
<u>DSPMSG</u>	Display Message	<b>*YES</b> , <b>*NO</b>	Optional, Positional 18
<u>TGTMQM</u>	Target Queue Manager	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 19
<u>DSPINF</u>	Display Information	Single values: <b>*ALL</b> , <b>*SUMMARY</b> , <b>*NONE</b> Other values (up to 6 repetitions): <b>*ACTGRP</b> , <b>*ID</b> , <b>*MSGGRP</b> , <b>*MSGDELTA</b> , <b>*OPGRP</b> , <b>*TRGRP</b>	Optional, Positional 20
<u>WAIT</u>	Wait Time	0-999999999, <b>*DFT</b>	Optional, Positional 21
<u>BIND</u>	Bind Option	<b>*OPEN</b> , <b>*NOTFIXED</b>	Optional, Positional 22

## Target object (QNAME)

Specifies the name of the target queue of the trace route message or, if displaying previously gathered information, the name of the queue storing the information.

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.

### **message-queue-manager-name**

Specify the name of the queue manager.

## Correlation Identifier (CRRLID)

Specifies the CorrelId to use when retrieving previously gathered information. The format of the 24 byte CorrelId is a 48 character hexadecimal string. You must supply a CorrelId if you are retrieving previously gathered information, rather than generating a trace route message.

The possible values are:

### **\*NONE**

No CorrelId is supplied.

### **correlation-identifier**

The 48 character hexadecimal string representing the 24 byte CorrelId.

## Message Persistence (MSGPST)

Specifies the persistence of the trace route message.

The possible values are:

### **\*NO**

The message will be put with MQPER\_NOT\_PERSISTENT.

### **\*YES**

The message will be put with MQPER\_PERSISTENT.

### **\*QUEUE**

The message will be put with MQPER\_PERSISTENCE\_AS\_Q\_DEF.

## Message Priority (MSGPRTY)

Specifies the priority of the trace route message.

The possible values are:

### **\*QUEUE**

The message will be put with MQPRI\_PRIORITY\_AS\_Q\_DEF.

### **message-priority**

The priority of the message ranging 0 through 9.

## Report Option (OPTION)

Specifies the report options of the trace route message. Reports generated on a non trace route enabled queue manager can potentially remain in the network undelivered, which is why most report options are disabled by default. By requesting full data to be returned, it allows the trace route information contained in the message to be returned in the result of a problem.

The possible values are:



**\*DFT**

Turns on MQRO\_ACTIVITY and MQRO\_DISCARD\_MSG.

**\*NONE**

No report options are set.

**\*ACTIVITY**

Turns on MQRO\_ACTIVITY.

**\*COA**

Turns on MQRO\_COA\_WITH\_FULL\_DATA.

**\*COD**

Turns on MQRO\_COD\_WITH\_FULL\_DATA.

**\*DISCARD**

Turns on MQRO\_DISCARD\_MSG.

**\*EXCEPTION**

Turns on MQRO\_EXCEPTION\_WITH\_FULL\_DATA.

**\*EXPIRATION**

Turns on MQRO\_EXPIRATION\_WITH\_FULL\_DATA.

**Reply Queue (RPLYQ)**

Specifies the name of the reply queue to which the reply and all report messages should be sent. This must exist on the local queue manager unless the RPLYMQM parameter is also specified. The reply queue should not be a temporary queue if the trace route message is to be persistent.

The possible values are:

**\*DFT**

The SYSTEM.DEFAULT.MODEL.QUEUE is used and the reply queue is by default a temporary dynamic queue.

**reply-queue**

The name of the reply queue to use.

**Reply Queue Manager (RPLYMQM)**

Specifies the queue manager to which replies are sent.

The possible values are:

**\*DFT**

Replies are sent to the local queue manager.

**reply-queue-manager**

The name of the reply to queue manager.

**Message Expiry (EXPIRY)**

Specifies the Expiry time, in seconds, of the trace route message.

The possible values are:

**\*DFT**

The default expiry time of 60 seconds is used.

**expiry-time**

The expiry time of the message ranging from 0 through 999999999.

## Pass Expiry (EXPRPT)

Specifies whether the expiry of the trace route message is passed to reports or the reply message. This effectively turns MQRO\_PASS\_DISCARD\_AND\_EXPIRY on and off. This allows users to keep the reports indefinitely if required.

The possible values are:

**\*YES**

Expiry is passed to reports or the reply message.

**\*NO**

Expiry is not passed to reports or the reply message.

## Route Accumulation (RTEINF)

Specifies that the route information is accumulated within the trace route message as it flows through the queue manager network.

The possible values are:

**\*NO**

No information is accumulated within the trace route message.

**\*YES**

Information is accumulated within the trace route message.

## Reply Message (RPLYMSG)

Requests that a reply message containing all accumulated information is returned to the reply to queue when the trace route message reaches its final destination (if this is permitted by the queue manager hosting the final destination queue).

The possible values are:

**\*NO**

No reply message is returned.

**\*YES**

A reply message is returned to the the reply to queue.

## Deliver Message (DLVRMSG)

Specifies whether the trace route message is delivered to getting applications if the message successfully arrives at the destination queue.

The possible values are:

**\*NO**

If the trace route message successfully arrives at the target queue it is not delivered to getting applications.

**\*YES**

The trace route message is delivered to a getting application if the message successfully arrives at the target queue. Specifying this option effectively gives permission for the message to arrive on a queue manager, whether it supports trace route or not.

## Forward Message (FWDMSG)

Specifies whether the trace route message is forwarded to the next queue manager in the route.

The possible values are:

**\*SUPPORT**

The trace route message is forwarded only to queue managers that can ensure that the delivery option is honoured.

**\*ALL**

The trace route message is forwarded on without any regard given to the next queue manager in the route. This option can be used to force a non-trace route enabled queue manager to accept trace route messages, even when they cannot process them in line with the delivery option.

**Maximum Activities (MAXACTS)**

Specifies the maximum number of activities that can take place on the trace route message before it is discarded.

The possible values are:

**\*NOMAX**

No maximum number of activities are specified.

**maximum-activities**

The maximum number of activities ranging from 1 through 999999999.

**Route Detail (DETAIL)**

Specifies how much detail about the route is requested.

The possible values are:

**\*LOW**

At this level of detail no information about queue manager activities is requested. This gives a very high level view of what user activity has taken place on the message.

**\*MEDIUM**

Low detail information, as well as information on the movements of the message within the queue manager is requested. This includes the work of the MCA.

**\*HIGH**

Low and medium detail, as well as more detailed information about the route the message took is requested. For example, in clustering this might include detail about why the route was chosen.

**Browse Only (BROWSE)**

Specifies whether messages returned are browsed only. This means that the information remains on the queue for future display operations.

The possible values are:

**\*NO**

Messages returned are not browse only.

**\*YES**

Messages returned are browse only.

**Display Message (DSPMSG)**

Specifies whether when a trace route message is generated the information returned is displayed.

The possible values are:

**\*YES**

The returned information is displayed.

**\*NO**

The returned information is not displayed. This allows DSPMQMRTE to exit as soon as the trace route message has been put to the target queue. On exit, a 48 character hexadecimal string is output, which is the MsgId on the trace route message that was generated and can be used as the CRRLID supplied to a subsequent DSPMQMRTE call.

## Target Queue Manager (TGTMQM)

Specifies the target queue manager for the trace route message.

The possible values are:

### **\*DFT**

No target queue manager is specified. Either the destination queue is a local queue, or there is a local definition of the queue.

### **target-queue-manager**

The target queue manager for the trace route message.

## Display Information (DSPINF)

Specifies how much of the information gathered should be displayed.

The possible values are:

### **\*ALL**

All available information is displayed.

### **\*SUMMARY**

Displays only the queues which the message was routed through.

### **\*NONE**

None of the available information will be displayed.

### **\*ACTGRP**

All non-group parameters in the Activity groups will be displayed.

### **\*ID**

Values with parameters identifiers MQBACF\_MSG\_ID or MQBACF\_CORREL\_ID are always displayed. This overrides \*MSGDELTA which normally prevents certain values in the Message groups from being displayed.

### **\*MSGGRP**

All non-group parameters in the Message groups are displayed.

### **\*MSGDELTA**

Like \*MSGGRP, except that information in the Message groups is only displayed where it has changed since the last operation took place.

### **\*OPGRP**

All non-group parameters in the Operation groups are displayed.

### **\*TRGRP**

All parameters in the TraceRoute groups are displayed.

## Wait Time (WAIT)

Specifies how long, in seconds, that DSPMQMRTE should wait before assuming that all a reply message or all the reports (depending on the options specified) that were generated en route that can be delivered to the reply queue have now done so.

The possible values are:

### **\*DFT**

DSPMQMRTE waits for 60 seconds longer than the Expiry time of the trace route message.

### **wait-time**

The time that DSPMQMRTE should wait.

## Bind Option (BIND)

Specifies whether the target queue is bound to a specific destination.

The possible values are:

**\*OPEN**

The target queue is bound to a specific destination. The queue is opened with option MQOO\_BIND\_ON\_OPEN.

**\*NOTFIXED**

The target queue is not bound to a specific destination. Typically this parameter is used when the trace route message is to be put across a cluster. The queue is opened with option MQOO\_BIND\_NOT\_FIXED.


**DSPMQMSPL (Display MQM Security Policies)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display MQM Security Policies (DSPMQMSPL) command displays security policies, that are used by Advanced Message Security to control how messages should be protected when being put, browsed, or destructively removed from queues.

The policy name associates digital signing and encryption protection for messages with queues matching the policy name.

**Parameters**

<i>Table 277. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>OUTPUT</u>	Output	<b>*</b> , *PRINT	Optional, Positional 1
<u>POLICY</u>	Policy name	Character value	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 3

**Output (OUTPUT)**

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

**Policy name (POLICY)**

Specifies the name of the security policy, the name of the policy matches the name of the queue that the policy applies to.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

## IBM i DSPMQMSTS (Display Queue Manager Status)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Display Message Queue Manager Status (DSPMQMSTS) command displays the status attributes of the specified local queue manager.

### Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , *DFT	Optional, Key, Positional 1
<u>OUTPUT</u>	Output	*, *PRINT	Optional, Positional 2

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

#### \*

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

#### \*PRINT

The output is printed with the job's spooled output.

## IBM i DSPMQMSUB (Display MQ Subscription)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Display MQ Subscription (DSPMQMSUB) command displays the attributes of an existing MQ subscription.

## Parameters

Keyword	Description	Choices	Notes
<u>SUBID</u>	Subscription identifier	Character value, <b>*NONE</b>	Optional, Positional 1
<u>SUBNAME</u>	Subscription name	Character value, <b>*NONE</b>	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 3
<u>OUTPUT</u>	Output	<b>*</b> , <b>*PRINT</b>	Optional, Positional 4

### Subscription identifier (SUBID)

The subscription identifier of the subscription to be displayed.

The possible values are:

#### subscription-name

Specify a maximum of 256 bytes for the subscription name.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

### Subscription name (SUBNAME)

The name of the subscription to be displayed.

The possible values are:

#### subscription-name

Specify a maximum of 256 bytes for the subscription name.

**Note:** Subscription names of greater than 256 bytes can be specified using MQSC.

### Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

#### \*DFT

Use the default Queue Manager.

#### queue-manager-name

The name of a Queue Manager.

### Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

#### \*PRINT

The output is printed with the job's spooled output.

## IBM i DSPMQMSVC (Display MQ Service)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Display MQ Service object (DSPMQMSVC) command displays the attributes of an existing MQ service object.

### Parameters

Keyword	Description	Choices	Notes
<u>SVCNAME</u>	Service name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Key, Positional 2
<u>OUTPUT</u>	Output	*, *PRINT	Optional, Positional 3

### Service name (SVCNAME)

The name of the service object to display.

The possible values are:

#### service-name

Specify the name of the service definition. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.

### Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

#### \*

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

#### \*PRINT

The output is printed with the job's spooled output.

## IBM i DSPMQMTOP (Display MQ Topic)

### Where allowed to run

All environments (\*ALL)



**Threadsafe**

Yes

The Display MQ Topic (DSPMQMTOP) command displays the attributes of an existing MQ topic object.

**Parameters**

*Table 281. Command parameters*

Keyword	Description	Choices	Notes
<u>TOPNAME</u>	Topic name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 2
<u>OUTPUT</u>	Output	<i>*, *PRINT</i>	Optional, Positional 3

**Topic name (TOPNAME)**

The name of the topic object to be displayed.

The possible values are:

**topic-name**

Specify the name of the topic object. The maximum length of the string is 48 bytes.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the Queue Manager.

The possible values are:

**\*DFT**

Use the default Queue Manager.

**queue-manager-name**

The name of a Queue Manager.

**Output (OUTPUT)**

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

 **DSPMQMVER (Display MQ Version)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Display MQ Version (DSPMQMVER) command provides the current MQ version.

## Parameters

Table 282. Command parameters			
Keyword	Description	Choices	Notes
<u>OUTPUT</u>	Output	<b>*</b> , *PRINT	Optional, Positional 1

### Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

The output is printed with the job's spooled output.

## ENDMQM (End Message Queue Manager)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The End Message Queue Manager (**ENDMQM**) command ends the specified local message queue manager or all queue managers. The attributes of the message queue managers are not affected and it can be restarted using the Start Message Queue Manager (**STRMQM**) command.

You can also use this command to fully quiesce all application programs connected to the queue manager or all queue managers.

The **ENDMQM** command's default parameters should not be changed with the CHGCMDDEF (Change Command Default) command.

## Parameters

Table 283. Command parameters			
Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 1
<u>OPTION</u>	Option	<b>*CNTRL</b> D, *IMMED, *WAIT, *PREEMPT	Optional, Positional 2
<u>INSTANCE</u>	Instance To End	<b>*ALL</b> , *STANDBY	Optional, Positional 3
<u>ALWSWITCH</u>	Allow Switchover	<b>*NO</b> , *YES	Optional, Positional 4
<u>RECONN</u>	Reconnect	<b>*NO</b> , *YES	Optional, Positional 5
<u>ENDCCTJOB</u>	End connected jobs	<b>*NO</b> , *YES	Optional, Positional 6
<u>RCDMQMIMG</u>	Record MQ Object Image	*NO, <b>*YES</b>	Optional, Positional 7
<u>TIMEOUT</u>	Timeout interval (seconds)	0-3600, <b>30</b>	Optional, Positional 8

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.

### **queue-manager-name**

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### **\*ALL**

All queue managers are ended.

## Option (OPTION)

Specifies whether processes that are connected to the queue manager are allowed to complete.

The possible values are:

### **\*CNTRLD**

Allow programs currently being processed to complete. An MQCONN call (or an MQOPEN or MQPUT1, which perform an implicit connection) fails. If ENDCCTJOB(\*YES) is specified, a controlled shutdown of the queue manager is attempted ten times. If the queue manager shuts down successfully, it is followed by immediate termination of the processes that are still connected to it.

### **\*IMMED**

End the queue manager immediately. All current MQI calls complete, but subsequent requests for MQI calls fail. Incomplete units of work are rolled back when the queue manager is next started. If ENDCCTJOB(\*YES) is specified, a controlled shutdown of the queue manager is followed if necessary, after an interval of TIMEOUT seconds, by an immediate shutdown of the queue manager. This is followed by immediate termination of processes connected to it.

### **\*WAIT**

End the queue manager in the same way as the \*CNTRLD option. However, control is returned only after the queue manager has stopped. This option is not allowed with MQMNAME(\*ALL). If ENDCCTJOB(\*YES) is specified, a single controlled shutdown of the queue manager is issued, which waits for all processes to disconnect. When this completes it is followed by the actions described in the ENDCCTJOB parameter.

### **\*PREEMPT**

**Use this type of shutdown only in exceptional circumstances** The queue manager stops without waiting for applications to disconnect or for MQI calls to complete. This can give unpredictable results for IBM MQ applications. All processes in the queue manager that fail to stop are ended 30 seconds after the command is issued. This option is not allowed with ENDCCTJOB(\*YES).

## Instance To End (INSTANCE)

Specifies whether to end all instances of a queue manager, or to end just a standby queue manager instance.

The possible values are:

### **\*ALL**

All instances of a queue manager are to be ended. This option can only be requested against a non-standby queue manager instance.

If a standby instance is running elsewhere, the ALWSWITCH parameter on the ENDMQM command will control whether the standby instance is itself ended.

**\*STANDBY**

Only the standby queue manager instance should be ended, any active queue manager instance will continue to run. This option can only be requested against a standby queue manager instance.

**Allow Switchover (ALWSWITCH)**

Specifies whether switchover to a standby instance of the queue manager is allowed when the active queue manager instance has ended.

The possible values are:

**\*NO**

Switchover to a standby queue manager instance is not allowed. Any standby instances that are running will also end on successful completion of this command. P.: Reconnectable client applications connected to this queue manager are instructed to disconnect.

**\*YES**

Switchover to a standby queue manager instance is attempted, if a standby queue manager instance is not running this command will fail and the active queue manager instance will remain active.

Reconnectable client applications connected to this queue manager instance are instructed to begin reconnect processing, to maintain connectivity.

**Reconnect (RECONN)**

Specifies whether client applications currently connected to this queue manager should attempt to reconnect to a queue manager instance.

The possible values are:

**\*NO**

Reconnectable client applications connected to this queue manager are instructed to disconnect.

**\*YES**

Reconnectable client applications connected to this queue manager are instructed to begin reconnect processing, to maintain connectivity.

**End connected jobs (ENDCCTJOB)**

Specifies whether all processes connected to the queue manager are forcibly terminated.

The possible values are:

**\*NO**

The queue manager or queue managers are ended but no further action is taken.

**\*YES**

The following steps are taken for each queue manager to be ended:

- If the queue manager is running and RCDMQMIMG(\*YES) has been specified, media images for all objects defined for the queue manager are recorded.
- The queue manager is ended in the appropriate manner (\*CNTRLD, \*WAIT, or \*IMMED).
- All shared memory and semaphores used by the queue manager are deleted irrespective of whether applications have disconnected from the queue manager. Applications that have not disconnected from a shared memory resource when this option is specified receive a return code of MQRC\_CONNECTION\_BROKEN (2009) the next time an MQI call is issued with an existing connection handle.

**Record MQ Object Image (RCDMQMIMG)**

Specifies whether media images are recorded for a queue manager.

The possible values are:

**\*YES**

If the queue manager is running, media images for all queue manager objects are recorded.

**\*NO**

Media images of queue manager objects are not recorded as part of the quiesce.

**Timeout interval (seconds) (TIMEOUT)**

Specifies the time interval in seconds between the controlled and immediate shutdowns of the queue manager when \*IMMED is specified. It also determines the number of seconds between attempts to shut down the queue manager when \*CNTRLD is specified.

The possible values are:

**30**

The default value is 30 seconds.

**timeout-interval**

Specify a value in seconds, in the range 0 through 3600.

 **ENDMQMBRK (End MQ Pub/Sub Broker)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The End IBM MQ Broker (ENDMQMBRK) command is used to stop a broker.

**Parameters**

*Table 284. Command parameters*

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1
<u>OPTION</u>	Option	<b>*CNTRLD</b> , *IMMED	Optional, Positional 2

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Option (OPTION)**

Specifies how the broker is ended.

The possible values are:

**\*CNTRLD**

Allows the broker to complete processing for any message that it has already started.

**\*IMMED**

Ends the broker immediately. The broker does not attempt any further gets or puts, and backs out any in-flight units-of-work. This might mean that a nonpersistent input message is published only to a subset of subscribers, or lost, depending on the broker configuration parameters.

## IBM i ENDMQMCHL (End MQ Channel)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The End MQ Channel (ENDMQMCHL) command closes an MQ channel, and the channel is no longer enabled for automatic restarts.

### Parameters

Keyword	Description	Choices	Notes
<u>CHLNAME</u>	Channel name	Character value	Required, Positional 1
<u>OPTION</u>	Option	<b>*CNTRLD</b> , *IMMED, *ABNORMAL	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 3
<u>STATUS</u>	Channel status	<b>*STOPPED</b> , *INACTIVE	Optional, Positional 4
<u>CONNNAME</u>	Connection name	Character value, <b>*NONE</b>	Optional, Positional 5
<u>RQMNAME</u>	Remote queue manager	Character value, <b>*NONE</b>	Optional, Positional 6

### Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

#### channel-name

Specify the channel name.

### Option (OPTION)

Specifies whether processing for the current batch of messages is allowed to finish in a controlled manner.

The possible values are:

#### \*CNTRLD

Allows processing of the current batch of messages to complete. No new batch is allowed to start.

#### \*IMMED

Ends processing of the current batch of messages immediately. This is likely to result in 'in-doubt' situations.

#### \*ABNORMAL

Ends processing of the current batch of messages immediately and terminates the channel thread or job. This is likely to result in 'in-doubt' situations.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

**message-queue-manager-name**

The name of a message queue manager.

**Channel status (STATUS)**

Specifies the required status of the channel after successful completion of the command.

The possible values are:

**\*STOPPED**

The channel status is set to STOPPED.

**\*INACTIVE**

The channel status is set to INACTIVE.

**Connection name (CONNAME)**

Specifies the connection name of the channel instance that you want to end.

**Remote queue manager (RQMNAME)**

Specifies the name of the remote queue manager of the channel instance that you want to end.

 **ENDMQMCONN (End Queue Manager Connection)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The End MQ Connections (ENDMQMCONN) command allows you to end a connection to the queue manager.

**Parameters**

<i>Table 286. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>CONN</u>	Connection Identifier	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 2

**Connection Identifier (CONN)**

The connection identifier to end.

The connection identifier is a 16 character hex string.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of a message queue manager.

 **ENDMQMSVR (End MQ Command Server)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The End MQ Command Server (ENDMQMSVR) command stops the MQ command server for the specified local queue manager.

**Parameters**

<i>Table 287. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1
<u>OPTION</u>	Option	<b>*CNTRLD</b> , <b>*IMMED</b>	Optional, Positional 2

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Option (OPTION)**

Specifies whether the command message currently being processed is allowed to complete.

The possible values are:

**\*CNTRLD**

Allows the command server to complete processing any command message that it has already started. No new message is read from the queue.

**\*IMMED**

Ends the command server immediately. Any action associated with a command message currently being processed might not be completed.

 **ENDMQMLSR (End MQ Listeners)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The End MQ Listener (ENDMQMLSR) command ends an MQ TCP/IP listener.

This command is valid only for TCP/IP transmission protocols.

Either a listener object or specific port can be specified.



## Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 1
<u>PORT</u>	Port number	1-65535, *ALL	Optional, Positional 2
<u>OPTION</u>	Option	<b>*CNTRLD</b> , *WAIT, *FORCE	Optional, Positional 3
<u>LSRNAME</u>	Listener name	<i>Character value</i> , <b>*NONE</b>	Optional, Positional 4

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

#### **queue-manager-name**

The name of a message queue manager.

### Port number (PORT)

The port number to be used by the listener.

The possible values are:

#### **\*SAME**

The attribute is unchanged.

#### **port-number**

The port number to be used.

### Option (OPTION)

Specifies the action taken after processes to end the listeners have been started.

#### **\*CNTRLD**

Processes are started to end all the listeners for the specified queue manager and control is returned before the listeners actually end.

#### **\*WAIT**

End the listeners for the specified queue manager in the same way as the \*CNTRLD option. However, control is returned only after all the listeners have ended.

### Listener name (LSRNAME)

The name of the MQ listener object to end.

The possible values are:

#### **\*NONE**

No listener object is specified.

#### **listener-name**

Specify the name of the listener definition. The maximum length of the string is 48 bytes.

## IBM i ENDMQMSVC (End MQ Service)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The End MQ Service (ENDMQMSVC) command ends an MQ service.

### Parameters

Keyword	Description	Choices	Notes
<u>SVCNAME</u>	Service name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 2

### Service name (SVCNAME)

The name of the MQ service object to end.

The possible values are:

#### \*NONE

No service object is specified.

#### service-name

Specify the name of the service definition. The maximum length of the string is 48 bytes.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

The name of a message queue manager.

## IBM i GRMQMAUT (Grant MQ Object Authority)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Grant MQ Authority (GRMQMAUT) command is used to grant specific authority for the MQ objects named in the command to another user or group of users.

Authority can be given to:

- Named users.
- Users (\*PUBLIC) who do not have authority specifically given to them.
- Groups of users who do not have any authority to the object.

The GRMQMAUT command can be used by anyone in the QMQMADM group, that is, anyone whose user profile specifies QMQMADM as a primary or supplemental group profile.

## Parameters

Table 290. Command parameters			
Keyword	Description	Choices	Notes
<u>OBJ</u>	Object name	Character value	Required, Positional 1
<u>OBJTYPE</u>	Object type	*ALL, *Q, *ALSO, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC, *RMTMQMNAME	Required, Positional 2
<u>USER</u>	User names	Single values: *PUBLIC, Other values (up to 50 repetitions): <i>Name</i>	Required, Positional 3
<u>AUT</u>	Authority	Values (up to 22 repetitions): *ALTUSR, *BROWSE, *CONNECT, *GET, *INQ, *PUT, *SET, *PUB, *SUB, *RESUME, *PASSALL, *PASSID, *SETALL, *SETID, *ADMCHG, *ADMCLR, *ADMCRRT, *ADMDLT, *ADMDSP, *ALL, *ALLADM, *ALLMQI, *NONE, *CTRL, *CTRLX, *SYSTEM	Required, Positional 4
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 5
<u>SRVCOMP</u>	Service Component name	Character value, <b>*DFT</b>	Optional, Positional 6

### Object name (OBJ)

Specifies the name of the objects for which specific authorities are granted.

The possible values are:

#### **\*ALL**

All objects of the type specified by the value of the OBJTYPE parameter at the time the command is issued. \*ALL cannot represent a generic profile.

#### **object-name**

Specify the name of an MQ object for which specific authority is given to one or more users.

#### **generic profile**

Specify the generic profile of the objects to be selected. A generic profile is a character string containing one or more generic characters anywhere in the string. This profile is used to match the object name of the object under consideration at the time of use. The generic characters are (?), (\*) and (\*\*).

? matches a single character in an object name.

\* matches any string contained within a qualifier, where a qualifier is the string between periods (.). For example ABC\* matches ABCDEF but not ABCDEF.XYZ.

\*\* matches one or more qualifiers. For example ABC.\*\*.XYZ matches ABC.DEF.XYZ and ABC.DEF.GHI.XYZ, \*\* can appear only once in a generic profile.

Specify the name required within quotation marks to ensure that your selection is precisely what you entered.

## Object type (OBJTYPE)

Specifies the type of the objects for which specific authorities are granted.

### **\*ALL**

All MQ object types.

### **\*Q**

All queue object types.

### **\*ALSQ**

Alias queue.

### **\*LCLQ**

Local queue.

### **\*MDLQ**

Model queue.

### **\*RMTQ**

Remote queue.

### **\*AUTHINFO**

Authentication Information object.

### **\*MQM**

Message Queue Manager.

### **\*NMLIST**

Namelist object.

### **\*PRC**

Process definition.

### **\*CHL**

Channel object.

### **\*CLTCN**

Client Connection Channel object.

### **\*LSR**

Listener object.

### **\*SVC**

Service object.

### **\*TOPIC**

Topic object.

### **\*RMTMQMNAME**

Remote queue manager name.

## User names (USER)

Specifies the name or names of users to whom authorities for the named object are being given. If user names are specified, the authorities are given specifically to those users. Authority given by this command can be revoked specifically by the Revoke MQ Authority (RVKMQMAUT) command.

### **\*PUBLIC**

All users of the system.

### **user-profile-name**

Specify the names of one or more users who are to be granted specific authority for the object. These names can also be group names. You can specify up to 50 user profile names.

## Authority (AUT)

Specifies the authority being given to the named users. Values for AUT can be specified as a list of specific and general authorities in any order, where the general authorities can be:

\*NONE, which creates a profile for the user with no authority to the specified object, or leaves the authority unchanged if a profile already exists.

\*ALL, which confers all authorities to the specified users.

\*ALLADM, which confers all of \*ADMCHG, \*ADMCLR, \*ADMCRRT, \*ADMDLT, \*ADMDSP, \*CTRL and \*CTRLX.

\*ALLMQI, which confers all of \*ALTUSR, \*BROWSE, \*CONNECT, \*GET, \*INQ, \*PUT, \*SET, \*PUB, \*SUB and \*RESUME.

Authorizations for different object types

### **\*ALL**

All authorizations. Applies to all objects.

### **\*ADMCHG**

Change an object. Applies to all objects except remote queue manager name.

### **\*ADMCLR**

Clear a queue. Applies to queues only.

### **\*ADMCRRT**

Create an object. Applies to all objects except remote queue manager name.

### **\*ADMDLT**

Delete an object. Applies to all objects except remote queue manager name.

### **\*ADMDSP**

Display the attributes of an object. Applies to all objects except remote queue manager name.

### **\*ALLADM**

Perform administration operations on an object. Applies to all objects except remote queue manager name.

### **\*ALLMQI**

Use all MQI calls applicable to an object. Applies to all objects.

### **\*ALTUSR**

Allow another user's authority to be used for MQOPEN and MQPUT1 calls. Applies to queue manager objects only.

### **\*BROWSE**

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option. Applies to queue objects only.

### **\*CONNECT**

Connect the application to a queue manager by issuing an MQCONN call. Applies to queue manager objects only.

### **\*CTRL**

Control startup and shutdown of channels, listeners and services.

### **\*CTRLX**

Reset sequence number and resolve indoubt channels.

### **\*GET**

Retrieve a message from a queue using an MGET call. Applies to queue objects only.

### **\*INQ**

Make an inquiry on an object using an MQINQ call. Applies to all objects except remote queue manager name.

### **\*PASSALL**

Pass all context on a queue. Applies to queue objects only.

**\*PASSID**

Pass identity context on a queue. Applies to queue objects only.

**\*PUT**

Put a message on a queue using an MQPUT call. Applies to queue objects and remote queue manager names only.

**\*SET**

Set the attributes of an object using an MQSET call. Applies to queue, queue manager, and process objects only.

**\*SETALL**

Set all context on an object. Applies to queue and queue manager objects only.

**\*SETID**

Set identity context on an object. Applies to queue and queue manager objects only.

**\*SYSTEM**

Connect the application to a queue manager for system operations. Applies to queue manager objects only.

Authorizations for MQI calls

**\*ALTUSR**

Allow another user's authority to be used for MQOPEN and MQPUT1 calls.

**\*BROWSE**

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

**\*CONNECT**

Connect the application to the specified queue manager by issuing an MQCONN call.

**\*GET**

Retrieve a message from a queue by issuing an MQGET call.

**\*INQ**

Make an inquiry on a specific queue by issuing an MQINQ call.

**\*PUT**

Put a message on a specific queue by issuing an MQPUT call.

**\*SET**

Set attributes on a queue from the MQI by issuing an MQSET call.

**\*PUB**

Open a topic to publish a message using the MQPUT call.

**\*SUB**

Create, Alter or Resume a subscription to a topic using the MQSUB call.

**\*RESUME**

Resume a subscription using the MQSUB call.

If you open a queue for multiple options, you must be authorized for each of them.

Authorizations for context

**\*PASSALL**

Pass all context on the specified queue. All the context fields are copied from the original request.

**\*PASSID**

Pass identity context on the specified queue. The identity context is the same as that of the request.

**\*SETALL**

Set all context on the specified queue. This is used by special system utilities.

**\*SETID**

Set identity context on the specified queue. This is used by special system utilities.

Authorizations for MQSC and PCF commands

**\*ADMCHG**

Change the attributes of the specified object.

**\*ADMCLR**

Clear the specified queue (PCF Clear queue command only).

**\*ADMCR**

Create objects of the specified type.

**\*ADMDLT**

Delete the specified object.

**\*ADMDS**

Display the attributes of the specified object.

**\*CTRL**

Control startup and shutdown of channels, listeners and services.

**\*CTRLX**

Reset sequence number and resolve indoubt channels.

Authorizations for generic operations

**\*ALL**

Use all operations applicable to the object.

all authority is equivalent to the union of the authorities alladm, allmqi, and system appropriate to the object type.

**\*ALLADM**

Perform all administration operations applicable to the object.

**\*ALLMQI**

Use all MQI calls applicable to the object.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Service Component name (SRVCOMP)**

Specifies the name of the installed authorization service to which the authorizations apply.

The possible values are:

**\*DFT**

Use the first installed authorization component.

**Authorization-service-component-name**

The component name of the required authorization service as specified in the queue manager qm.ini file.

 **PNGMQMCHL (Ping MQ Channel)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Ping MQ Channel (PNGMQMCHL) command tests a channel by sending data as a special message, to the remote message queue manager and checks that the data is returned. This command is successful

only from the sending end of an inactive channel, and the data used is generated by the local message queue manager.

## Parameters

Keyword	Description	Choices	Notes
<u>CHLNAME</u>	Channel name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 2
<u>DATACNT</u>	Data count	16-32768, <b>64</b>	Optional, Positional 3
<u>CNT</u>	Count	1-16, <b>1</b>	Optional, Positional 4

### Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

#### **channel-name**

Specify the channel name.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### **\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

#### **message-queue-manager-name**

The name of a message queue manager.

### Data count (DATACNT)

Specifies the length of the data in bytes. The actual number of bytes might be less than the amount requested depending on the operating system and communication protocol being used.

The possible values are:

#### **64**

The default value is 64 bytes.

*data-count* Specify a value ranging from 16 through 32768.

### Count (CNT)

Specifies the number of times that the channel is to be pinged.

The possible values are:

#### **1**

The channel is pinged once.

*ping-count* Specify a value ranging from 1 through 16.



**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Record MQ Object Image (RCDMQMIMG) command is used to provide a marker for the selected set of MQ objects, so that the Re-create MQM Object (RCRMQMOBJ) command can recover this set of objects from journal data recorded subsequently.

This command is intended to enable journal receivers, detached prior to the current date, to be disconnected. On successful completion of this command those journals are no longer required to be present for a Re-create MQ Object (RCRMQMOBJ) command on this set of MQM Objects to succeed.

**Parameters**

Keyword	Description	Choices	Notes
<u>OBJ</u>	Object name	Character value, *ALL	Required, Positional 1
<u>OBJTYPE</u>	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *MQM, *NMLIST, *PRC, *CHL, *CLTCN, *LSR, *SVC, *SYNCFILE, *TOPIC	Required, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 3
<u>DSPJRNDTA</u>	Display Journal Receiver Data	*YES, *NO	Optional, Positional 4

**Object name (OBJ)**

Specifies the name of the objects that should be recorded. This is a 48-character MQ object or generic object name.

The possible values are:

**\*ALL**

All MQ objects of the specified type (OBJTYPE) are recorded.

**generic-object-name**

Specify the generic name of the objects to be recorded. A generic name is a character string followed by an asterisk (\*). For example, ABC\*. It selects all objects that have names which start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**object-name**

The name of an MQ object to be recorded.

**Object type (OBJTYPE)**

Specifies the type of the objects to be re-created.

The possible values are:

**\*ALL**

Specifies all MQ object types.

**\*Q**

Specifies MQ queue objects with names specified by OBJ.

**\*ALSQ**

Specifies MQ alias queue objects with names specified by OBJ.

**\*LCLQ**

Specifies MQ local queue objects with names specified by OBJ.

**\*MDLQ**

Specifies MQ model queues objects with names specified by OBJ.

**\*RMTQ**

Specifies MQ remote queue objects with names specified by OBJ.

**\*AUTHINFO**

Specifies MQ authentication information objects with names specified by OBJ.

**\*CTLG**

Specifies the MQ queue manager catalog object. This has the same name as the queue manager object.

**\*MQM**

Specifies the Message Queue Manager object.

**\*CHL**

Specifies MQ channel objects with names specified by OBJ.

**\*CLTCN**

Specifies MQ MQI client connection channel objects with names specified by OBJ.

**\*NMLIST**

Specifies MQ namelist objects with names specified by OBJ.

**\*PRC**

Specifies MQ process objects with names specified by OBJ.

**\*LSR**

Specifies MQ listener objects with names specified by OBJ.

**\*SVC**

Specifies MQ service objects with names specified by OBJ.

**\*SYNCFILE**

Specifies the MQ channel synchronisation file.

**\*TOPIC**

Specifies the MQ topic objects with names specified by OBJ.

### **Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**message-queue-manager-name**

Specify the name of the queue manager.

### **Display Journal Receiver Data (DSPJRNDTA)**

Specifies whether additional messages should be written to the job log when the command completes to inform the user which journal receivers are still required by IBM MQ.

The possible values are:

**\*NO**

No messages are written to the job log.

**\*YES**

Messages will be sent to the job log when the command completes. The messages will contain details about which journal receivers are required by IBM MQ.

**IBM i RCRMQMOBJ (Re-create MQ Object)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Re-create MQ Object (RCRMQMOBJ) command is used to provide a recovery mechanism for damaged MQ objects. The command completely re-creates the objects from information recorded in the MQ journals. If no damaged objects exist, no action is performed.

**Parameters**

Table 293. Command parameters

Keyword	Description	Choices	Notes
<u>OBJ</u>	Object name	Character value, *ALL	Required, Positional 1
<u>OBJTYPE</u>	Object type	*ALL, *Q, *ALSOQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *MQM, *NMLIST, *PRC, *CHL, *CLTCN, *LSR, *SVC, *SYNCFILE, *CLCHLTAB, *TOPIC	Required, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 3

**Object name (OBJ)**

Specifies the name of the objects which should be re-created if they are damaged. This is a 48-character MQ object or generic object name.

The possible values are:

**\*ALL**

All damaged MQ objects of the specified type (OBJTYPE) are re-created.

**generic-object-name**

Specify the generic name of the objects to be re-created. A generic name is a character string followed by an asterisk (\*). For example, ABC\*. It selects all objects that have names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**object-name**

The name of an MQ object to be re-created if it is damaged.

## Object type (OBJTYPE)

Specifies the object type of the objects to be re-created.

The possible values are:

### **\*ALL**

Specifies all MQ object types.

### **\*Q**

Specifies MQ queue objects with names specified by OBJ.

### **\*ALSQ**

Specifies MQ alias queue objects with names specified by OBJ.

### **\*LCLQ**

Specifies MQ local queue objects with names specified by OBJ.

### **\*MDLQ**

Specifies MQ model queues with names specified by OBJ.

### **\*RMTQ**

Specifies MQ remote queue objects with names specified by OBJ.

### **\*AUTHINFO**

Specifies MQ authentication information objects with names specified by OBJ.

### **\*CTLG**

Specifies the message queue manager catalog object. The catalog object has the same name as the message queue manager object. It holds the names of MQ objects. A user needs authorities on this object to be able to start or stop the message queue manager, or, to create or delete MQ queues and process definitions.

### **\*MQM**

Specifies the message queue manager. This object holds the attributes of the message queue manager.

### **\*CHL**

Specifies MQ channel objects with names specified by OBJ.

### **\*CLTCN**

Specifies MQ MQI client connection channel objects with names specified by OBJ.

### **\*NMLIST**

Specifies MQ namelist objects with names specified by OBJ.

### **\*PRC**

Specifies MQ process objects with names specified by OBJ.

### **\*LSR**

Specifies MQ listener objects with names specified by OBJ.

### **\*SVC**

Specifies MQ service objects with names specified by OBJ.

### **\*SYNCFILE**

Specifies the MQ channel synchronisation file.

### **\*SYNCFILE**

Specifies the MQ MQI client channel table file.

### **\*TOPIC**

Specifies MQ topic objects with names specified by OBJ.

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**message-queue-manager-name**

Specify the name of the queue manager.


**RFRMQM (Refresh Message Queue Manager)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Refresh Message Queue manager (RFRMQM) performs special operations on queue managers.

**Parameters**

*Table 294. Command parameters*

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Required, Positional 1
<u>TYPE</u>	Refresh Type	<b>*CONFIGEV</b> , <i>*PROXYSUB</i>	Required, Positional 2
<u>OBJECT</u>	Object Type	<b>*ALL</b> , <i>Specified objects</i>	Optional, Positional 3
<u>NAME</u>	Object Name	<b>*ALL</b> , <i>generic- object-name, object-name</i>	Optional, Positional 4
<u>INCLINT</u>	Include Interval	<b>*NONE</b> , <i>include- interval</i>	Optional, Positional 5

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue\_manager\_name**

Specify the name of the queue manager.

**Refresh Type (TYPE)**

The type of queue manager refresh to be performed.

The possible values are:

**\*CONFIGEV**

Requests that the queue manager generates a configuration event message for every object that matches the selection criteria specified by the OBJECT, NAME, and INCLINT parameters.

**\*PROXYSUB**

Requests that the queue manager resynchronizes the proxy subscriptions that are held with, and on behalf of, queue managers that are connected in a hierarchy or publish/subscribe cluster.

**Object Type (OBJECT)**

Requests that only objects of the specified type are included in the refresh.

This parameter is only valid for TYPE(\*CONFIGEV)

The possible values are:

**\*ALL**

All specified objects.

**Specific objects**

Select from:

- \*QUEUE
- \*QLOCAL
- \*QMODEL
- \*QALIAS
- \*QREMOTE
- \*CHANNEL
- \*NAMELIST
- \*POLICY
- \*PROCESS
- \*QMGR
- \*AUTHINFO
- \*AUTHREC

**Object Name (NAME)**

Requests that only objects whose names match the name specified are included in the refresh.

This parameter is only valid for TYPE(\*CONFIGEV)

The possible values are:

**\*ALL**

All object names are included.

***generic-object-name***

Specify the generic name of the objects to be included. A generic name is a character string, followed by an asterisk (\*), for example ABC\*, and it selects all queues having names that start with the character string.

***object-name***

Specify the object name to be included.

**Include Interval (INCLINT)**

Specifies a value in minutes, defining a period immediately before the current time, and requests that only objects that have been created or changed within that period are included in the refresh.

This parameter is only valid for TYPE(\*CONFIGEV)

The possible values are:

**\*NONE**

No time limit is used.

***include-interval***

Specify the include interval in minutes (0-999999).

 **RRMQMAUT (Refresh IBM MQ Authority)****Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The IBM MQ security cache refresh (RFRMQMAUT) command refreshes the IBM MQ object authority manager security cache.

## Parameters

Table 295. Command parameters			
Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 1
<u>TYPE</u>	Refresh Type	<b>*AUTHSERV</b> , *SSL	Optional, Positional 2

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager to perform the security refresh.

The possible values are:

#### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

#### \*DFT

Specifies that the default queue manager should be used.

### Refresh Type (TYPE)

The type of security refresh to be performed. The possible values are:

#### \*AUTHSERV

Refreshes the list of authorizations held internally by the authorization services component.

#### \*SSL

Refreshes the cached view of the TLS Key Repository allowing updates to become effective when the command has completed successfully. Also refreshes the locations of the LDAP servers to be used for Certificate Revocation Lists and the Key Repository.

## RFRMQMCL (Refresh MQ Cluster)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Refresh MQ Cluster (RFRMQMCL) command refreshes locally held cluster information (including any autodefined channels that are in doubt), and forces it to be rebuilt. This enables you to perform a "cold-start" on the cluster.

## Parameters

Table 296. Command parameters			
Keyword	Description	Choices	Notes
<u>CLUSTER</u>	Cluster Name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>REPOS</u>	Refresh Repository	<b>*NO</b> , *YES	Optional, Positional 3

## Cluster Name (CLUSTER)

The name of the cluster to be refreshed.

The possible values are:

**\*\***

The queue manager is refreshed in all of the clusters to which it belongs.

If Refresh Repository is also set to \*YES, then the queue manager restarts its search for repository queue managers, using information in the local cluster-sender channel definitions.

**name**

Specify the name of the cluster.

## Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

## Refresh Repository (REPOS)

Specifies whether the information about repository queue managers should be refreshed.

The possible values are:

**\*NO**

Do not refresh repository information.

**\*YES**

Refresh repository information. This value cannot be specified if the queue manager is itself a repository manager.

## RMVMQMINF (Remove Queue Manager Info.)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Remove Message Queue Manager Information (RMVMQMINF) command removes configuration information for a queue manager. This command can be used, for example, to remove a secondary queue manager instance by removing reference to shared queue manager data.

## Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value	Optional, Positional 1

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager to remove information for.



### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

## IBM i RMVMQMJRN (Remove Queue Manager Journal)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Remove Queue Manager Journal command (RMVMQMJRN) removes a queue manager journal. This command can be used, for example, to remove a remote journal previously used for a standby or multi-instance queue manager.

### Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 1
<u>JRN</u>	Queue Manager Journal	Character value, *DFT	Optional, Positional 2
<u>RMTJRNRDB</u>	Remote Relational Database	Character value	Optional, Positional 3

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager associated with the journal.

#### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

### Queue Manager Journal (JRN)

Specifies the journal name to create.

The possible values are:

#### \*DFT

The journal name is chosen by the system. If a local journal already exists for the queue manager on this system - the existing local journal name is used, otherwise a unique name is generated of the format AMQxJRN where x is a character in the range 'A - Z'.

#### journal-name

Specify the name of the journal. The name can contain up to 10 characters. Journal receiver names will be derived from this journal name by truncating at the 4th character (or at the last character if the journal name is shorter than 4 characters) and appending zeroes. If the local queue manager library already contains a local journal, its name must match that supplied. Only one local journal can exist in a queue manager library. DLTMQM will not remove journal artifacts from a queue manager library unless they are prefixed with "AMQ".

### Remote Relational Database (RMTJRNRDB)

Specifies the name of the relational database directory entry that contains the remote location name of the target system. Use the WRKRDBDIRE command to locate and existing entry or configure a new relational database directory entry for the target system.

## relational-database-directory-entry

Specify the name of the relational database directory entry. The name can contain up to 18 characters.

## IBM i RSMQMCLQM (Resume Cluster Queue Manager)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

Use the RSMQMCLQM command to inform other queue managers in a cluster that the local queue manager is again available for processing and can be sent messages. It reverses the action of the SPDMQMCLQM command.

## Parameters

Keyword	Description	Choices	Notes
<u>CLUSTER</u>	Cluster Name	Character value	Optional, Positional 1
<u>CLUSNL</u>	Cluster Name List	Character value	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 3

### Cluster Name (CLUSTER)

Specifies the name of the cluster for which the queue manager is available for processing.

#### cluster-name

Specify the name of the cluster.

### Cluster Name List (CLUSNL)

Specifies the namelist specifying a list of clusters for which the queue manager is available for processing.

#### namelist

Specify the name of the namelist.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

#### \*DFT

Use the default queue manager.

#### queue-manager-name

Specify the name of the queue manager.

## IBM i RSTMQMCHL (Reset MQ Channel)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Reset MQ Channel (RSTMQMCHL) command resets the message sequence number for an MQ channel to a specified sequence number for use the next time that the channel is started.

You are recommended to use this command for Sender(\*SDR), Server (\*SVR) and Cluster-sender (\*CLUSSDR) channels only.

If you use this command for a Receiver (\*RCVR), Requester (\*RQSTR) or Cluster-receiver (\*CLUSRCVR) channel, the value at the other end of the channel is NOT reset. You must reset the values separately.

The command does not work for Server-connection (\*SVRCN) channels.

## Parameters

Keyword	Description	Choices	Notes
<u>CHLNAME</u>	Channel name	Character value	Required, Positional 1
<u>MSGSEQNUM</u>	Message sequence number	1-999999999, <b>1</b>	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 3

### Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

#### channel-name

Specify the channel name.

### Message sequence number (MSGSEQNUM)

Specifies the new message sequence number.

The possible values are:

#### 1

The new message sequence number is 1.

#### message-sequence-number

Specify the new message sequence number ranging from 1 through 999999999.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

#### message-queue-manager-name

The name of a message queue manager.

## RSTMQMCL (Reset Cluster)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

Use the Reset Cluster (RSTMQMCL) command to forcibly remove a queue manager from a cluster.

## Parameters

Keyword	Description	Choices	Notes
<u>CLUSTER</u>	Cluster Name	Character value	Required, Positional 1
<u>QMNAME</u>	Queue Manager Name for removal	Character value, *QMID	Required, Positional 2
<u>ACTION</u>	Action	*FRCRMV	Optional, Positional 3
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 4
<u>QUEUES</u>	Remove Queues	*NO, *YES	Optional, Positional 5
<u>QMID</u>	Queue Manager Id for removal	Character value	Optional, Positional 6

### Cluster Name (CLUSTER)

Specifies the name of cluster from which the queue manager is to be forcibly removed.

#### cluster-name

Specify the name of the cluster.

### Queue Manager Name for removal (QMNAME)

Specifies the name of the queue manager to be forcibly removed.

The possible values are:

#### \*QMID

This enables you to specify the identifier of the queue manager to be forcibly removed.

#### queue-manager-name

Specify the name of the queue manager.

### Action (ACTION)

Specifies the action to take on the specified queue manager.

#### \*FRCRMV

Requests that the queue manager is forcibly removed from the cluster. This might be needed to ensure correct cleanup after a queue manager has been deleted. This action can be requested by a repository queue manager only.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

#### \*DFT

Use the default queue manager.

#### queue-manager-name

Specify the name of the queue manager.

### Remove Queues (QUEUES)

Specifies whether cluster queues should be removed from the cluster.

The possible values are:

**\*NO**

Do not remove the queues belonging to the queue manager being removed from the cluster.

**\*YES**

Remove queues belonging to the queue manager being removed from the cluster.

**Queue Manager Id for removal (QMID)**

Specifies the identifier of the queue manager to be forcibly removed.

**queue-manager-identifier**

Specify the identifier of the queue manager.


**RSVMQMCHL (Resolve MQ Channel)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Resolve MQ Channel (RSVMQMCHL) command requests a channel to commit or backout in-doubt messages.

This command is used when the other end of a link fails during the confirmation period, and for some reason it is not possible to reestablish the connection.

In this situation, the sending end remains in an in-doubt state, about whether the messages were received. Any outstanding units of work need to be resolved with either backout or commit.

\*BCK restores messages to the transmission queue and \*CMT discards them.

Use this command for sender (\*SDR) and server (\*SVR) channels only.

**Parameters**

<i>Table 302. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>CHLNAME</u>	Channel name	Character value	Required, Positional 1
<u>OPTION</u>	Resolve option	*CMT, *BCK	Required, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , *DFT	Optional, Positional 3

**Channel name (CHLNAME)**

Specifies the name of the channel definition.

The possible values are:

**channel-name**

Specify the channel name.

**Resolve option (OPTION)**

Specifies whether to back out or commit the messages.

The possible values are:

**\*CMT**

The messages are committed, that is, they are deleted from the transmission queue.

**\*BCK**

The messages are backed out, that is, they are restored to the transmission queue.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

**message-queue-manager-name**

The name of a message queue manager.

 **RUNMQSC (Run MQSC commands)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Run IBM MQ Commands (RUNMQSC) command allows you to issue MQSC commands interactively for the specified queue manager.

**Parameters**

<i>Table 303. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

**queue-manager-name**

Specify the name of the queue manager.

 **RVKMQMAUT (Revoke MQ Object Authority)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Revoke MQ Authority (RVKMQMAUT) command is used to reset, or take away specific or all authority for the named objects from the users named in the command.

The RVKMQMAUT command can be used by anyone in the QMQMADM group, that is, anyone whose user profile specifies QMQMADM as a primary or supplemental group profile.

## Parameters

Table 304. Command parameters			
Keyword	Description	Choices	Notes
<u>OBJ</u>	Object name	Character value	Required, Positional 1
<u>OBJTYPE</u>	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC, *RMTMQMNAME	Required, Positional 2
<u>USER</u>	User names	Single values: *PUBLIC, Other values (up to 50 repetitions): <i>Name</i>	Required, Positional 3
<u>AUT</u>	Authority	Values (up to 22 repetitions): *ALTUSR, *BROWSE, *CONNECT, *GET, *INQ, *PUT, *SET, *PUB, *SUB, *RESUME, *PASSALL, *PASSID, *SETALL, *SETID, *ADMCHG, *ADMCLR, *ADMCRRT, *ADMDLT, *ADMDSP, *ALL, *ALLADM, *ALLMQI, *REMOVE, *CTRL, *CTRLX, *SYSTEM	Required, Positional 4
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 5
<u>SRVCOMP</u>	Service Component name	Character value, <b>*DFT</b>	Optional, Positional 6

### Object name (OBJ)

Specifies the name of the objects for which specific authorities are revoked.

The possible values are:

#### **\*ALL**

All objects of the type specified by the value of the OBJTYPE parameter at the time the command is issued. \*ALL cannot represent a generic profile.

#### **object-name**

Specify the name of an MQ object for which specific authority is given to one or more users.

#### **generic profile**

Specify the generic profile of the objects to be selected. A generic profile is a character string containing one or more generic characters anywhere in the string. This profile is used to match the object name of the object under consideration at the time of use. The generic characters are (?), (\*) and (\*\*).

? matches a single character in an object name.

\* matches any string contained within a qualifier, where a qualifier is the string between fullstops (.). For example ABC\* matches ABCDEF but not ABCDEF.XYZ.

\*\* matches one or more qualifiers. For example ABC.\*\*.XYZ matches ABC.DEF.XYZ and ABC.DEF.GHI.XYZ, \*\* can only appear once in a generic profile.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

## Object type (OBJTYPE)

Specifies the type of the objects for which specific authorities are revoked.

### **\*ALL**

All MQ object types.

### **\*Q**

All queue object types.

### **\*ALSQ**

Alias queue.

### **\*LCLQ**

Local queue.

### **\*MDLQ**

Model queue.

### **\*RMTQ**

Remote queue.

### **\*AUTHINFO**

Authentication Information object.

### **\*MQM**

Message Queue Manager.

### **\*NMLIST**

Namelist object.

### **\*PRC**

Process definition.

### **\*CHL**

Channel object.

### **\*CLTCN**

Client Connection Channel object.

### **\*LSR**

Listener object.

### **\*SVC**

Service object.

### **\*TOPIC**

Topic object.

### **\*RMTMQMNAME**

Remote queue manager name.

## User names (USER)

Specifies the user names of one or more users whose specific authorities to the named object are being removed. If a user was given the authority by USER(\*PUBLIC) being specified in the Grant MQ Authority (GRTMQMAUT) command, the same authorities are revoked by \*PUBLIC being specified in this parameter. Users given specific authority by having their names identified in the GRTMQMAUT command must have their names specified on this parameter to remove the same authorities.

The possible values are:

### **\*PUBLIC**

The specified authorities are taken away from users who do not have specific authority for the object, who are not on the authorization list, and whose user group has no authority. Users who have specific authority still retain their authorities to the object.



## **user-profile-name**

Specify the user names of one or more users who are having the specified authorities revoked. The authorities listed in the AUT parameter are being specifically taken away from each identified user. This parameter cannot be used to remove public authority from specific users; only authorities that were specifically given to them can be specifically revoked. You can specify up to 50 user profile names.

## **Authority (AUT)**

Specifies the authority being reset or taken away from the users specified in the USER parameter. You can specify values for AUT as a list of specific and general authorities in any order, where the general authorities can be:

\*REMOVE, which deletes the profile. It is not the same as \*ALL, because \*ALL leaves the profile in existence with no authorities. \*REMOVE cannot be specified with user QMQMADM unless the object is a generic profile or with user QMQM when the object type is \*MQM.

\*ALL, which confers all authorities to the specified users.

\*ALLADM, which confers all of \*ADMCHG, \*ADMCLR, \*ADMCRRT, \*ADMDLT, \*ADMDSP, \*CTRL and \*CTRLX.

\*ALLMQI, which confers all of \*ALTUSR, \*BROWSE, \*CONNECT, \*GET, \*INQ, \*PUT, \*SET, \*PUB, \*SUB and \*RESUME.

Authorizations for different object types

### **\*ALL**

All authorizations. Applies to all objects.

### **\*ADMCHG**

Change an object. Applies to all objects except remote queue manager name.

### **\*ADMCLR**

Clear a queue. Applies to queues only.

### **\*ADMCRRT**

Create an object. Applies to all objects except remote queue manager name.

### **\*ADMDLT**

Delete an object. Applies to all objects except remote queue manager name.

### **\*ADMDSP**

Display the attributes of an object. Applies to all objects except remote queue manager name.

### **\*ALLADM**

Perform administration operations on an object. Applies to all objects except remote queue manager name.

### **\*ALLMQI**

Use all MQI calls applicable to an object. Applies to all objects.

### **\*ALTUSR**

Allow another user's authority to be used for MQOPEN and MQPUT1 calls. Applies to queue manager objects only.

### **\*BROWSE**

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option. Applies to queue objects only.

### **\*CONNECT**

Connect the application to a queue manager by issuing an MQCONN call. Applies to queue manager objects only.

### **\*CTRL**

Control startup and shutdown of channels, listeners and services.

### **\*CTRLX**

Reset sequence number and resolve indoubt channels.

**\*GET**

Retrieve a message from a queue using an MGET call. Applies to queue objects only.

**\*INQ**

Make an inquiry on an object using an MQINQ call. Applies to all objects except remote queue manager name.

**\*PASSALL**

Pass all context on a queue. Applies to queue objects only.

**\*PASSID**

Pass identity context on a queue. Applies to queue objects only.

**\*PUT**

Put a message on a queue using an MQPUT call. Applies to queue objects and remote queue manager names only.

**\*SET**

Set the attributes of an object using an MQSET call. Applies to queue, queue manager, and process objects only.

**\*SETALL**

Set all context on an object. Applies to queue and queue manager objects only.

**\*SETID**

Set identity context on an object. Applies to queue and queue manager objects only.

**\*SYSTEM**

Connect the application to a queue manager for system operations. Applies to queue manager objects only.

Authorizations for MQI calls

**\*ALTUSR**

Allow another user's authority to be used for MQOPEN and MQPUT1 calls.

**\*BROWSE**

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

**\*CONNECT**

Connect the application to the specified queue manager by issuing an MQCONN call.

**\*GET**

Retrieve a message from a queue by issuing an MQGET call.

**\*INQ**

Make an inquiry on a specific queue by issuing an MQINQ call.

**\*PUT**

Put a message on a specific queue by issuing an MQPUT call.

**\*SET**

Set attributes on a queue from the MQI by issuing an MQSET call.

**\*PUB**

Open a topic to publish a message using the MQPUT call.

**\*SUB**

Create, Alter or Resume a subscription to a topic using the MQSUB call.

**\*RESUME**

Resume a subscription using the MQSUB call.

If you open a queue for multiple options, you must be authorized for each of them.

Authorizations for context

**\*PASSALL**

Pass all context on the specified queue. All the context fields are copied from the original request.

**\*PASSID**

Pass identity context on the specified queue. The identity context is the same as that of the request.

**\*SETALL**

Set all context on the specified queue. This is used by special system utilities.

**\*SETID**

Set identity context on the specified queue. This is used by special system utilities.

Authorizations for MQSC and PCF commands

**\*ADMCHG**

Change the attributes of the specified object.

**\*ADMCLR**

Clear the specified queue (PCF Clear queue command only).

**\*ADMCR**

Create objects of the specified type.

**\*ADMDEL**

Delete the specified object.

**\*ADMDS**

Display the attributes of the specified object.

**\*CTRL**

Control startup and shutdown of channels, listeners and services.

**\*CTRLX**

Reset sequence number and resolve indoubt channels.

Authorizations for generic operations

**\*ALL**

Use all operations applicable to the object.

all authority is equivalent to the union of the authorities alladm, allmqi, and system appropriate to the object type.

**\*ALLADM**

Perform all administration operations applicable to the object.

**\*ALLMQI**

Use all MQI calls applicable to the object.

**\*REMOVE**

Delete the authority profile to the specified object.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Service Component name (SRVCOMP)**

Specifies the name of the installed authorization service to which the authorizations apply.

The possible values are:

**\*DFT**

Use the first installed authorization component.

**Authorization-service-component-name**

The component name of the required authorization service as specified in the Queue manager's qm.ini file.

**Where allowed to run**

All environments (\*ALL)

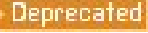




**Threadsafe**

Yes

The Set MQM Security Policy (SETMQMSPL) command sets security policies that are used by Advanced Message Security to control how messages should be protected when being put, browsed, or destructively removed from queues.

The policy name associates digital signing and encryption protection for messages with queues matching the policy name.

**Parameters**

Keyword	Description	Choices	Notes
<u>POLICY</u>	Policy name	Character value	Required, Key, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Required, Key, Positional 2
<u>SIGNALG</u>	Signature algorithm	*NONE,  *MD5,  *SHA1, *SHA256, *SHA384, *SHA512	Optional, Positional 3
<u>ENCALG</u>	Encryption algorithm	*NONE,  *RC2,  *DES,  *TRIPLEDES, *AES128, *AES256	Optional, Positional 4
<u>SIGNER</u>	Authorized signers	*NONE, Character value	Optional, Positional 5
<u>RECIPI</u>	Intended recipients	*NONE, Character value	Optional, Positional 6
<u>TOLERATE</u>	Tolerate unprotected	*NO, *YES	Optional, Positional 7
<u>REMOVE</u>	Remove policy	*NO, *YES	Optional, Positional 8
<u>KEYREUSE</u>	Key reuse	*DISABLED, *UNLIMITED, integer value	Optional, Positional 9

**Policy name (POLICY)**

Name of the policy, required.

The policy name must match the name of the queue which is to be protected.

The name of the new authentication information object to create.

**Message Queue Manager name (MQMNAME)**

The name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of an existing message queue manager. The maximum string length is 48 characters.

## Signature algorithm (SIGNALG)

Specifies the digital signature algorithm from one of the following values:

**\*NONE**

Messages are not signed.

**Deprecated \*MD5**

Messages are signed using the MD5 message digest algorithm.

**Deprecated \*SHA1**

Messages are signed using the SHA-1 secure hash algorithm.

**\*SHA256**

Messages are signed using the SHA-256 secure hash algorithm.

**\*SHA384**

Messages are signed using the SHA-384 secure hash algorithm.

**\*SHA512**

Messages are signed using the SHA-512 secure hash algorithm.

## Encryption algorithm (ENCALG)

Specifies the encryption algorithm to use when protecting messages from one of the following values:

**\*NONE**

Messages are not encrypted.

**Deprecated \*RC2**

Messages are encrypted using the [RC2](#) Rivest Cipher algorithm.

**Deprecated \*DES**

Messages are encrypted using the [DES](#) Data Encryption Standard algorithm.

**Deprecated \*TRIPLEDES**

Messages are encrypted using the [Triple DES](#) Data Encryption Standard algorithm.

**\*AES128**

Messages are encrypted using the AES 128-bit key Advanced Encryption Standard algorithm.

**\*AES256**

Messages are encrypted using the AES 256-bit key Advanced Encryption Standard algorithm.

## Authorized signers (SIGNER)

Specifies a list of *X500* distinguished names representing authorized message signers that are checked when browsing or destructively removing a message from a queue. If an authorized signer list is specified, only messages that are signed with a certificate identified in the list are accepted during message retrieval, even if the recipient keystore can verify the message signer.

This parameter is valid only when a signature algorithm ( [SIGNALG](#) ) has also been specified.

Note that distinguished names are case sensitive, and it is important that you enter the distinguished names exactly as they appear in the digital certificate.

The possible values are:

**\*NONE**

When handling signed messages, beyond checking the signers certificate validity, the policy does not restrict the identity of the message signer when retrieving messages.

***x500-distinguished-name***

When handling signed messages, beyond checking certificate validity, the message must have been signed by a certificate matching one of the distinguished names.

**Intended recipients (RECIP)**

Specifies a list of *X500* distinguished names representing the intended recipients that are used when putting a encrypted message to a queue. If a policy has specified an encryption algorithm (ENCALG) then at least one recipient distinguished name must be specified.

This parameter is valid only when an encryption algorithm ( [ENCALG](#) ) has also been specified.

Note that distinguished names are case sensitive, and it is important that you enter the distinguished names exactly as they appear in the digital certificate.

The possible values are:

**\*NONE**

Messages are not encrypted.

***x500-distinguished-name***

When putting messages, the message data is encrypted using the distinguished name as an intended recipient. Only the listed recipients are able to retrieve and decrypt the message.

**Tolerate unprotected (TOLERATE)**

Specifies whether messages that are not protected can still be browsed or destructively removed from a queue. This parameter can be used to gradually introduce a security policy for applications, allowing any messages that were created before the policy was introduced to be processed.

The possible values are:

**\*NO**

Messages that do not conform to the current policy are not returned to applications.

**\*YES**

Messages that have not been protected are allowed to be retrieved by applications.

**Remove policy (REMOVE)**

Specifies whether a policy is being created or removed.

The possible values are:

**\*NO**

The policy is created or altered if it already exists.

**\*YES**

The policy is removed. The only other parameters that are valid with this parameter value are policy name ( [POLICY](#) ) and queue manager name ( [MQMNAME](#) ).

**Key reuse (KEYREUSE)**

Specifies the number of times that an encryption key can be re-used, in the range 1-9,999,999, or the special values *\*DISABLED* or *\*UNLIMITED*.

Note that this is a maximum number of times a key can be reused, therefore a value of 1 means, at most, two messages can use the same key.

**\*DISABLED**

Prevents a symmetric key from being reused

## \*UNLIMITED

Allows a symmetric key to be reused any number of times.



**Attention:** Key reuse is valid only for CONFIDENTIALITY policies, that is, **SIGNALG** set to *\*NONE* and **ENCALG** set to an algorithm value. For all other policy types, you must omit the parameter, or set the **KEYREUSE** value to *\*DISABLED*.

IBM i

## SPDMQMCLQM (Suspend Cluster Queue Manager)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

Use the SPDMQMCLQM command to inform other queue managers in a cluster that the local queue manager is not available for processing and cannot be sent messages. Its action can be reversed by the RSMMQMCLQM command.

## Parameters

Table 306. Command parameters

Keyword	Description	Choices	Notes
<u>CLUSTER</u>	Cluster Name	Character value	Optional, Positional 1
<u>CLUSNL</u>	Cluster Name List	Character value	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 3
<u>MODE</u>	Mode	<b>*QUIESCE</b> , <b>*FORCE</b>	Optional, Positional 4

### Cluster Name (CLUSTER)

Specifies the name of the cluster for which the queue manager is no longer available for processing.

#### cluster-name

Specify the name of the cluster.

### Cluster Name List (CLUSNL)

Specifies the name of the namelist specifying a list of clusters for which the queue manager is no longer available for processing.

#### namelist

Specify the name of the namelist.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

#### \*DFT

Use the default queue manager.

#### queue-manager-name

Specify the name of the queue manager.

### Mode (MODE)

Specifies how the suspension of availability is to take effect:

**\*QUIESCE**

Other queue managers in the cluster are advised that the local queue manager should not be sent further messages.

**\*FORCE**

All inbound and outbound channels to other queue managers in the cluster are stopped forcibly.


**STRMQM (Start Message Queue Manager)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Start Message Queue Manager (STRMQM) command starts the local queue manager.

**Parameters**

<i>Table 307. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 1
<u>RDEFSYS</u>	Redefine system objects	*YES, *NO	Optional, Positional 2
<u>FIXDIRS</u>	Fix directories	*YES, *NO	Optional, Positional 3
<u>STRSTDTL</u>	Startup Status Detail	*ALL, *MIN	Optional, Positional 4
<u>STRSVC</u>	Service startup	*YES, *NO	Optional, Positional 5
<u>REPLAY</u>	Perform replay only	*YES, *NO	Optional, Positional 6
<u>ACTIVATE</u>	Activate backup	*YES, *NO	Optional, Positional 7
<u>STANDBY</u>	Permit Standby Queue Manager	*YES, *NO	Optional, Positional 8

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Redefine system objects (RDEFSYS)**

Specifies whether the default and system objects are redefined.

**\*NO**

Do not redefine the system objects.

**\*YES**

Starts the queue manager, redefines the default and system objects, then stops the queue manager. Any existing system and default objects belonging to the queue manager are replaced if you specify this flag.



## Fix directories (FIXDIRS)

Specifies whether missing or damaged queue manager directories are re-created.

### **\*NO**

Do not re-create any missing queue manager directories. If any damaged or missing directories are encountered during startup, the startup attempt will report an error and the STRMQM command will end immediately.

### **\*YES**

Starts the queue manager and if required re-creates any damaged or missing directories. This option should be used when performing media recovery of a queue manager.

## Startup Status Detail (STRSTSDL)

Specifies the detail of status messages that are issued while starting the queue manager.

### **\*ALL**

Display all startup status messages. This level of detail includes periodically displaying messages detailing transaction recovery and log replay. This level of detail can be useful in tracking queue manager startup progress following the abnormal termination of a queue manager.

### **\*MIN**

Displays a minimum level of status messages.

## Service startup (STRSVC)

Specifies whether the additional following QMGR components are started when the queue manager is started:

- The Channel Initiator
- The Command Server
- Listeners with CONTROL set to QMGR or STARTONLY
- Services with CONTROL set to QMGR or STARTONLY

### **\*YES**

Start the channel initiator, command server, listeners and services when the queue manager is started.

### **\*NO**

Do not start the channel initiator, command server, listeners or services when the queue manager is started.

## Perform replay only (REPLAY)

Whether the queue manager is being started to perform replay only. This enables a backup copy of a queue manager on a remote machine to replay logs created by the corresponding active machine, and to allow the backup queue manager to be activated in the event of a disaster on the active machine.

### **\*NO**

The queue manager is not being started to perform replay only.

### **\*YES**

The queue manager is being started to perform replay only. The STRMQM command will end when replay is complete.

## Activate backup (ACTIVATE)

Specifies whether to mark a queue manager as active. A queue manager that has been started with the REPLAY option is marked as a backup queue manager and cannot be started before it has been activated.

### **\*NO**

The queue manager is not to be marked as active.

**\*YES**

The queue manager is to be marked as active. Once a queue manager has been activated then it can be started as a normal queue manager using the STRMQM command without the REPLAY and ACTIVATE options.

**Permit Standby Queue Manager (STANDBY)**

Specifies whether the queue manager can start as a standby instance if an active instance of the queue manager is already running on another system. Also specifies whether this instance of the queue manager will permit standby instances of the same queue manager on other systems in preparation for failover.

**\*NO**

The queue manager is started normally.

**\*YES**

The queue manager is permitted to start as a standby instance, and it permits other standby instances of the same queue manager to be started.

**IBM i STRMQMBRK (Start MQ Pub/Sub Broker)****Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Start IBM MQ broker (STRMQMBRK) command starts a broker for a specified queue manager.

**Parameters**

<i>Table 308. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>MQMNAME</u>	Message Queue Manager name	Character value	Required, Positional 1
<u>PARENTMQM</u>	Parent Message Queue Manager	Character value	Optional, Positional 2

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Parent Message Queue Manager (PARENTMQM)**

Specifies the name of the queue manager that provides the parent broker function. Before you can add a broker to the network, channels in both directions must exist between the queue manager that hosts the new broker, and the queue manager that hosts the parent.

On restart, this parameter is optional. If present, it must be the same as it was when previously specified. If this is the root-node broker, the queue manager specified becomes its parent. You cannot specify the name of the parent broker when you use triggering to start a broker.

After a parent has been specified, it is only possible to change parentage in exceptional circumstances in conjunction with the CLRMQMBRK command. By changing a root node to become the child of an existing broker, two hierarchies can be joined. This causes subscriptions to be propagated across the two hierarchies, which now become one. After that, publications start to flow across them. To ensure predictable results, it is essential that you quiesce all publishing applications at this time.

If the changed broker detects a hierarchical error (that is, if the new parent is found also to be a descendant), it immediately shuts down. The administrator must then use CLRMQMGRK at both the changed broker and the new, false parent to restore the previous status. A hierarchical error is detected by propagating a message up the hierarchy, which can complete only when the relevant brokers and links are available.

## IBM i STRMQMCHL (Start MQ Channel)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Start MQ Channel (STRMQMCHL) command starts an MQ channel.

### Parameters

Table 309. Command parameters			
Keyword	Description	Choices	Notes
<u>CHLNAME</u>	Channel name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 2

### Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

#### channel-name

Specify the channel name.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

#### message-queue-manager-name

The name of a message queue manager.

## IBM i STRMQMCHLI (Start MQ Channel Initiator)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Start MQ Channel Initiator (STRMQMCHLI) command starts an MQ channel initiator.

## Parameters

Keyword	Description	Choices	Notes
<u>QNAME</u>	Queue name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 2

### Queue name (QNAME)

Specifies the name of the initiation queue for the channel initiation process. That is, the initiation queue that is specified in the definition of the transmission queue.

The possible values are:

#### queue-name

Specify the name of the initiation queue.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

#### message-queue-manager-name

The name of a message queue manager.

## STRMQMCSVR (Start MQ Command Server)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Start MQ Command Server (STRMQMCSVR) command starts the MQ command server for the specified queue manager.

## Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 1

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

#### queue-manager-name

Specify the name of the queue manager.

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

Use the Start IBM MQ Dead-Letter Queue Handler (STRMQMDLQ) command to perform various actions on selected messages. The command specifies a set of rules that can both select a message and perform the action on that message.

The STRMQMDLQ command takes its input from the rules table as specified by SRCFILE and SRCMBR. When the command processes, the results and a summary are written to the printer spooler file.

**Note:**

The WAIT keyword, defined in the rules table, determines whether the dead-letter queue handler ends immediately after processing messages, or waits for new messages to arrive.

**Parameters**

*Table 312. Command parameters*

Keyword	Description	Choices	Notes
<u>UDLMSGQ</u>	Undelivered message queue	<i>Character value</i> , *DFT, *NONE	Required, Positional 1
<u>SRCMBR</u>	Member containing input	<i>Name</i> , *FIRST	Required, Positional 2
<u>SRCFILE</u>	Input file	Qualified object name	Optional, Positional 3
	Qualifier 1: Input file	<i>Name</i> , <b>QXTSRC</b>	
	Qualifier 2: Library	<i>Name</i> , *LIBL, *CURLIB	
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , *DFT, *NONE	Optional, Positional 4

**Undelivered message queue (UDLMSGQ)**

Specifies the name of the local undelivered message queue that is to be processed.

The possible values are:

**\*DFT**

The local undelivered-message queue used is taken from the default queue manager for the installation. If this option is specified, the INPUTQ keyword stated in the rules table is overridden by the default undelivered-message queue for the queue manager.

**undelivered-message-queue-name**

Specify the name of the local undelivered-message queue to be used. If this option is specified, the INPUTQ keyword stated in the rules table is overridden by the stated undelivered-message queue.

**\*NONE**

The queue that is named by the INPUTQ keyword in the rules table is used, or the system-default dead-letter queue if the INPUTQ keyword in the rules table is blank.

**Member containing input (SRCMBR)**

Specifies the name of the source member, containing the user-written rules table to be processed.

The possible values are:

**\*FIRST**

The first member of the file is used.

**source-member-name**

Specify the name of the source member.

**Input file (SRCFILE)**

Specifies the name of the source file and library, in the form LIBRARY/FILE, that contains the user-written rules table to be processed.

The possible values are:

**\*LIBL**

Search the library list for the file name.

**\*CURLIB**

Use the current library.

**source-library-name**

Specify the name of the library that is being used.

The possible values are:

**QTXTSRC**

Use QTXTSRC.

**source-file-name**

Specify the name of the source file.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**\*NONE**

The queue manager that is named by the INPUTQM keyword in the rules table is used, or the system-default queue manager if the INPUTQM keyword in the rules table is blank.

 **STRMQMLSR (Start MQ Listener)****Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Start MQ Listener (STRMQMLSR) command starts an MQ TCP/IP listener.

This command is valid for TCP/IP transmission protocols only.

You can specify either a listener object or specific listener attributes.

## Parameters

Keyword	Description	Choices	Notes
<u>PORT</u>	Port number	1-65535, <b>*DFT</b>	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 2
<u>IPADDR</u>	IP Address	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 3
<u>BACKLOG</u>	Listener backlog	0-999999999, <b>*DFT</b>	Optional, Positional 4
<u>LSRNAME</u>	Listener name	<i>Character value</i> , <b>*NONE</b>	Optional, Positional 5

### Port number (PORT)

The port number to be used by the listener.

The possible values are:

#### **\*DFT**

Port number 1414 is used.

#### **port-number**

The port number to be used.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

#### **queue-manager-name**

The name of a message queue manager.

### IP Address (IPADDR)

The IP address to be used by the listener.

The possible values are:

#### **\*DFT**

The listener will listen on all IP addresses available to the TCP/IP stack.

#### **ip-addr**

The IP address to be used.

### Listener backlog (BACKLOG)

The number of concurrent connection requests the listener supports.

The possible values are:

#### **\*DFT**

255 concurrent connection requests are supported.

#### **backlog**

The number of concurrent connection requests supported.

## Listener name (LSRNAME)

The name of the MQ listener object to be started.

The possible values are:

### \*NONE

No listener object is specified.

### listener-name

Specify the name of the listener object to be started.

## STRMQMMQSC (Start MQSC Commands)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Start MQSC Commands (STRMQMMQSC) command initiates a set of IBM MQ Commands (MQSC) and writes a report to the printer spooler file.



**Attention:** Do not use the QTEMP library as the input library to STRMQMMQSC, as the usage of the QTEMP library is limited. You must use another library as an input file to the command.

Each report consists of the following elements:

- A header identifying MQSC as the source of the report.
- A numbered listing of the input MQSC commands.
- A syntax error message for any commands in error.
- A message indicating the outcome of running each correct command.
- Other messages for general errors running MQSC, as needed.
- A summary report at the end.

## Parameters

Keyword	Description	Choices	Notes
<u>SRCMBR</u>	Member containing input	<i>Name</i> , *FIRST	Required, Positional 1
<u>SRCFILE</u>	Input file	Qualified object name	Optional, Positional 2
	Qualifier 1: Input file	<i>Name</i> , <b>QMQSC</b>	
	Qualifier 2: Library	<i>Name</i> , *LIBL, *CURLIB	
<u>OPTION</u>	Option	*RUN, *VERIFY, *MVS	Optional, Positional 3
<u>WAIT</u>	Wait time	1-999999	Optional, Positional 4
<u>LCLMQMNAME</u>	Local Message Queue Manager	Character value	Optional, Positional 5
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , *DFT	Optional, Positional 6

### Member containing input (SRCMBR)

Specifies the name of the source member, containing the MQSC, to be processed.

The possible values are:



**source-member-name**

Specify the name of the source member.

**\*FIRST**

The first member of the file is used.

**Input file (SRCFILE)**

Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the MQSC to be processed.

The possible values are:

**\*LIBL**

The library list is searched for the file name.

**\*CURLIB**

The current library is used.

**source-library-name**

Specify the name of the library to be used.

The possible values are:

**QMQSC**

QMQSC is used.

**source-file-name**

Specify the name of the source file.

**Option (OPTION)**

Specifies how the MQSC commands are to be processed.

The possible values are:

**\*RUN**

If this value is specified and a value for the WAIT parameter is not specified the MQSC commands are processed directly by the local queue manager. If this value is specified and a value is also specified for the WAIT parameter the MQSC commands are processed indirectly by a remote queue manager,

**\*VERIFY**

The MQSC commands are verified and a report is written, but the commands are not run.

**\*MVS**

The MQSC commands are processed indirectly by a remote queue manager running under MVS/ESA. If you specify this option you must also specify a value for the WAIT parameter.

**Wait time (WAIT)**

Specifies the time in seconds that the STRMQMMQSC command waits for replies to indirect MQSC commands. Specifying a value for this parameter indicates that MQSC commands are executed in indirect mode by a remote queue manager. Specifying a value for this parameter is only valid when the OPTION parameter is specified as \*RUN or \*MVS.

In indirect mode, MQSC commands are queued on the command queue of a remote queue manager. Reports from the commands are then returned to the local queue manager specified in MQMNAME. Any replies received after this time are discarded, however, the MQSC command is still run.

The possible values are:

**1 - 999999**

Specify the waiting time in seconds.

## Local Message Queue Manager (LCLMQMNAME)

Specifies the name of the local queue manager through which indirect mode operation is to be performed.

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.

### **message-queue-manager-name**

Specify the name of the queue manager.

## STRMQMSVC (Start MQ Service)

### **Where allowed to run**

All environments (\*ALL)

### **Threadsafe**

Yes

The Start MQ Service (STRMQMSVC) command starts an MQ service.

## Parameters

Keyword	Description	Choices	Notes
<u>SVCNAME</u>	Service name	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2

## Service name (SVCNAME)

The name of the MQ service object to be started.

The possible values are:

### **\*NONE**

No service object is specified.

### **service-name**

Specify the name of the service definition. The maximum length of the string is 48 bytes.

## Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.

### **queue-manager-name**

The name of a message queue manager.

## STRMQMTRM (Start MQ Trigger Monitor)

### **Where allowed to run**

All environments (\*ALL)

## Threadsafe

Yes

The Start MQ Trigger Monitor (STRMQMTRM) command starts the MQ trigger monitor for the specified queue manager.

## Parameters

Keyword	Description	Choices	Notes
<u>INITQNAME</u>	Initiation queue	Character value	Required, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2

### Initiation queue **INITQNAME**

Specifies the name of the initiation queue.

#### **initiation-queue-name**

Specify the name of the initiation queue

### Message Queue Manager name (**MQMNAME**)

Specifies the name of the message queue manager.

The possible values are:

#### **\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

#### **message-queue-manager-name**

The name of a message queue manager.

## **TRCMQM (Trace MQ)**

### Where allowed to run

All environments (\*ALL)

## Threadsafe

Yes

The Trace MQ (TRCMQM) command controls tracing for all MQ jobs. TRCMQM, which sets tracing on or off, can trace message queue interface (MQI) functions, function flow, and IBM MQ for IBM i components together with any messages issued by IBM MQ.

## Parameters

Keyword	Description	Choices	Notes
<u>TRCEARLY</u>	Trace early	<b>*NO</b> , *YES	Optional, Positional 1
<u>SET</u>	Trace option setting	<b>*ON</b> , *OFF, *STS, *END	Optional, Positional 2
<u>OUTPUT</u>	Output	<b>*MQM</b> , *MQMFMT, *PEX, *ALL	Optional, Positional 3
<u>TRCLEVEL</u>	Trace level	<b>*DFT</b> , *DETAIL, *PARMS	Optional, Positional 4

Table 317. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>TRCTYPE</u>	Trace types	Single values: <b>*ALL</b> Other values (up to 14 repetitions): <b>*API, *CMTRY, *COMMS, *CSDATA, *CSFLOW, *LQMDATA, *LQMFLOW, *OTHDATA, *OTHFLOW, *RMTDATA, *RMTFLOW, *SVCDATA, *SVCFLOW, *VSNDATA</b>	Optional, Positional 5
<u>EXCLUDE</u>	Exclude types	Single values: <b>*NONE</b> Other values (up to 14 repetitions): <b>*API, *CMTRY, *COMMS, *CSDATA, *CSFLOW, *LQMDATA, *LQMFLOW, *OTHDATA, *OTHFLOW, *RMTDATA, *RMTFLOW, *SVCDATA, *SVCFLOW, *VSNDATA</b>	Optional, Positional 6
<u>INTERVAL</u>	Trace interval	1-32000000, <b>*NONE</b>	Optional, Positional 7
<u>MAXSTG</u>	Maximum storage to use	1-16, <b>*DFT</b>	Optional, Positional 8
<u>DATASIZE</u>	Trace data size	1-99999999, <b>*DFT, *ALL, *NONE</b>	Optional, Positional 9
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 10
<u>JOB</u>	Job information	Values (up to 8 repetitions): <i>Element list</i>	Optional, Positional 11
	Element 1: Job name	Qualified job name	
	Qualifier 1: Job name	Generic name, name	
	Qualifier 2: User	<i>Character value</i> , <b>X"</b>	
	Qualifier 3: Number	<i>Character value</i> , <b>X"</b>	
	Element 2: Thread identifier	<i>Character value</i> , <b>*NONE, *INITIAL</b>	
<u>STRCTL</u>	Trace start control	Values (up to 8 repetitions): <i>Character value</i> , <b>*NONE</b>	Optional, Positional 12
<u>ENDCTL</u>	Trace end control	Values (up to 8 repetitions): <i>Character value</i> , <b>*NONE</b>	Optional, Positional 13

### Trace early (TRCEARLY)

Specifies whether early tracing is selected.

Early tracing applies to all jobs for all queue managers. If a queue manager is not currently active or does not exist, then early trace will become effective during start-up or creation.

**\*NO**

Early tracing is not enabled.

**\*YES**

Early tracing is enabled.

**Trace option setting (SET)**

Specifies the collection of trace records.

The possible values are:

**\*ON**

The collection of trace records is started.

For TRCEARLY(\*NO), the collection of trace records will not be started until after the queue manager is available.

**\*OFF**

The collection of trace records is stopped. Trace records are written to files in the trace collection directory.

**\*STS**

The status of any active trace collections are written to a spool file. Any other parameters specified on the TRCMQM will be ignored.

**\*END**

The collection of trace records is stopped for all queue managers.

**Output (OUTPUT)**

Identifies the type of trace output that this command applies.

The possible values are:

**\*MQM**

This command applies to the collection of binary IBM MQ trace output in the directory specified by the TRCDIR parameter.

**\*MQMfmt**

This command applies to the collection of formatted IBM MQ trace output in the directory specified by the TRCDIR parameter.

**\*PEX**

This command applies to the collection of Performance Explorer (PEX) trace output.

**\*ALL**

This option applies to the collection of both IBM MQ unformatted trace and PEX trace output.

**Trace level (TRCLEVEL)**

Activates tracing level for flow processing trace points.

The possible values are:

**\*DFT**

Activates tracing at default level for flow processing trace points.

**\*DETAIL**

Activates tracing at high-detail level for flow processing trace points.

**\*PARMS**

Activates tracing at default-detail level for flow processing trace points.

## Trace types (TRCTYPE)

Specifies the type of trace data to store in the trace file. If this parameter is omitted, all trace points are enabled.

The possible values are:

### **\*ALL**

All the trace data as specified by the following keywords is stored in the trace file.

### **trace-type-list**

You can specify more than one option from the following keywords, but each option can occur only once.

### **\*API**

Output data for trace points associated with the MQI and major queue manager components.

### **\*CMTRY**

Output data for trace points associated with comments in the MQ components.

### **\*COMMS**

Output data for trace points associated with data flowing over communications networks.

### **\*CSDATA**

Output data for trace points associated with internal data buffers in common services.

### **\*CSFLOW**

Output data for trace points associated with processing flow in common services.

### **\*LQMDATA**

Output data for trace points associated with internal data buffers in the local queue manager.

### **\*LQMFLOW**

Output data for trace points associated with processing flow in the local queue manager.

### **\*OTHDATA**

Output data for trace points associated with internal data buffers in other components.

### **\*OTHFLOW**

Output data for trace points associated with processing flow in other components.

### **\*RMTDATA**

Output data for trace points associated with internal data buffers in the communications component.

### **\*RMTFLOW**

Output data for trace points associated with processing flow in the communications component.

### **\*SVCDATA**

Output data for trace points associated with internal data buffers in the service component.

### **\*SVCFLOW**

Output data for trace points associated with processing flow in the service component.

### **\*VSNDATA**

Output data for trace points associated with the version of IBM MQ running.

## Exclude types (EXCLUDE)

Specifies the type of trace data to omit from the trace file. If this parameter is omitted, all trace points specified in TRCTYPE are enabled.

The possible values are:

### **\*ALL**

All the trace data as specified by the following keywords is stored in the trace file.

### **trace-type-list**

You can specify more than one option from the following keywords, but each option can occur only once.

**\*API**

Output data for trace points associated with the MQI and major queue manager components.

**\*CMTRY**

Output data for trace points associated with comments in the MQ components.

**\*COMMS**

Output data for trace points associated with data flowing over communications networks.

**\*CSDATA**

Output data for trace points associated with internal data buffers in common services.

**\*CSFLOW**

Output data for trace points associated with processing flow in common services.

**\*LQMDATA**

Output data for trace points associated with internal data buffers in the local queue manager.

**\*LQMFLOW**

Output data for trace points associated with processing flow in the local queue manager.

**\*OTHDATA**

Output data for trace points associated with internal data buffers in other components.

**\*OTHFLOW**

Output data for trace points associated with processing flow in other components.

**\*RMTDATA**

Output data for trace points associated with internal data buffers in the communications component.

**\*RMTFLOW**

Output data for trace points associated with processing flow in the communications component.

**\*SVCDATA**

Output data for trace points associated with internal data buffers in the service component.

**\*SVCFLOW**

Output data for trace points associated with processing flow in the service component.

**\*VSNDATA**

Output data for trace points associated with the version of IBM MQ running.

**Trace interval (INTERVAL)**

Specifies an interval in seconds that trace should be collected for. If this parameter is omitted then trace will continue to be collected until it is stopped manually via the TRCMQM commands or an FDC with a probe identifier specified in ENDCTL is encountered.

The possible values are:

**collection-interval**

Specify a value in seconds ranging from 1 through 32000000.

You cannot specify a value for both INTERVAL and ENDCTL.

**Maximum storage to use (MAXSTG)**

Specifies the maximum size of storage to be used for the collected trace records.

The possible values are:

**\*DFT**

The default maximum is 1 megabyte (1024 kilobytes).

**maximum-megabytes**

Specify a value ranging from 1 through 16.

## Trace data size (DATASIZE)

Specifies the number of bytes of user data included in the trace.

The possible values are:

### **\*DFT**

The default trace value is used.

### **\*ALL**

All the user data is traced.

### **\*NONE**

This option will turn off the trace for sensitive user data.

### **data-size-in-bytes**

Specify a value in ranging from 1 through 99999999.

## Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

This parameter is only valid when TRCEARLY is set to \*NO.

When TRCEARLY is set to \*YES all queue managers are traced.

The possible values are:

### **\*DFT**

Trace the default queue manager.

### **queue-manager-name**

Specify the name of the queue manager to trace.

## Job information (JOB)

Specifies which jobs are to be traced.

The value of this parameter can be one of the following:

### **generic-jobname**

A generic 10 character jobname. All jobs that match the jobname will be enabled to collect trace. For example 'AMQ\*' will collect trace for all jobs with a prefix of AMQ.

### **Job-name/User/Number**

A fully qualified jobname. Only the job specified by the qualified jobname will be traced.

### **Job-name/User/Number/thread-identifier**

A fully qualified jobname and associated thread identifier. Only the thread in the job specified by the qualified jobname will be traced. Note that the thread identifier is the internal identifier allocated by IBM MQ, it is not related to the IBM i thread identifier.

## Trace start control (STRCTL)

Specifies that trace is started when an FDC with one of the specified probe identifiers is generated.

### **AANNNNNN**

A probe identifier is an 8 character string of the format (AANNNNNN) where A represents alphabetic characters and N represents numeric digits.

Up to 8 probe identifiers may be specified.

## Trace end control (ENDCTL)

Specifies that trace is ended when an FDC with one of the specified probe identifiers is generated.



## AANNNNNN

A probe identifier is an 8 character string of the format (AANNNNNN) where A represents alphabetic characters and N represents numeric digits.

Up to 8 probe identifiers may be specified.

You cannot specify a value for both ENDCTL and INTERVAL.

### Related tasks

[Using trace on IBM i](#)

## WRKMQM (Work with MQ Queue Manager)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Work with Queue Managers (WRKMQM) command allows you to work with one or more queue manager definitions, and allows you to perform the following operations:

- Change a queue manager
- Create a queue manager
- Delete a queue manager
- Start a queue manager
- Display a queue manager
- End a queue manager
- Work with channels of a queue manager
- Work with namelists of a queue manager
- Work with queues of a queue manager
- Work with processes of a queue manager

### Parameters

Table 318. Command parameters			
Keyword	Description	Choices	Notes
<a href="#">MQMNAME</a>	Message Queue Manager name	Character value, *ALL	Optional, Positional 1

### Message Queue Manager name (MQMNAME)

Specifies the name or names of the message queue managers to select.

The possible values are:

#### \*ALL

All queue managers are selected.

#### generic-queue-manager-name

Specify the generic name of the queue managers to select. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all queue managers having names that start with the character string. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

**Note:** You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered. You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

## IBM i WRKMQMAUT (Work with MQ Authority)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Work with MQ Authority (WRKMQMAUT) displays a list of all the authority profile names and their types, which match the specified parameters. This enables you to delete, work with and create the authority records for an MQM authority profile record.

### Parameters

Keyword	Description	Choices	Notes
<u>OBJ</u>	Object/Profile name	<i>Character value</i> , *ALL	Optional, Positional 1
<u>OBJTYPE</u>	Object type	*Q, *PRC, *MQM, *NMLIST, *AUTHINFO, *LSR, *SVC, *CHL, *CLTCN, *ALL, *TOPIC, *RMTMQMNAME	Optional, Positional 2
<u>OUTPUT</u>	Output	*, *PRINT	Optional, Positional 3
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , *DFT	Optional, Positional 4
<u>SRVCOMP</u>	Service Component name	<i>Character value</i> , *DFT	Optional, Positional 5

### Object name (OBJ)

Specify the object name or authority profile name of the object to select.

The possible values are:

#### \*ALL

All authority records matching the specified object type are listed. \*ALL cannot represent a generic profile.

#### object-name

Specify the name of an MQ object; all authority records for which the object name or generic profile name match this object name are selected.

#### generic profile

Specify the generic profile of an MQ object; only the authority record which exactly matches the generic profile is selected. A generic profile is a character string containing one or more generic characters anywhere in the string. The generic characters are (?), (\*) and (\*\*).

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

### Object type (OBJTYPE)

Specifies the object type of the authority profile to select.

**\*ALL**

All MQ object types.

**\*Q**

All queue object types.

**\*AUTHINFO**

Authentication Information object.

**\*MQM**

Message Queue Manager.

**\*NMLIST**

Namelist object.

**\*PRC**

Process definition.

**\*CHL**

Channel object.

**\*CLTCN**

Client Connection Channel object.

**\*LSR**

Listener object.

**\*SVC**

Service object.

**\*TOPIC**

Topic object.

**\*RMTMQMNAME**

Remote queue manager name.

**Output (OUTPUT)**

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

A detailed list of the users and their authorities registered with the selected authority profile record is printed with the job's spooled output.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Service Component name (SRVCOMP)**

Specify the name of the installed authorization service in which to search for the authorities to display.

The possible values are:

**\*DFT**

All installed authorization components are searched for the specified authority profile name and object type.

**Authorization-service-component-name**

The component name of the authorization service as specified in the Queue manager's qm.ini file.


**WRKMQMAUTD (Work with MQ Authority Data)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Authority Records (WRKMQMAUTD) displays a list of all the users registered to a particular authority profile name and type. This enables you to grant, revoke, delete and create authority records.

**Parameters**

<i>Table 320. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>OBJ</u>	Object/Profile name	Character value	Required, Positional 1
<u>OBJTYPE</u>	Object type	*Q, *PRC, *MQM, *NMLIST, *AUTHINFO, *CHL, *CLTCN, *SVC, *LSR, *TOPIC	Required, Positional 2
<u>USER</u>	User name	<i>Name</i> , *PUBLIC, <b>*ALL</b>	Optional, Positional 3
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 4
<u>SRVCOMP</u>	Service Component name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 5

**Object name (OBJ)**

Specify the object name or authority profile name of the object to select.

**object-name**

Specify the name of an MQ object; all authority records for which the object name or generic profile name match this object name are selected.

**generic profile**

Specify the generic profile of an MQ object; only the authority record which exactly matches the generic profile is selected. A generic profile is a character string containing one or more generic characters anywhere in the string. The generic characters are (?), (\*) and (\*\*).

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

**Object type (OBJTYPE)**

Specifies the object type of the authority profile to select.

**\*Q**

All queue object types.

**\*AUTHINFO**

Authentication Information object.

**\*MQM**

Message Queue Manager.

**\*NMLIST**

Namelist object.

**\*PRC**

Process definition.

**\*CHL**

Channel object.

**\*CLTCN**

Client Connection Channel object.

**\*LSR**

Listener object.

**\*SVC**

Service object.

**\*TOPIC**

Topic object.

**User name (USER)**

Specifies the name of the user for whom authorities for the named object are displayed.

The possible values are:

**\*ALL**

List all relevant users.

**\*PUBLIC**

The user name implying all users of the system.

**user-profile-name**

Specify the name of the user.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Service Component name (SRVCOMP)**

Specify the name of the installed authorization service in which to search for the authorities to display.

The possible values are:

**\*DFT**

All installed authorization components are searched for the specified authority profile name and object type.

**Authorization-service-component-name**

The component name of the authorization service as specified in the Queue manager's qm.ini file.

**IBM i WRKMQMAUTI (Work with AuthInfo objects)****Where allowed to run**

All environments (\*ALL)

## Threadsafe

Yes

The Work with MQ AuthInfo objects (WRKMQMAUTI) command allows you to work with multiple authentication information objects which are defined on the local queue manager.

This enables you to change, copy, create, delete, display, and display and change authority to an MQ authentication information object.

## Parameters

Keyword	Description	Choices	Notes
<u>AINAME</u>	AuthInfo name	Character value, <b>*ALL</b>	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATA, *ALTTIME, *AUTHTYPE, *CONNAME, *TEXT, *USERNAME, *OCSPURL	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### AuthInfo name (AINAME)

The name or names of the authentication information objects.

The possible values are:

#### **\*ALL or \***

All authentication information objects are selected.

#### **generic-authinfo-name**

The generic name of the authentication information objects. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all authentication information objects having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

#### **authentication-information-name**

Specify the name of a single authentication information object.

### Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of an existing message queue manager. The maximum string length is 48 characters.

**Filter command (WHERE)**

This parameter can be used to selectively display those AuthInfo objects with particular AuthInfo attributes only.

The parameter takes three arguments, a keyword, an operator, and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ALTDATE**

The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

**\*ALTTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*AUTHTYPE**

The type of the authentication information object.

The filter value is one of the following:

**\*CRLLDAP**

The type of the authentication information object is CRLLDAP.

**\*OCSP**

The type of the authentication information object is OCSP.

**\*IDPWOS**

Connection authentication user ID and password checking is done using the operating system.

**\*IDPWLDAP**

Connection authentication user ID and password checking is done using an LDAP server.

**\*CONNAME**

The address of the host on which the LDAP server is running.

The filter value is the address name.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the queue.

**\*USERNAME**


The distinguished name of the user.

The filter value is the distinguished name.

**\*OCSPURL**

The OCSP Responder URL.

The filter value is the URL name.

 **WRKMQMCHL (Work with MQ Channels)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with IBM MQ Channels (WRKMQMCHL) command allows you to work with one or more channel definitions. This enables you to create, start, end, change, copy, delete, ping, display and reset channels, and resolve in-doubt units of work.



## Parameters

<i>Table 322. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>CHLNAME</u>	Channel name	<i>Character value, *ALL</i>	Optional, Positional 1
<u>CHLTYPE</u>	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN, *ALL	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 3
<u>STATUS</u>	Channel status	*ALL, *INACTIVE, *STOPPED, *BINDING, *RETRYING, *RUNNING, *SWITCHING	Optional, Positional 4

Table 322. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 5
	Element 1: Filter keyword	*AFFINITY, *ALTDATA, *ALTTIME, *BATCHHB, *BATCHINT, *BATCHLIM, *BATCHSIZE, *CLNTWGHT, *CLUSNL, *CLUSTER, *CLWLPRTY, *CLWLANK, *CLWLWGHT, *COMPHDR, *COMPMSG, *CONNAME, *CVTMSG, *DSCITV, *HRTBTINTVL, *KAINT, *LOCLADDR, *LONGRTY, *LONGTMR, *MAXINST, *MAXINSTC, *MAXMSGLEN, *MCANAME, *MCATYPE, *MCAUSRID, *MODENAME, *MONCHL, *MSGEXIT, *MSGRTYDATA, *MSGRTYEXIT, *MSGRTYITV, *MSGRTYNBR, *MSGUSRDATA, *NETPRTY, *NPMSPEED, *PROPCTL, *PUTAUT, *RCVEXIT, *RCVUSRDATA, *SCYEXIT, *SCYUSRDATA, *SEQNUMWRAP, *SHARECNV, *SHORTRTY, *SHORTTMR, *SNDEXIT, *SNDUSRDATA, *SSLCAUTH, *SSLCIPH, *SSLPEER, *STATCHL, *TEXT, *TGTMQNAME, *TMQNAME, *TPNAME, *TRPTYPE, *USERID	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Channel name (CHLNAME)

Specifies the name or names of the IBM MQ channel definitions to be selected.

The possible values are:

**\*ALL**

All channel definitions are selected.

**generic-channel-name**

Specify the generic name of the channel definitions to be selected. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all channel definitions having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**channel-name**

Specify the name of the channel definition.

**Channel type (CHLTYPE)**

Specifies the type of channel definitions that are to be displayed.

The possible values are:

**\*ALL**

All the channel types are selected.

**\*SDR**

Sender channel

**\*SVR**

Server channel

**\*RCVR**

Receiver channel

**\*RQSTR**

Requester channel

**\*SVRCN**

Server-connection channel

**\*CLUSSDR**

Cluster-sender channel

**\*CLUSRCVR**

Cluster-receiver channel

**\*CLTCN**

Client-connection channel

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

**message-queue-manager-name**

The name of a message queue manager.

**Channel status (STATUS)**

Specifies the status type of the IBM MQ channel definitions to be selected.

The possible values are:

**\*ALL**

Channels with any status are selected.

**\*BINDING**

Only channels with a binding status are selected.

**\*INACTIVE**

Only channels with an inactive status are selected.

**\*RETRYING**

Only channels with a retrying status are selected.

**\*RUNNING**

Only channels with a running status are selected.

**\*STOPPED**

Only channels with a stopped status are selected.

**\*SWITCHING**

Only channels with a switching status are selected.

**Filter command (WHERE)**

This parameter can be used to selectively display those channels with particular channel attributes only.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*AFFINITY**

Connection Affinity.

The filter value is one of the following:

**\*PREFERRED**

Preferred connection affinity.

**\*NONE**

No connection affinity.

**\*ALTDATE**

The date on which the definition or information was last altered.

The filter value is the data in the form yyyy-mm-dd.

**\*ALTTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*BATCHHB**

Batch heartbeat interval in milliseconds.

The filter value is the integer interval time.

**\*BATCHINT**

Batch interval in milliseconds.

The filter value is the integer interval time.

**\*BATCHLIM**

Batch data limit in kilobytes.

The limit of the amount of data that can be sent through a channel.

**\*BATCHSIZE**

Batch size.

The filter value is the integer batch size.

**\*CLNTWGHT**

Client channel weight.

The filter value is the integer client channel weight.

**\*CLUSNL**

Cluster namelist.

The filter value is the list of cluster names.

**\*CLUSTER**

The cluster to which the channel belongs.

The filter value is the name of the cluster.

**\*CLWL RANK**

Cluster workload rank.

The filter value is the integer rank.

**\*CLWL PRTY**

Cluster workload priority.

The filter value is the integer priority.

**\*CLWLWGHT**

Cluster workload weight.

The filter value is the integer weight.

**\*COMP HDR**

Header compression.

The filter value is one of the following:

**\*NONE**

No header data compression is performed.

**\*SYSTEM**

Header data compression is performed.

**\*COMP MSG**

Message compression.

The filter value is one of the following:

**\*NONE**

No message data compression is performed.

**\*RLE**

Message data compression is performed using RLE.

**\*ZLIB HIGH**

Message data compression is performed using ZLIB compression. A high level of compression is preferred.

**\*ZLIB FAST**

Message data compression is performed using ZLIB compression. A fast compression time is preferred.

**\*ANY**

Any compression technique supported by the queue manager can be used.

**\*CON NAME**

Remote connection name.

The filter value is the connection name string.

**\*CVT MSG**

Whether the message is converted before transmission.

The filter value is one of the following:

**\*YES**

The application data in the message is converted before sending.

**\*NO**

The application data in the message is not converted before sending.

**\*DSC ITV**

Disconnect interval in seconds.

The filter value is the integer interval time.

**\*HRTBTINTVL**

Heartbeat interval in seconds.

The filter value is the integer interval time.

**\*KAINT**

Keep alive interval in seconds.

The filter value is the integer interval time.

**\*LOCLADDR**

Local connection name.

The filter value is the connection name string.

**\*LONGRTY**

Long retry count.

The filter value is the integer count.

**\*LONGTMR**

Long retry interval in seconds.

The filter value is the integer interval time.

**\*MAXINST**

Maximum instances of an individual server-connection channel.

The filter value is the integer number of instances.

**\*MAXINSTC**

Maximum instances of an individual server-connection channel from a single client.

The filter value is the integer number of instances.

**\*MAXMSGLEN**

Maximum message length.

The filter value is the integer length.

**\*MCANAME**

Message channel agent name.

The filter value is the agent name.

**\*MCATYPE**

Whether the message channel agent program should run as a thread or process.

The filter value is one of the following:

**\*PROCESS**

The message channel agent runs as a separate process.

**\*THREAD**

The message channel agent runs as a separate thread.

**\*MCAUSRID**

Message channel agent user identifier.

The filter value is the user identifier string.

**\*MODENAME**

SNA mode name.

The filter value is the mode name string.

**\*MONCHL**

Channel Monitoring.

The filter value is one of the following:

**\*QMGR**

The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONCHL.

**\*OFF**

Online Monitoring Data collection for this channel is disabled.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

**\*MSGEXIT**

Message exit name.

The filter value is the exit name.

**\*MSGRTYDATA**

Message retry exit user data.

The filter value is the user data string.

**\*MSGRTYEXIT**

Message retry exit name.

The filter value is the exit name.

**\*MSGRTYITV**

Message retry interval interval in seconds.

The filter value is the integer interval time.

**\*MSGRTYNBR**

Number of message retries.

The filter value is the integer number of retries.

**\*MSGUSRDATA**

Message exit user data.

The filter value is the user data string.

**\*NETPRTY**

Network connection priority ranging from 0 through 9.

The filter value is the integer priority value.

**\*NPMSPEED**

Whether the channel supports fast nonpersistent messages.

The filter value is one of the following:

**\*FAST**

The channel supports fast nonpersistent messages.

**\*NORMAL**

The channel does not support fast nonpersistent messages.

**\*PROPCTL**

Message Property Control.

The filter value is one of the following:

**\*COMPAT**

Compatibility mode

**\*NONE**

No properties sent to remote queue manager.



**\*ALL**

All properties sent to remote queue manager.

**\*PUTAUT**

Whether the user identifier in the context information is used.

The filter value is one of the following:

**\*DFT**

No authority check is made before the message is put on the destination queue.

**\*CTX**

The user identifier in the message context information is used to establish authority to put the message.

**\*RCVEXIT**

Receive exit name.

The filter value is the exit name.

**\*RCVUSRDATA**

Receive exit user data.

The filter value is the user data string.

**\*SCYEXIT**

Security exit name.

The filter value is the exit name.

**\*SCYUSRDATA**

Security exit user data.

The filter value is the user data string.

**\*SEQNUMWRAP**

Maximum message sequence number.

The filter value is the integer sequence number.

**\*SHARECNV**

The number of shared conversations over a TCP/IP socket.

The filter value is the integer number of shared conversations.

**\*SHORTRTY**

Short retry count.

The filter value is the integer count.

**\*SHORTTMR**

Short retry interval in seconds.

The filter value is the integer interval time.

**\*SNDEXIT**

Send exit name.

The filter value is the exit name.

**\*SNDUSRDATA**

Send exit user data.

The filter value is the user data string.

**\*SSLCAUTH**

Whether the channel should carry out client authentication over TLS.

The filter value is one of the following:

**\*REQUIRED**

Client authentication is required.

**\*OPTIONAL**

Client authentication is optional.

**\*SSLCIPH**

The CipherSpec using in TLS channel negotiation.

The filter value is the name of the CipherSpec.

**\*SSLPEER**

The X500 peer name used in TLS channel negotiation.

The filter value is the peer name.

**\*STATCHL**

Channel Statistics.

The filter value is one of the following:

**\*QMGR**

The collection of statistics data is inherited from the setting of the queue manager attribute STATCHL.

**\*OFF**

Statistics data collection for this channel is disabled.

**\*LOW**

Statistics data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Statistics data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Statistics data collection is turned on with a high ratio of data collection.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the channel.

**\*TGTMQMNAME**

Target queue manager name.

The filter value is the target queue manager of the channel.

**\*TMQNAME**

Transmission queue name.

The filter value is the name of the queue.

**\*TPNAME**

The SNA transaction program name.

The filter value is the program name string.

**\*TRPTYPE**

Transport type.

The filter value is one of the following:

**\*TCP**

Transmission Control Protocol / Internet Protocol (TCP/IP).

**\*LU62**

SNA LU 6.2.

**\*USERID**

Task user identifier.

The filter value is the user identifier string.

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Channel Status (WRKMQMCHST) command allows you to work with the status of one or more channel definitions.

**Parameters**

<i>Table 323. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>CHLNAME</u>	Channel name	<i>Character value</i> , <b>*ALL</b>	Optional, Positional 1
<u>CONNNAME</u>	Connection name	<i>Character value</i> , <b>*ALL</b>	Optional, Positional 2
<u>TMQNAME</u>	Transmission queue name	<i>Character value</i> , <b>*ALL</b>	Optional, Positional 3
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 4
<u>CHLSTS</u>	Channel status	<b>*ALL</b> , *SAVED, *CURRENT	Optional, Positional 5
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 6
	Element 1: Filter keyword	*CHLSTS, *CHLTYPE, *COMPHDR, *COMPMSG, *CONNNAME, *INDOUBT, *INDMSGGS, *INDSEQNO, *LSTSEQNO, *MONCHL, *RMTMQMNAME, *RMTVERSION, *SHARECNV, *STATUS, *SUBSTATE, *TMQNAME, *XQMSGSA, *LSTMSGDATE, *LSTMSGTIME, *MSGGS	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

**Channel name (CHLNAME)**

Specifies the name of the channel definition.

The possible values are:

**\*ALL**

All channel definitions are selected.

**generic-channel-name**

Specify the generic name of the channel definitions to be selected. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all channel definitions having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**channel-name**

Specify the name of the channel definition.

**Connection name (CONNNAME)**

Specifies the name of the machine to connect.

The possible values are:

**\*ALL**

All the channels are selected.

**generic-connection-name**

Specify the generic connection name of the required channels.

**connection-name**

Specify the connection name of the required channels.

**Transmission queue name (TMQNAME)**

Specifies the name of the transmission queue.

The possible values are:

**\*ALL**

All the transmission queues are selected.

**generic-transmission-queue-name**

Specify the generic name of the transmission queues.

**transmission-queue-name**

Specify the name of the transmission queue. A transmission queue name is required if the channel definition type (CHLTYPE) is \*SDR or \*SVR.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

**message-queue-manager-name**

The name of a message queue manager.

**Channel status (CHLSTS)**

Specifies the type of channel status to display.

The possible values are:

**\*SAVED**

Saved channel status only is displayed. Status is not saved until a persistent message is transmitted across a channel, or a nonpersistent message is transmitted with a NPMSPEED of NORMAL. Because status is saved at the end of each batch, a channel has no saved status until at least one batch has been transmitted.

**\*CURRENT**

Current channel status only is displayed. This applies to channels that have been started, or on which a client has connected, and that have not finished or disconnected normally. The current status data is updated as messages are sent or received.

**\*ALL**

Both saved and current channel status is displayed.

**Filter command (WHERE)**

This parameter can be used to selectively display the status of only those channels with particular channel status attributes.

The parameter takes three arguments, a keyword, an operator, and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*CHLSTS**

The type of channel status.

The filter value is one of the following:

**\*CURRENT**

Current status for an active channel.

**\*SAVED**

Saved status for an active or inactive channel.

**\*CHLTYPE**

The type of channel.

The filter value is one of the following:

**\*SDR**

Sender channel.

**\*SVR**

Server channel.

**\*RCVR**

Receiver channel.

**\*RQSTR**

Requester channel.

**\*CLUSSDR**

Cluster-sender channel.

**\*CLUSRCVR**

Cluster-receiver channel.

**\*SVRCN**

Server-connection channel.

**\*COMPHDR**

Whether the channel performs header data compression.

The filter value is one of the following:

**\*NONE**

No header data compression is performed.

**\*SYSTEM**

Header data compression is performed.

**\*COMPMSG**

Whether the channel performs message data compression.

The filter value is one of the following:

**\*NONE**

No message data compression is performed.

**\*RLE**

Message data compression is performed using RLE.

**\*ZLIBHIGH**

Message data compression is performed using ZLIB compression. A high level of compression is preferred.

**\*ZLIBFAST**

Message data compression is performed using ZLIB compression. A fast compression time is preferred.

**\*CONNAME**

The connection name of the channel.

The filter value is the connection name string.

**\*INDOUBT**

Whether there are any in-doubt messages in the network.

The filter value is either \*NO or \*YES.

**\*INDMSGS**

The number of in-doubt messages.

The filter value is the integer number of messages.

**\*INDSEQNO**

The sequence number of the message that is in-doubt.

The filter value is the integer sequence number.

**\*LSTMSGTIME**

The time the last message was sent on the channel.

The filter value is the time in the form hh:mm:ss.

**\*LSTMSGDATE**

The date that the last message was sent on the channel.

The filter value is the data in the form yyyy-mm-dd

**\*LSTSEQNO**

The last message sequence number.

The filter value is the integer sequence number.

**\*MONCHL**

The current level of monitoring data collection for the channel.

The filter value is one of the following:

**\*NONE**

No monitoring data is collected.

**\*LOW**

A low ratio of monitoring data is collected.

**\*MEDIUM**

A medium ratio of monitoring data is collected.

**\*HIGH**

A high ratio of monitoring data is collected.

**\*MSGS**

The number of messages that have been sent on the channel.

The filter value is the integer number of messages.

**\*RMTMQMNAME**

The remote message queue manager.

The filter value is the message queue manager name.

**\*RMTVERSION**

The remote partner version.

The filter value is the integer format of the remote partner version.

**\*SHARECNV**

The number of shared conversations over a TCP/IP socket.

The filter value is the integer number of shared conversations.

**\*STATUS**

The status of the channel.

The filter value is one of the following:

**\*BINDING**

The channel is establishing a session.

**\*INACTIVE**

The channel has ended processing normally or the channel has never started.

**\*INITIALIZING**

The channel initiator is attempting to start the channel.

**\*PAUSED**

The channel is waiting for the message retry interval.

**\*REQUESTING**

The channel has been requested to start.

**\*RETRYING**

A previous attempt to establish a connection has failed. The channel will retry the connection after the specified interval.

**\*RUNNING**

The channel is transferring or is ready to transfer data.

**\*STARTING**

The channel is ready to begin negotiation with the target MCA.

**\*STOPPED**

The channel has been stopped.

**\*STOPPING**

The channel has been requested to stop.

**\*SWITCHING**

The channel is switching transmission queues.

**\*SUBSTATE**

The channel substate.

The filter value is one of the following:

**\*ENDBATCH**

End of batch processing.

**\*SEND**

Sending data.

**\*RECEIVE**

Receiving data.

**\*SERIALIZE**

Serializing with the partner channel.

**\*RESYNCH**

Resynchronizing with the partner channel.

**\*HEARTBEAT**

Heartbeat processing.



**\*SCYEXIT**

Processing a security exit.

**\*RCVEXIT**

Processing a receive exit.

**\*SENDEXIT**

Processing a send exit.

**\*MSGEXIT**

Processing a message exit.

**\*MREXIT**

Processing a message-retry exit.

**\*CHADEXIT**

Processing a channel auto-definition exit.

**\*NETCONNECT**

Connecting to remote machine.

**\*SSLHANDSHK**

Establishing a TLS connection.

**\*NAMESERVER**

Requesting information from a name server.

**\*MQPUT**

MQPUT processing.

**\*MQGET**

MQGET processing.

**\*MQICALL**

Processing an MQI call.

**\*COMPRESS**

Compressing or extracting data.

**\*TMQNAME**

The transmission queue of the channel.

The filter value is the queue name.

**\*XQMSGSA**

The number of messages queued on the transmission queue available for MQGET. This field is valid for cluster-sender channels.

The filter value is the integer number of messages.

## **WRKMQMCL (Work with MQ Clusters)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Clusters command, **WRKMQMCL**, allows you to work with multiple cluster queue manager definitions that are defined on the local queue manager.

**Parameters**

<i>Table 324. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>CLUSQMGR</u>	Cluster Queue Manager name	<i>Character value</i> , <b>*ALL</b>	Optional, Positional 1

Table 324. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *BATCHHB, *BATCHINT, *BATCHLIM, *BATCHSIZE, *CHLNAME, *CLUSDATE, *CLUSQMGR, *CLUSTER, *CLUSTIME, *CLWLPRTY, *CLWLRRANK, *CLWLWGHT, *COMPHDR, *COMPMSG, *CONNAME, *CVTMSG, *DFNTYPE, *DSCITV, *HRTBTINTVL, *KAINT, *LOCLADDR, *LONGRTY, *LONGTMR, *MAXMSGLEN, *MCANAME, *MCATYPE, *MCAUSRID, *MONCHL, *MSGEXIT, *MSGRTYDATA, *MSGRTYEXIT, *MSGRTYITV, *MSGRTYNBR, *MSGUSRDATA, *NETPRTY, *NPMSPEED, *PUTAUT, *QMID, *QMTYPE, *RCVEXIT, *RCVUSRDATA, *SCYEXIT, *SCYUSRDATA, *SEQNUMWRAP, *SHORTRTY, *SHORTTMR, *SNDEXIT, *SNDUSRDATA, *SSLCAUTH, *SSLCIPH, *SSLPEER, *STATCHL, *STATUS, *SUSPEND, *TEXT, *TRPTYPE, *USERID, *XMITQ	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Cluster Queue Manager name (CLUSQMGR)

Specifies the name or names of the cluster queue manager definitions.

**\*ALL**

All cluster queue manager definitions are selected.

**generic-cluster-queue-manager-name**

Specify the generic name of the MQ cluster queue manager definitions. A generic name is a character string followed by an asterisk (\*)> For example ABC\*, it selects all cluster queue manager definitions having names that start with the character string. You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered. You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**cluster-queue-manager-name**

Specify the name of the MQ cluster queue manager definition.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Filter command (WHERE)**

This parameter can be used to selectively display only those cluster queue managers with particular attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ALTDATE**

The date on which the definition or information was last altered.

The filter value is the data in the form yyyy-mm-dd.

**\*ALTTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*BATCHHB**

Batch heartbeat interval in milliseconds.

The filter value is the integer interval time.

**\*BATCHINT**

Batch interval in milliseconds.

The filter value is the integer interval time.

**\*BATCLIM**

Batch data limit in kilobytes.

The limit of the amount of data that can be sent through a channel.

**\*BATCSIZE**

Batch size.

The filter value is the integer batch size.

**\*CHANNEL**

The channel name of the cluster queue manager.

The filter value is the name of the channel.

**\*CLUSDATE**

The date on which the definition became available to the local queue manager.

The filter value is the data in the form yyyy-mm-dd.

**\*CLUSQMGR**

The cluster queue manager name.

The filter value is the name of the cluster queue manager.

**\*CLUSTER**

The cluster to which the cluster queue manager belongs.

The filter value is the name of the cluster.

**\*CLUSTIME**

The time at which the definition became available to the local queue manager.

The filter value is the time in the form hh:mm:ss.

**\*CLWLRANK**

Cluster workload rank.

The filter value is the integer rank.

**\*CLWLPRTY**

Cluster workload priority.

The filter value is the integer priority.

**\*CLWLWGHT**

Cluster workload weight.

The filter value is the integer weight.

**\*COMPHDR**

Header compression.

The filter value is one of the following:

**\*NONE**

No header data compression is performed.

**\*SYSTEM**

Header data compression is performed.

**\*COMPMSG**

Message compression.

The filter value is one of the following:

**\*NONE**

No message data compression is performed.

**\*RLE**

Message data compression is performed using RLE.

**\*ZLIBHIGH**

Message data compression is performed using ZLIB compression. A high level of compression is preferred.

**\*ZLIBFAST**

Message data compression is performed using ZLIB compression. A fast compression time is preferred.

**\*ANY**

Any compression technique supported by the queue manager can be used.

**\*CONNAME**

Remote connection name.

The filter value is the connection name string.

**\*CVTMSG**

Whether the message should be converted before transmission.

The filter value is one of the following:

**\*YES**

The application data in the message is converted before sending.

**\*NO**

The application data in the message is not converted before sending.

**\*DFNTYPE**

How the cluster channel was defined.

The filter value is one of the following:

**\*CLUSSDR**

As a cluster-sender channel from an explicit definition.

**\*CLUSSDRA**

As a cluster-sender channel by auto-definition alone.

**\*CLUSSDRB**

As a cluster-sender channel by auto-definition and an explicit definition.

**\*CLUSRCVR**

As a cluster-receiver channel from an explicit definition.

**\*DSCITV**

Disconnect interval in seconds.

The filter value is the integer interval time.

**\*HRTBTINTVL**

Heartbeat interval in seconds.

The filter value is the integer interval time.

**\*KAINT**

Keep alive interval in seconds.

The filter value is the integer interval time.

**\*LOCLADDR**

Local connection name.

The filter value is the connection name string.

**\*LONGRTY**

Long retry count.

The filter value is the integer count.

**\*LONGTMR**

Long retry interval in seconds.

The filter value is the integer interval time.

**\*MAXMSGLEN**

Maximum message length.

The filter value is the integer length.

**\*MCANAME**

Message channel agent name.

The filter value is the agent name.

**\*MCATYPE**

Whether the message channel agent program should run as a thread or process.

The filter value is one of the following:

**\*PROCESS**

The message channel agent runs as a separate process.

**\*THREAD**

The message channel agent runs as a separate thread.

**\*MCAUSRID**

Message channel agent user identifier.

The filter value is the user identifier string.

**\*MONCHL**

Channel Monitoring.

The filter value is one of the following:

**\*QMGR**

The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONCHL.

**\*OFF**

Online Monitoring Data collection for this channel is disabled.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

**\*MSGEXIT**

Message exit name.

The filter value is the exit name.

**\*MSGRTYDATA**

Message retry exit user data.

The filter value is the user data string.

**\*MSGRTYEXIT**

Message retry exit name.

The filter value is the exit name.

**\*MSGRTYITV**

Message retry interval interval in seconds.

The filter value is the integer interval time.

**\*MSGRTYNBR**

Number of message retries.

The filter value is the integer number of retries.

**\*MSGUSRDATA**

Message exit user data.

The filter value is the user data string.

**\*NETPRTY**

Network connection priority in the range 0 through 9.

The filter value is the integer priority value.

**\*NPMSPEED**

Whether the channel supports fast non persistent messages.

The filter value is one of the following:

**\*FAST**

The channel supports fast non persistent messages.

**\*NORMAL**

The channel does not support fast non persistent messages.

**\*PUTAUT**

Whether the user identifier in the context information should be used.

The filter value is one of the following:

**\*DFT**

No authority check is made before the message is put on the destination queue.

**\*CTX**

The user identifier in the message context information is used to establish authority to put the message.

**\*QMID**

The internally generated unique name of the cluster queue manager.

The filter value is the unique name.

**\*QMTYPE**

The function of the cluster queue manager in the cluster.

The filter value is one of the following:

**\*REPOS**

Provides a full repository service.

**\*NORMAL**

Does not provide a full repository service.

**\*RCVEXIT**

Receive exit name.

The filter value is the exit name.

**\*RCVUSRDATA**

Receive exit user data.

The filter value is the user data string.

**\*SCYEXIT**

Security exit name.

The filter value is the exit name.

**\*SCYUSRDATA**

Security exit user data.

The filter value is the user data string.

**\*SEQNUMWRAP**

Maximum message sequence number.

The filter value is the integer sequence number.

**\*SHORTRTY**

Short retry count.

The filter value is the integer count.

**\*SHORTTMR**

short retry interval in seconds.

The filter value is the integer interval time.

**\*SNDEXIT**

Send exit name.

The filter value is the exit name.

**\*SNDUSRDATA**

Send exit user data.

The filter value is the user data string.



**\*SSLCAUTH**

Whether the channel should carry out client authentication over TLS.

The filter value is one of the following:

**\*REQUIRED**

Client authentication is required.

**\*OPTIONAL**

Client authentication is optional.

**\*SSLCIPH**

The CipherSpec using in TLS channel negotiation.

The filter value is the name of the CipherSpec.

**\*SSLPEER**

The X500 peer name used in TLS channel negotiation.

The filter value is the peer name.

**\*STATCHL**

Channel Statistics.

The filter value is one of the following:

**\*QMGR**

The collection of statistics data is inherited from the setting of the queue manager attribute STATCHL.

**\*OFF**

Statistics data collection for this channel is disabled.

**\*LOW**

Statistics data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Statistics data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Statistics data collection is turned on with a high ratio of data collection.

**\*STATUS**

The current status of the channel for this cluster queue manager.

The filter value is one of the following:

**\*STARTING**

The channel is waiting to become active.

**\*BINDING**

The channel is performing channel negotiation.

**\*INACTIVE**

The channel is not active.

**\*INITIALIZING**

The channel initiator is attempting to start a channel.

**\*RUNNING**

The channel is either transferring messages, or is waiting for messages to arrive on the transmission queue.

**\*STOPPING**

The channel is stopping, or a close request has been received.

**\*RETRYING**

A previous attempt to establish a connection has failed. The MCA will reattempt connection after the specified time interval.

**\*PAUSED**

The channel is waiting for the message-retry interval to complete before retrying an MQPUT operation.

**\*STOPPED**

The channel has either been manually stopped, or the retry limit has been reached.

**\*REQUESTING**

A local requester channel is requesting services from a remote MCA.

**\*SUSPEND**

Whether this cluster queue manager is suspended from the cluster or not.

The filter value is either \*NO or \*YES.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the channel.

**\*TMQNAME**

Transmission queue name.

The filter value is the name of the queue.

**\*USERID**

Task user identifier.

The filter value is the user identifier string.

**\*XMITQ**

Name of cluster transmission queue.

The filter value is the transmission queue name string.

 **WRKMQMCLQ (Work with MQ Cluster Queues)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Cluster Queues (WRKMQMCLQ) command allows you to work with cluster queues that are defined on the local queue manager.

**Parameters**

*Table 325. Command parameters*

Keyword	Description	Choices	Notes
<u>QNAME</u>	Queue name	Character value, *ALL	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, *DFT	Optional, Positional 2
<u>CLUSTER</u>	Cluster name	Character value, *ALL	Optional, Positional 3

Table 325. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 4
	Element 1: Filter keyword	*ALTDATA, *ALTTIME, *CLUSDATE, *CLUSQMGR, *CLUSQTYPE, *CLUSTER, *CLUSTIME, *DEFBIND, *DFTMSGPST, *DFTPTY, *PUTENBL, *QMID, *TEXT	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Queue name (QNAME)

Specifies the name or names of the cluster queue definitions.

#### **\*ALL**

All cluster queue definitions are selected.

#### **generic-queue-name**

Specify the generic name of the MQ cluster queue definitions. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all cluster queue definitions having names that start with the character string. You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered. You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

#### **queue-name**

Specify the name of the MQ cluster queue definition.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

#### **\*DFT**

Use the default queue manager.

#### **queue-manager-name**

Specify the name of the queue manager.

### Cluster name (CLUSTER)

Specifies the name of the cluster.

#### **\*ALL**

All cluster definitions are selected.

#### **generic-cluster-name**

Specify the generic name of the MQ cluster definitions. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all cluster definitions having names that start with the character string. You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered. You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**cluster-name**

Specify the name of the MQ cluster definition.

**Filter command (WHERE)**

This parameter can be used to selectively display only those cluster queues with particular cluster queue attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ALTDATE**

The date on which the definition or information was last altered.

The filter value is the data in the form yyyy-mm-dd.

**\*ALTTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*CLUSDATE**

The date on which the definition became available to the local queue manager.

The filter value is the date in the form yyyy-mm-dd.

**\*CLUSQMGR**

The name of the queue manager that hosts the queue.

The filter value is the name of the queue manager.

**\*CLUSQTYPE**

Cluster queue type.

The filter value is one of the following:

**\*LCL**

The cluster queue represents a local queue.

**\*ALS**

The cluster queue represents an alias queue.

**\*RMT**

The cluster queue represents a remote queue.

**\*MQMALS**

The cluster queue represents a queue manager alias.

**\*CLUSTER**

The name of the cluster that the queue is in.

The filter value is the name of the cluster.

**\*CLUSTIME**

The time at which the definition became available to the local queue manager.

The filter value is the time in the form hh:mm:ss.

**\*DEFBIND**

Default message binding.

The filter value is one of the following:

**\*OPEN**

The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

**\*NOTFIXED**

The queue handle is not bound to any particular instance of the cluster queue.

**\*GROUP**

When the queue is opened, the queue handle is bound to a specific instance of the cluster queue for as long as there are messages in a message group. All messages in a message group are allocated to the same destination instance.

**\*DFTMSGPST**

Default persistence of the messages put on this queue.

The filter value is one of the following:

**\*NO**

Messages on this queue are lost across a restart of the queue manager.

**\*YES**

Messages on this queue survive a restart of the queue manager.

**\*DFTPTY**

Default priority of the messages put on the queue.

The filter value is the integer priority value.

**\*PUTENBL**

Whether applications are permitted to put messages to the queue.

The filter value is one of the following:

**\*NO**

Messages cannot be added to the queue.

**\*YES**

Messages can be added to the queue by authorized applications.

**\*QMID**

Internally generated unique name of the queue manager that hosts the queue.

The filter value is the name of the queue manager.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the queue.

## **WRKMQCONN (Work with MQ Connections)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Connections (WRKMQCONN) command allows you to work with connection information for applications that are connected to the queue manager.

This enables you to display connection handles and end connections to the queue manager.

**Parameters**

Keyword	Description	Choices	Notes
<u>CONN</u>	Connection Identifier	Character value, <b>*ALL</b>	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2

Table 326. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*APPLDESC, *APPLTAG, *APPTYPE, *CHLNAME, *CONNAME, *PID, *TID, *UOWLOGDA, *UOWLOGTI, *UOWSTDA, *UOWSTTI, *URTYPE, *USERID	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Connection Identifier (CONN)

The connection identifiers to work with.

The possible values are:

**\*ALL**

All connection identifiers are selected.

**connection-id**

Specify the name of a specific connection identifier. The connection identifier is a 16 character hex string.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of a message queue manager.

### Filter command (WHERE)

This parameter can be used to selectively display only those queue manager connections with particular connection attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*APPLDESC**

The description of the application connected to the queue manager.

The filter value is the application description string.

**\*APPLTAG**

The tag of the application connected to the queue manager.

The filter value is the application tag string.

**\*APPTYPE**

The type of application connected to the queue manager.

The possible values are:

**\*DEF**

The attribute is unchanged.

**\*CICS**

Represents a CICS/400 application.



- \*UNIX**  
Represents a UNIX or Linux application.
- \*OS400**  
Represents an IBM i application.
- \*WINDOWS**  
Represents a Windows application.
- \*WINDOWS\_NT**  
Represents a Windows NT application.
- integer**  
User-defined application type in the range 65536 through 999999999.
- \*CHLNAME**  
The name of the channel that owns the connection.  
The filter value is the channel name.
- \*CONNAME**  
The connection name associated with the channel that owns the connection.  
The filter value is the connection name.
- \*PID**  
The process identifier of the application that is connected to the queue manager.  
The filter value is the process identifier integer.
- \*TID**  
The thread identifier of the application that is connected to the queue manager.  
The filter value is the thread identifier integer.
- \*UOWLOGDA**  
The date that the transaction associated with the connection first wrote to the log.  
The filter value is the date in the form yyyy-mm-dd.
- \*UOWLOGTI**  
The time that the transaction associated with the connection first wrote to the log.  
The filter value is the time in the form hh:mm:ss.
- \*UOWSTDA**  
The date that the transaction associated with the connection was started.  
The filter value is the date in the form yyyy-mm-dd.
- \*UOWSTTI**  
The time that the transaction associated with the connection was started.  
The filter value is the time in the form hh:mm:ss.
- \*URTYPE**  
The type of unit of recovery identifier as seen by the queue manager.  
The filter value is one of the following:
  - \*QMGR**  
A queue manager transaction.
  - \*XA**  
An externally coordinated transaction. This includes units of work which have been established using IBM i Start Commitment Control (STRCMTCTL).
- \*USERID**  
The user identifier associated with the connection.  
The filter value is user identifier name.

## IBM i WRKMQMJRN (Work Queue Manager Journals)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Work With Queue Manager Journals command (WRKMQMJRN) displays a list of all the journals which are associated with a specific queue manager. This command can be used, for example, to configure remote journaling for a multi-instance queue manager.

### Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 1

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager to work with journals.

#### queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

## IBM i WRKMQMLSR (Work with MQ Listeners)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Work with MQ Listener objects (WRKMQMLSR) command allows you to work with listener objects which are defined on the local queue manager.

This enables you to change, copy, create, delete, start, stop & display listener objects display and change authority to an MQ listener object.

This command also enables you to view the current status of all running listeners on the current system.

### Parameters

Keyword	Description	Choices	Notes
<u>OPTION</u>	Option	<b>*STATUS</b> , *OBJECT	Optional, Positional 1
<u>LSRNAME</u>	Listener name	<i>Character value, *ALL</i>	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 3

Table 328. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 4
	Element 1: Filter keyword	*ALTDATA, *ALTTIME, *BACKLOG, *CONTROL, *IPADDR, *PORT, *TEXT	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Option (OPTION)

This option enables you to select whether you want to information on listener status or listener object definitions.

The possible values are:

#### **\*STATUS**

Listener status information is displayed.

The parameters LSRNAME and WHERE are ignored. If MQMNAME is specified only the status of listeners running on the specified queue manager are displayed.

#### **\*OBJECT**

Listener object information is displayed.

### Listener name (LSRNAME)

The name or names of the listener objects.

The possible values are:

#### **\*ALL or \***

All listener objects are selected.

#### **generic-listener-name**

The generic name of the listener objects. A generic name is a character string followed by an asterisk (\*), for example ABC\*, and it selects all listener objects having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

#### **listener-name**

Specify the name of a single listener object.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

**queue-manager-name**

The name of a message queue manager.

**Filter command (WHERE)**

This parameter can be used to selectively display only those listener objects with particular listener attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ALTDATE**

The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

**\*ALLTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*BACKLOG**

The number of concurrent connection requests supported.

The filter value is the integer backlog value.

**\*CONTROL**

Whether the listener is started and stopped with the queue manager.

The filter value is one of the following:

**\*MANUAL**

The listener is not automatically started or stopped.

**\*QMGR**

The listener is started and stopped as the queue manager is started and stopped.

**\*STARTONLY**

The listener is started as the queue manager is started, but is not requested to stop when the queue manager is stopped.

**\*IPADDR**

The local IP Address to be used by the listener.

The filter value is the IP Address.

**\*PORT**


The port number to be used by the listener.

The filter value is the integer port value.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the listener.

 **WRKMQMSG (Work with MQ Messages)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Messages (WRKMQMSG) command lists the messages on a specified local queue and allows you to work with those messages. From the list of messages, you can display the contents of a message and its associated message descriptor (MQMD).

**Parameters**

Keyword	Description	Choices	Notes
<u>QNAME</u>	Queue name	Character value	Required, Positional 1

Table 329. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>FIRST</u>	First Message	1-30000, <b>1</b>	Optional, Positional 3
<u>MAXMSG</u>	Maximum number of messages	1-30000, <b>48</b>	Optional, Positional 4
<u>MAXMSGLEN</u>	Maximum message size	128-999999, <b>1024</b>	Optional, Positional 5

### Queue name (QNAME)

Specifies the name of the local queue.

The possible values are:

#### queue-name

Specify the name of the local queue.

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

#### \*DFT

Use the default queue manager.

#### queue-manager-name

Specify the name of the queue manager.

### First Message (FIRST)

Specifies the number of the first message to display.

The possible values are:

#### 1

The number of the first message to display is 1.

#### message-number

Specify the number of the first message to display ranging from 1 through 30 000.

### Maximum number of messages (MAXMSG)

Specifies the maximum number of messages to display.

The possible values are:

#### 48

Display a maximum of 48 messages.

#### count-value

Specify a value for the maximum number of messages to display ranging from 1 through 30 000.

### Maximum message size (MAXMSGLEN)

Specifies the maximum size of message data to display.

The size of a message, greater than the value specified, is suffixed by a plus (+) character to indicate that the message data is truncated.

The possible values are:

**1024**

The size of the message data is 1024 bytes.

**length-value**

Specify a value ranging from 128 through 999999.

**IBM i WRKMQMNL (Work with MQ Namelist)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Namelists (WRKMQMNL) command allows you to work with multiple namelist definitions that are defined on the local queue manager. This enables you to copy, change, display, delete, display authority and edit authority of an MQ namelist object.

**Parameters**

Table 330. Command parameters

Keyword	Description	Choices	Notes
<u>NAMELIST</u>	Namelist	Character value, <b>*ALL</b>	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATA, *ALTTIME, *NAMECNT, *NAMES, *TEXT	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

**Namelist (NAMELIST)**

Specifies the name or names of the namelists.

The possible values are:

**\*ALL**

All namelist definitions are selected.

**generic-namelist-name**

Specify the generic name of the MQ namelists. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all namelists having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**namelist-name**

Specify the name of the MQ namelist.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

The default queue manager is used.

**message-queue-manager-name**

Specify the name of the queue manager.

**Filter command (WHERE)**

This parameter can be used to selectively display only those namelists with particular namelist attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.



**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ALTDAT**

The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

**\*ALTTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*NAMECNT**

The number of names in the namelist.

The filter value is the integer number of names.

**\*NAMES**

The names in the namelist.

The filter value is the string name.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the queue.

## **WRKMQMPC (Work with MQ Processes)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Processes (WRKMQMPC) command allows you to work with multiple process definitions that are defined on the local queue manager. This enables you to copy, change, display, delete, display authority, and edit authority of an MQ process object.

**Parameters**

<i>Table 331. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>PRCNAME</u>	Process name	<i>Character value, *ALL</i>	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value, *DFT</i>	Optional, Positional 2

Table 331. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATA, *ALTTIME, *APPID, *APPTYPE, *ENVDATA, *TEXT, *USRDATA	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Process name (PRCNAME)

Specifies the name or names of the process definitions.

The possible values are:

#### **\*ALL**

All process definitions are selected.

#### **generic-process-name**

Specify the generic name of the MQ process definitions. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all process definitions having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

#### **process-name**

Specify the name of the MQ process definition.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

#### **queue-manager-name**

The name of a message queue manager.

### Filter command (WHERE)

This parameter can be used to selectively display only those processes with particular process attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

#### **\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ALTDATE**

The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

**\*ALLTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*APPID**

The name of the application to start.

The filter value is the name of the application.

**\*APPTYPE**

The type of application connected to the queue manager.

The possible values are:

**\*DEF**

The attribute is unchanged.

**\*CICS**

Represents a CICS/400 application.

**\*UNIX**

Represents a UNIX or Linux application.

**\*OS400**

Represents an IBM i application.

**\*WINDOWS**

Represents a Windows application.

**\*WINDOWS\_NT**

Represents a Windows NT application.

**integer**

User-defined application type in the range 65536 through 999999999.

**\*ENVDATA**

Environment data pertaining to the application.

The filter value is the environment data.

**\*TEXT**


Descriptive comment.

The filter value is the text description of the queue.

**\*USRDATA**

User data pertaining to the application.

The filter value is the user data.

 **WRKMQMQ (Work with MQ Queues)**

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Queues (WRKMQMQ) command provides the function to work with multiple queues that are defined on the local queue manager. Using this command you can copy, change, display, delete, display authority and edit authority of an MQ Queue object.

**Parameters**

Keyword	Description	Choices	Notes
<u>QNAME</u>	Queue name	Character value, <b>*ALL</b>	Optional, Positional 1
<u>QTYPE</u>	Queue type	<b>*ALL</b> , *ALS, *LCL, *MDL, *RMT	Optional, Positional 2
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 3

Table 332. Command parameters (continued)

Keyword	Description	Choices	Notes
<u>CLUSTER</u>	Cluster name	Character value, <b>*ALL</b>	Optional, Positional 4
<u>CLUSNL</u>	Cluster namelist name	Character value, <b>*ALL</b>	Optional, Positional 5
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 6
	Element 1: Filter keyword	*ACCTQ, *ALTDATE, *ALTTIME, *BKTTHLD, *BKTQNAME, *CLUSDATE, *CLUSNL, *CLUSQMGR, *CLUSQTYPE, *CLUSTER, *CLUSTIME, *CLWLPRTY, *CLWLRANK, *CLWLUSEQ, *CRDATE, *CRTIME, *CURDEPTH, *DEFBIND, *DFTPUTRESP, *DFNTYPE, *DFTMSGPST, *DFTPTY, *DFTSHARE, *DISTLIST, *FULLEVT, *GETDATE, *GETENBL, *GETTIME, *HDNBKTCNT, *HIGHEVT, *HIGHTHLD, *INITQNAME, *IPPROCS, *JOBS, *LOWEVT, *LOWTHLD, *MAXDEPTH, *MAXMSGLEN, *MEDIAREC, *MONQ, *MSGAGE, *MSGDLYSEQ, *MSGREADAHD, *NPMCLASS, *OPPROCS, *PRCNAME, *PROPCTL, *PUTDATE, *PUTENBL, *PUTTIME, *QMID, *QTYPE, *RMTMQMNAME, *RMTQNAME, *RTNITV, *SHARE, *SRVEVT, *SRVITV, *STATQ, *TARGTYPE, *TEXT, *TGTQNAME, *TMQNAME, *TRGDATA, *TRGDEPTH, *TRGENBL, *TRGMSGPTY, *TRGTYPE, *UNCOM, *USAGE	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
Element 3: Filter value	Character value		

### Queue name (QNAME)

The name or names of the queues to be selected. The queues selected by this parameter can be further limited to a particular type, if the QTYPE keyword is specified.

The possible values are:

**\*ALL**

All queues are selected.

**generic-queue-name**

Specify the generic name of the queues to be selected. A generic name is a character string, followed by an asterisk (\*). For example ABC\*, it selects all queues having names that start with the character string.

Specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**queue-name**

Specify the name of the queue.

## Queue type (QTYPE)

This parameter can be specified to limit the queues that are displayed to a particular type.

The possible values are:

**\*ALL**

All queue types.

**\*ALS**

Alias queues.

**\*LCL**

Local queues.

**\*MDL**

Model queues.

**\*RMT**

Remote queues.

## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

## Cluster name (CLUSTER)

This parameter can be specified to limit the queues that are displayed to be members of a particular cluster.

The possible values are:

**\*ALL**

All clusters.

**generic-cluster-name**

The generic name of a cluster.

**cluster-name**

The name of a cluster.

## Cluster namelist name (CLUSNL)

This parameter can be specified to limit the queues that are displayed to be members of clusters within a cluster namelist.

The possible values are:

### **\*ALL**

All cluster namelists.

### **generic-cluster-namelist-name**

The generic name of a cluster namelist.

### **cluster-namelist-name**

The name of a cluster namelist.

## Filter command (WHERE)

This parameter can be used to selectively display only those queues with particular queue attributes.

The parameter takes three arguments, a keyword, an operator, and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

### **\*GT**

Greater than.

Applicable to integer and non-generic string values.

### **\*LT**

Less than.

Applicable to integer and non-generic string values

### **\*EQ**

Equal to.

Applicable to integer and non-generic string values.

### **\*NE**

Not equal to.

Applicable to integer and non-generic string values.

### **\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

### **\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

### **\*LK**

Like.

Applicable to generic string values.

### **\*NL**

Not like.

Applicable to generic string values.

### **\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ACCTQ**

Queue Accounting.

The filter value is one of the following values:

**\*QMGR**

Accounting data collection is based upon the setting of the queue manager attribute ACCTQ.

**\*OFF**

Accounting data collection for this queue is disabled.

**\*ON**

Accounting data collection is enabled for this queue.

**\*ALTDATA**

The date on which the definition or information was last altered.

The filter value is the data in the form yyyy-mm-dd.

**\*ALTTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*BKTTHLD**

Backout threshold.

The filter value is the integer threshold value.

**\*BKTQNAME**

Backout requeue name.

The filter value is the name of the queue.

**\*CLUSDATE**

The date on which the definition became available to the local queue manager.

The filter value is the date in the form yyyy-mm-dd.

**\*CLUSNL**

The namelist that defines the clusters that the queue is in.

The filter value is the name of the namelist.

**\*CLUSQMGR**

The name of the queue manager that hosts the queue.

The filter value is the name of the queue manager.

**\*CLUSQTYPE**

Cluster queue type.

The filter value is one of the following values:



- \*LCL**  
The cluster queue represents a local queue.
- \*ALS**  
The cluster queue represents an alias queue.
- \*RMT**  
The cluster queue represents a remote queue.
- \*MQMALS**  
The cluster queue represents a queue manager alias.
- \*CLUSTER**  
The name of the cluster that the queue is in.  
The filter value is the name of the cluster.
- \*CLUSTIME**  
The time at which the definition became available to the local queue manager.  
The filter value is the time in the form hh:mm:ss.
- \*CLWLPRTY**  
Cluster workload priority.  
The filter value is the integer priority.
- \*CLWLANK**  
Cluster workload rank.  
The filter value is the integer rank.
- \*CLWLUSEQ**  
Cluster workload queue use.  
The filter value is one of the following values:
  - \*QMGR**  
The value is inherited from the Queue Manager CLWLUSEQ attribute.
- \*LOCAL**  
The local queue is the sole target of the MQPUT.
- \*ANY**  
The queue manager treats such a local queue as another instance of the cluster queue for the purposes of workload distribution.
- \*CRDATE**  
The date on which the queue was created.  
The filter value is the date in the form yyyy-mm-dd.
- \*CRTIME**  
The time at which the queue was created.  
The filter value is the time in the form hh:mm:ss.
- \*CURDEPTH**  
Current depth of queue.  
The filter value is the integer depth value.
- \*DEFBIND**  
Default message binding.  
The filter value is one of the following values:
  - \*OPEN**  
The queue handle is bound to a specific instance of the cluster queue when the queue is opened.
- \*NOTFIXED**  
The queue handle is not bound to any instance of the cluster queue.

**\*GROUP**

When the queue is opened, the queue handle is bound to a specific instance of the cluster queue for as long as there are messages in a message group. All messages in a message group are allocated to the same destination instance.

**\*DFTPUTRESP**

Default Put Response.

The filter value is one of the following values:

**\*SYNC**

The put operation is issued synchronously.

**\*ASYNC**

The put operation is issued asynchronously.

**\*DFNTYPE**

Queue definition type.

The filter value is one of the following values:

**\*PREDEF**

Predefined queue.

**\*PERMDYN**

Permanent dynamic queue.

**\*TEMPDYN**

Temporary dynamic queue.

**\*DFTMSGPST**

Default persistence of the messages put on this queue.

The filter value is one of the following values:

**\*NO**

Messages on this queue are lost across a restart of the queue manager.

**\*YES**

Messages on this queue survive a restart of the queue manager.

**\*DFTPTY**

Default priority of the messages put on the queue.

The filter value is the integer priority value.

**\*DFTSHARE**

Default share option on a queue opened for input.

The filter value is one of the following values:

**\*NO**

The open request is for exclusive input from the queue.

**\*YES**

The open request is for shared input from the queue.

**\*DISTLIST**

Whether distribution lists are supported by the partner queue manager.

The filter value is one of the following values:

**\*NO**

Distribution lists are not supported by the partner queue manager.

**\*YES**

Distribution lists are supported by the partner queue manager.

**\*FULLEVT**

Whether Queue Depth Full events are generated.

The filter value is one of the following values:

**\*NO**

Queue Depth Full events are not generated.

**\*YES**

Queue Depth Full events are generated.

**\*GETDATE**

The date on which the last message was got from the queue since queue manager start. This field is only present when Queue Monitoring is not set to \*OFF.

The filter value is the data in the form yyyy-mm-dd.

**\*GETENBL**

Whether applications are permitted to get messages from the queue.

The filter value is one of the following values:

**\*NO**

Applications cannot retrieve messages from the queue.

**\*YES**

Authorized applications can retrieve messages from the queue.

**\*GETTIME**

The time at which the last message was got from the queue since queue manager start. This field is only present when Queue Monitoring is not set to \*OFF.

The filter value is the time in the form hh:mm:ss.

**\*HDNBKTCNT**

Whether the backout count is hardened.

The filter value is one of the following values:

**\*NO**

The backout count is not hardened.

**\*YES**

The backout count is hardened.

**\*HIGHEVT**

Whether Queue Depth High events are generated.

The filter value is one of the following values:

**\*NO**

Queue Depth High events are not generated.

**\*YES**

Queue Depth High events are generated.

**\*HIGHTHLD**

Queue Depth High event generation threshold.

The filter value is the integer threshold value.

**\*INITQNAME**

Initiation queue.

The filter value is the name of the queue.

**\*IPPROCS**

Number of handles indicating that the queue is open for input.

The filter value is the integer number of handles.

**\*JOBS**

The current number of jobs that have the queue open.

The filter value is the integer number of jobs.

**\*LOWEVT**

Whether Queue Depth Low events are generated.

The filter value is one of the following values:

**\*NO**

Queue Depth Low events are not generated.

**\*YES**

Queue Depth Low events are generated.

**\*LOWTHLD**

Queue Depth Low event generation threshold.

The filter value is the integer threshold value.

**\*MAXDEPTH**

Maximum depth of queue.

The filter value is the integer number of messages.

**\*MAXMSGLEN**

Maximum message length.

The filter value is the integer message length.

**\*MEDIAREC**

The journal receiver containing the last media recovery image. This field is only present for local queues.

The filter value is the journal receiver string.

**\*MONQ**

Online Monitoring Data.

The filter value is one of the following values:

**\*QMGR**

The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONQ.

**\*OFF**

Online Monitoring Data collection for this queue is disabled.

**\*LOW**

Monitoring data collection is turned on with a low ratio of data collection.

**\*MEDIUM**

Monitoring data collection is turned on with a moderate ratio of data collection.

**\*HIGH**

Monitoring data collection is turned on with a high ratio of data collection.

**\*MSGAGE**

The age in seconds of the oldest message on the Queue. This field is only present when Queue Monitoring is not set to \*OFF.

The filter value is the integer message age.

**\*MSGDLYSEQ**

Message delivery sequence.

The filter value is one of the following values:

**\*PTY**

Messages are delivered in FIFO order within priority.

**\*FIFO**

Messages are delivered in FIFO order regardless of priority.

**\*NPMCLASS**

Non-persistent message class.

The filter value is one of the following values:

**\*NORMAL**

Non-persistent message class is normal.

**\*HIGH**

Non-persistent message class is high.

**\*MSGREADAHD**

Message read ahead.

The filter value is one of the following values:

**\*DISABLED**

Read ahead is disabled.

**\*NO**

Non-persistent messages are not sent to the client ahead of an application requesting them.

**\*YES**

Non-persistent messages are sent to the client ahead of an application requesting them.

**\*OPPROCS**

Number of handles indicating that the queue is open for output.

The filter value is the integer number of handles.

**\*PRCNAME**

Process name.

The filter value is the name of the process.

**\*PROPCTL**

Message Property Control.

The filter value is one of the following values:

**\*COMPAT**

Compatibility mode

**\*NONE**

No properties are returned to the application.

**\*ALL**

All properties are returned to the application.

**\*FORCE**

Properties are returned to the application in one or more MQRFH2 headers.

**\*V6COMPAT**

An MQRFH2 header is returned formatted as it was sent. Its code page and encoding might be altered. If the message is a publication it might have a psc folder inserted into its contents.

**\*PUTDATE**

The date on which the last message was put to the queue since queue manager start. This field is only present when Queue Monitoring is not set to \*OFF.

The filter value is the data in the form yyyy-mm-dd.

**\*PUTENBL**

Whether applications are permitted to put messages to the queue.

The filter value is one of the following values:

**\*NO**

Messages cannot be added to the queue.

**\*YES**

Messages can be added to the queue by authorized applications.

**\*PUTTIME**

The time at which the last message was put to the queue since queue manager start. This field is only present when Queue Monitoring is not set to \*OFF.

The filter value is the time in the form hh:mm:ss.

**\*QMID**

Internally generated unique name of the queue manager that hosts the queue.

The filter value is the name of the queue manager.

**\*QTYPE**

Queue type.

The filter value is one of the following values:

**\*LCL**

Local queue.

**\*ALS**

Alias queue.

**\*RMT**

Remote queue.

**\*MDL**

Model queue.

**\*RMTMQMNAME**

Remote queue manager name.

The filter value is the name of the queue manager.

**\*RMTQNAME**

Name of the local queue, as known by the remote queue manager.

The filter value is the name of the queue.

**\*RTNITV**

Retention interval.

The filter value is the integer interval value.

**\*SHARE**

Whether the queue can be shared.

The filter value is one of the following values:

**\*NO**

Only a single application instance can open the queue for input.

**\*YES**

More than one application instance can open the queue for input.

**\*SRVEVT**

Whether service interval events are generated.

The filter value is one of the following values:

**\*HIGH**

Service Interval High events are generated.

**\*OK**

Service Interval OK events are generated.

**\*NONE**

No service interval events are generated.

**\*SRVITV**

Service interval event generation threshold.

The filter value is the integer threshold value.

**\*STATQ**

Statistics data.

The filter value is one of the following values:

**\*QMGR**

Statistics data collection is based upon the setting of the queue manager attribute STATQ.

**\*OFF**

Statistics data collection for this queue is disabled.

**\*ON**

Statistics data collection is enabled for this queue.

**\*TARGTYPE**

Target Type.

The filter value is one of the following values:

**\*QUEUE**

Queue object.

**\*TOPIC**

Topic object.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the queue.

**\*TGTQNAME**

Target queue for which this queue is an alias.

The filter value is the name of the queue.

**\*TMQNAME**

Transmission queue name.

The filter value is the name of the queue.

**\*TRGDATA**

Trigger data.

The filter value is the text of the trigger message.

**\*TRGDEPTH**

Trigger depth.

The filter value is the integer number of messages.

**\*TRGENBL**

Whether triggering is enabled.

The filter value is one of the following values:

**\*NO**

Triggering is not enabled.

**\*YES**

Triggering is enabled.

**\*TRGMSGPTY**

Threshold message priority for triggers.

The filter value is the integer priority value.

**\*TRGTYPE**

Trigger type.

The filter value is one of the following values:

**\*FIRST**

When the number of messages on the queue goes from 0 to 1.

**\*ALL**

Every time a message arrives on the queue.

**\*DEPTH**

When the number of messages on the queue equals the value of the TRGDEPTH attribute.

**\*NONE**

No trigger messages are written.

**\*UNCOM**

The number of uncommitted changes pending for the queue.

The filter value is one of the following values:

**\*NO**

There are no uncommitted changes pending.

**\*YES**

There are uncommitted changes pending.

**\*USAGE**

Whether the queue is a transmission queue.

The filter value is one of the following values:

**\*NORMAL**

The queue is not a transmission queue.

**\*TMQ**

The queue is a transmission queue.

## WRKMQMSPL (Work with MQM Security Policies)

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQM Security Policies (WRKMQMSPL) command lists all security policies for a queue manager.

Security Policies are used by Advanced Message Security to control how messages should be protected when being put, browsed, or destructively removed from queues.

Additionally, [DSPMQM](#) displays whether security policies are enabled for the queue manager. Note that the Advanced Message Security license must be installed when the queue manager was started for this to occur.

**Parameters**

Keyword	Description	Choices	Notes
<a href="#">OUTPUT</a>	Output	<b>*</b> , *PRINT	Optional, Positional 1
<a href="#">MQMNAME</a>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 2

**Output (OUTPUT)**

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.



The possible values are:

**\***

Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

**\*PRINT**

A detailed list of the users and their authorities registered with the selected authority profile record is printed with the job's spooled output.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

## WRKMQMSTS (Work with Queue Status)

### Where allowed to run

All environments (\*ALL)

### Threadsafe

Yes

The Work with Queue Status (WRKMQMSTS) command lists the jobs which have an IBM MQ queue currently open. The command allows you to determine what options a queue was opened with and also allows you to check to see which channels and connections have a queue open.

### Parameters

Keyword	Description	Choices	Notes
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 1
<u>QNAME</u>	Queue name	Character value	Optional, Positional 2
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*APPLDESC, *APPLTAG, *BROWSE, *CHLNAME, *CONNNAME, *INPUT, *INQUIRE, *JOB, *OUTPUT, *SET, *URTYPE	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

**\*DFT**

Use the default queue manager.

**queue-manager-name**

Specify the name of the queue manager.

**Queue name (QNAME)**

Specifies the name of the local queue.

The possible values are:

**queue-name**

Specify the name of the local queue.

**Filter command (WHERE)**

This parameter can be used to selectively display only the jobs with particular attributes that have the queue open.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*APPLDESC**

The description of the application which has the queue open.

The filter value is the application description string.

**\*APPLTAG**

The tag of the application which has the queue open.

The filter value is the application tag string.

**\*BROWSE**

Whether the job has the queue open for browsing.

The filter value is either \*NO or \*YES.

**\*CHLNAME**

The name of the channel which has the queue open.

The filter value is the channel name.

**\*CONNNAME**

The connection name of the channel which has the queue open.

The filter value is the connection name.

**\*INPUT**

Whether the job has the queue open for input.

The filter value is one of the following:

**\*NO**

The job does not have the queue open for input.

**\*SHARED**

The job has the queue open for shared input.

**\*EXCL**

The job has the queue open for exclusive input.

**\*INQUIRE**

Whether the job has the queue open for inquiry.

The filter value is either \*NO or \*YES.

**\*JOB**

The name of the job which has the queue open.

The filter value is the job name.

**\*OUTPUT**

Whether the job has the queue open for output.

The filter value is either \*NO or \*YES.

**\*SET**

Whether the job has the queue open for set.

The filter value is either \*NO or \*YES.

**\*URTYPE**

The type of unit of work recovery identifier.

The filter value is one of the following:

**\*QMGR**

Queue manager unit of work recovery identifier.

**\*XA**

XA unit of work recovery identifier.

## WRKMQMSUB (Work with MQ Subscriptions)

**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Subscriptions (WRKMQMSUB) command allows you to work with multiple subscriptions that are defined on the local queue manager. This enables you to copy, change, display and delete IBM MQ subscriptions.

**Parameters**

<i>Table 335. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>SUBNAME</u>	Subscription name	Character value, <b>*ALL</b>	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*DEST, *DESTCLASS, *DESTCRLID, *DESTMQM, *EXPIRY, *PSPROP, *PUBACCT, *PUBAPPID, *PUBPTY, *REQONLY, *SELECTOR, *SELTYPE, *SUBSCOPE, *SUBID, *TOPICOBJ, *TOPICSTR, *USERDATA, *VARUSER, *WSHEMA	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

**Subscription name (SUBNAME)**

Specifies the name or names of the subscriptions.

The possible values are:

**\*ALL**

All subscriptions are selected.

**generic-subscription-name**

Specify the generic name of the MQ subscriptions. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all subscriptions having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

**subscription-name**

Specify the name of the MQ subscription.

**Message Queue Manager name (MQMNAME)**

Specifies the name of the Queue Manager.

The possible values are:

**\*DFT**

Use the default Queue Manager.

**queue-manager-name**

The name of a Queue Manager.

**Filter command (WHERE)**

This parameter can be used to selectively display only those subscriptions with particular subscription attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*DEST**

The destination queue for messages published to this subscription.

The filter value is the name of the queue.

**\*DESTCLASS**

Specifies whether this is a managed subscription.

The filter value is one of the following:

**\*MANAGED**

The destination is managed.

**\*PROVIDED**

The destination is a queue.

**\*DESTCRLID**

The correlation identifier for messages published to this subscription.

The filter value is the 48 character hexadecimal string representing the 24 byte correlation identifier.

**\*DESTMQM**

The destination queue manager for messages published to this subscription.

The filter value is the name of the queue manager.

**\*EXPIRY**

The expiry time of the subscription.

The filter value is the integer expiry time.

**\*PSPROP**

The manner in which publish / subscribe related message properties are added to messages sent to this subscription.

The filter value is one of the following:

**\*NONE**

Publish / subscribe properties are not added to the message.

**\*COMPAT**

Publish / subscribe properties are added to the message to maintain compatibility with V6 Publish / Subscribe.

**\*RFH2**

Publish / subscribe properties are added to the message within an RFH 2 header.

**\*PUBACCT**

The accounting token for messages published to this subscription.

The filter value is the 64 character hexadecimal string representing the 32 byte publish accounting token.

**\*PUBAPPID**

The publish application identity for messages published to this subscription.

The filter value is the publish application identifier.

**\*PUBPTY**

The priority of the message sent to this subscription.

The filter value is the integer priority.

**\*REQONLY**

Whether the subscriber will poll for updates via MQSUBRQ API, or whether all publications are delivered to this subscription.

The filter value is one of the following:

**\*YES**

Publications are only delivered to this subscription in response to an MQSUBRQ API.

**\*NO**

All publications on the topic are delivered to this subscription.

**\*SELECTOR**

The SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

The filter value is the selector string.

**\*SELTYPE**

The type of SQL 92 selector string that has been specified.

The filter value is one of the following:

**\*NONE**

No selector has been specified.

**\*STANDARD**

A selector string has been specified that only references properties of the message and uses the standard selector syntax.

**\*EXTENDED**

A selector string has been specified that uses extended selectors syntax, typically by referencing the content of the message. Selector strings of this type cannot be handled internally by the queue manager; the use of extended message selectors can only be handled by another program, such as IBM Integration Bus.

**\*SUBSCOPE**

Determines whether this subscription is forwarded to other queue managers, so that the subscriber receives messages published at those other queue managers.

The filter value is one of the following:

**\*ALL**

The subscription is forwarded to all queue managers directly connected through a publish/subscribe collective or hierarchy.

**\*QMGR**

The subscription forwards messages published on the topic only within this queue manager.

**Note:** Individual subscribers can only restrict **SUBSCOPE**. If the parameter is set to ALL at topic level, then an individual subscriber can restrict it to QMGR for this subscription. However, if the parameter is set to QMGR at topic level, then setting an individual subscriber to ALL has no effect.

**\*SUBID**

The subscription identifier associated with the subscription.

The filter value is the 48 character hexadecimal string representing the 24 byte subscription identifier.

**\*TOPICOBJ**

The topic object associated with the subscription.

The filter value is the name of the topic object.

**\*TOPICSTR**

The topic string associated with the subscription.

The filter value is the topic string.

**\*USERDATA**

The user data associated with the subscription.

The filter value is the user data.

**\*VARUSER**

Whether user profiles other than the creator of the subscription can connect to it.

The filter value is one of the following:

**\*ANY**

Any user profiles can connect to the subscription.

**\*FIXED**

Only the user profile that created the subscription can connect to it.

**\*WSHEMA**

The schema to be used when interpreting wildcard characters in the topic string.

The filter value is one of the following:

**\*TOPIC**

Wildcard characters represent portions of the topic hierarchy.

**\*CHAR**

Wildcard characters represent portions of strings.

**IBM i WRKMQMSVC (Work with MQ Service object)****Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The Work with MQ Service objects (WRKMQMSVC) command allows you to work with multiple service objects that are defined on the local queue manager.

This enables you to start, stop, change, copy, create, delete, display, and display and change authority to an MQ service object.



## Parameters

Table 336. Command parameters			
Keyword	Description	Choices	Notes
<u>SVCNAME</u>	Service name	Character value, <b>*ALL</b>	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATA, *ALTTIME, *CONTROL, *ENDARG, *ENDCMD, *STDERR, *STDOUT, *STRARG, *STRCMD, *TEXT, *TYPE	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Service name (SVCNAME)

The name or names of the service objects.

The possible values are:

#### **\*ALL or \***

All service objects are selected.

#### **generic-service-name**

The generic name of the service objects. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all service objects having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

#### **service-name**

Specify the name of a single service object.

### Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

#### **\*DFT**

Use the default queue manager.

#### **queue-manager-name**

The name of a message queue manager.

### Filter command (WHERE)

This parameter can be used to selectively display only those service objects with particular service attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ALTDATE**

The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

**\*ALTTIME**

The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*CONTROL**

Whether the service is started and stopped with the queue manager.

The filter value is one of the following:

**\*MANUAL**

The service is not automatically started or stopped.

**\*QMGR**

The service is started and stopped as the queue manager is started and stopped.

**\*STARTONLY**

The service is started as the queue manager is started, is not be requested to stop when the queue manager is stopped.

**\*ENDARG**

The arguments passed to the end program when the service is requested to stop.

The filter value is the arguments string.

**\*ENDCMD**

The name of the executable to run when the service is requested to stop.

The filter value is the program name string.

**\*STDERR**

The standard error path.

The filter value is the path name.

**\*STDOUT**

The standard output path.

The filter value is the path name.

**\*STRARG**

The arguments passed to the program at startup.

The filter value is the arguments string.

**\*STRCMD**

The name of the program to run.

The filter value is the program name string.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the service.

**\*TYPE**

Mode in which to run service.

The filter value is one of the following:

**\*CMD**

When started the command is executed but no status is collected or displayed.

**\*SVR**

The status of the executable started is monitored and displayed.

 **WRKMQM TOP (Work with MQ Topics)****Where allowed to run**

All environments (\*ALL)

## Threadsafe

Yes

The Work with MQ Topics (WRKMQMTOP) command allows you to work with multiple topic objects that are defined on the local queue manager. This enables you to copy, change, display, delete, display authority, edit authority, record and recover an MQ topic object.

## Parameters

Table 337. Command parameters

Keyword	Description	Choices	Notes
<u>TOPNAME</u>	Topic name	Character value, <b>*ALL</b>	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	Character value, <b>*DFT</b>	Optional, Positional 2
<u>WHERE</u>	Filter command	Single values: <b>*NONE</b> Other values: <i>Element list</i>	Optional, Positional 3
	Element 1: Filter keyword	*ALTDAT, *ALTTIME, *DFTMSGPST, *DFTPTY, *DFTPUTRESP, *DURSUB, *MGDDURMDL, *MGDNDURMDL, *NPMSGDLV, *PMSGDLV, *PUBENBL, *SUBENBL, *TEXT, *TOPNAME, *TOPICSTR, *WILDCARD	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	Character value	

### Topic name (TOPNAME)

Specifies the name or names of the topic objects.

The possible values are:

#### **\*ALL**

All topic objects are selected.

#### **generic-topic-name**

Specify the generic name of the MQ topic objects. A generic name is a character string followed by an asterisk (\*). For example ABC\*, it selects all topic objects having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the uppercase and lowercase versions of a generic name on a single panel, without requesting all the names.

#### **topic-name**

Specify the name of the MQ topic object.

### Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

**\*DFT**

Use the default Queue Manager.

**queue-manager-name**

The name of a Queue Manager.

**Filter command (WHERE)**

This parameter can be used to selectively display only those topics with particular topic attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

**\*GT**

Greater than.

Applicable to integer and non-generic string values.

**\*LT**

Less than.

Applicable to integer and non-generic string values

**\*EQ**

Equal to.

Applicable to integer and non-generic string values.

**\*NE**

Not equal to.

Applicable to integer and non-generic string values.

**\*GE**

Greater than or equal to.

Applicable to integer and non-generic string values.

**\*LE**

Less than or equal to.

Applicable to integer and non-generic string values.

**\*LK**

Like.

Applicable to generic string values.

**\*NL**

Not like.

Applicable to generic string values.

**\*CT**

Contains.

Applicable to non-generic list values.

**\*EX**

Excludes.

Applicable to non-generic list values.

**\*CTG**

Contains generic.

Applicable to generic list values.

**\*EXG**

Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

**\*ALTDATE**

The date on which the object or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

**\*ALLTIME**

The time at which the object or information was last altered.

The filter value is the time in the form hh:mm:ss.

**\*DFTMSGPST**

The default persistence for messages associated with this topic.

The filter value is one of the following:

**\*ASPARENT**

Default persistence for messages is inherited from the parent topic.

**\*NO**

Messages associated with this topic are lost across a restart of the queue manager.

**\*YES**

Messages associated with this topic survive a restart of the queue manager.

**\*DFTPURRESP**

Default Put Response.

The filter value is one of the following:

**\*ASPARENT**

The default response type is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

**\*SYNC**

Put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are issued as if MQPMO\_SYNC\_RESPONSE had been specified instead.

**\*ASYNC**

Put operations to the queue that specify MQPMO\_RESPONSE\_AS\_Q\_DEF are always issued as if MQPMO\_ASYNC\_RESPONSE had been specified instead.

**\*DFTPTY**

Default priority for messages associated with this topic.

The filter value is the integer priority value.

**\*DURSUB**

Specifies whether the topic permits durable subscriptions.

The filter value is one of the following:

**\*ASPARENT**

This topic behaves in the same way as the parent topic.

**\*NO**

This topic does not permit durable subscriptions.

**\*YES**

This topic does permit durable subscriptions.

**\*MGDDURMDL**

The name of the model queue for managed durable subscriptions.

The filter value is the name of the queue.

**\*MGDNDURMDL**

The name of the model queue for managed non-durable subscriptions.

The filter value is the name of the queue.

**\*NPMSGDLV**

Specifies the delivery mechanism for non-persistent messages published to this topic.

The filter value is one of the following:

**\*ALL**

All non-persistent messages are published to this topic.

**\*ALLDUR**

All durable non-persistent messages are published to this topic.

**\*ALLAVAIL**

All available non-persistent messages are published to this topic.

**\*ASPARENT**

This topic behaves in the same way as the parent topic.

**\*PMSGDLV**

Specifies the delivery mechanism for persistent messages published to this topic.

The filter value is one of the following:

**\*ALL**

All persistent messages are published to this topic.

**\*ALLDUR**

All durable persistent messages are published to this topic.

**\*ALLAVAIL**

All available persistent messages are published to this topic.

**\*ASPARENT**

This topic behaves in the same way as the parent topic.

**\*PUBENBL**

Specifies whether the topic allows publications.

The filter value is one of the following:

**\*ASPARENT**

This topic behaves in the same way as the parent topic.

**\*NO**

This topic does not have publication enabled.

**\*YES**

This topic does have publication enabled.

**\*SUBENBL**

Specifies whether the topic allows subscriptions.

The filter value is one of the following:

**\*ASPARENT**

This topic behaves in the same way as the parent topic.

**\*NO**

This topic does not allow subscriptions.

**\*YES**

This topic allows subscriptions.

**\*TEXT**

Descriptive comment.

The filter value is the text description of the topic.

**\*TOPNAME**

The name of the topic.

The filter value is the name of the topic.

**\*TOPICSTR**

The topic string, used to identify the topic node.

The filter value is a character string.

**\*WILDCARD**

Specifies the behavior of wildcard subscriptions with respect to this topic.

The filter value is one of the following:

**\*PASSTHRU**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will receive publications made to this topic and to topic strings more specific than this topic.

**\*BLOCK**

Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will not receive publications made to this topic or to topic strings more specific than this topic.


**WRKMQMTRN (Work with MQ Transactions)**
**Where allowed to run**

All environments (\*ALL)

**Threadsafe**

Yes

The work with MQ transactions (WRKMQMTRN) command lists details of internally or externally coordinated in-doubt transactions.

**Parameters**

<i>Table 338. Command parameters</i>			
<b>Keyword</b>	<b>Description</b>	<b>Choices</b>	<b>Notes</b>
<u>TYPE</u>	Transaction type	<b>*ALL</b> , *EXT, *INT, *MQI, *XA, *OS400	Optional, Positional 1
<u>MQMNAME</u>	Message Queue Manager name	<i>Character value</i> , <b>*DFT</b>	Optional, Positional 2

**Transaction type (TYPE)**

Specifies the type of transactions.

**\*ALL**

Requests details of all the in-doubt transactions.

**\*EXT**

Requests details of externally coordinated, in-doubt transactions. Such transactions are those for which IBM MQ has been asked to prepare to commit, but has not yet been informed of the transaction outcome.

**\*INT**

Requests details of internally coordinated, in-doubt transactions. Such transactions are those for which each resource manager has been asked to prepare to commit, but IBM MQ has yet to inform the resource managers of the transaction outcome.



## Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

### **\*DFT**

Use the default queue manager.




### **message-queue-manager-name**

Specify the name of the queue manager.

## MFT commands reference

All Managed File Transfer (MFT) commands are listed with links to their detailed descriptions.

Command name	Purpose
<b>Commands for migration:</b>	
<b>Commands for configuration:</b>	
<a href="#">fteChangeDefaultConfigurationOptions</a>	Change the default configuration options that you want Managed File Transfer to use
<a href="#">fteCreateAgent</a>	Create a Managed File Transfer Agent
<a href="#">fteCreateBridgeAgent</a>	Create a Managed File Transfer protocol bridge agent
<a href="#">fteCreateCDAgent</a>	Create a Managed File Transfer Connect:Direct bridge agent
<a href="#">fteCreateEnvironment</a>	Set the environment variable for the configuration and transfer of files for the Redistributable Managed File Transfer Agent.
<a href="#">fteCreateLogger</a>	Create a Managed File Transfer logger
<a href="#">fteDefine</a>	Generate the configuration scripts necessary to define the specified objects.
<a href="#">fteDelete</a>	Generate the configuration scripts necessary to remove the specified objects.
<a href="#">fteDeleteAgent</a>	Delete a particular Managed File Transfer Agent
<a href="#">fteDeleteLogger</a>	Delete a Managed File Transfer logger
<a href="#">fteModifyAgent</a>	Windows only. Modify an agent, Connect:Direct bridge agent, or protocol bridge agent to run as a Windows service.
<a href="#">fteModifyLogger</a>	Windows only. Modify the logger to run as a Windows service.
<a href="#">fteSetupCommands</a>	Specify the details of the queue manager that connects to the IBM MQ network when you issue commands
<a href="#">fteSetupCoordination</a>	Configure a Managed File Transfer coordination queue manager
<b>Commands for administration:</b>	
<a href="#">fteCancelTransfer</a>	Cancel a file transfer
<a href="#">fteCleanAgent</a>	Clean up the queues used by an agent
<a href="#">fteClearMonitorHistory</a>	Clear the history of a resource monitor
<a href="#">fteCreateMonitor</a>	Create and start a new resource monitor
<a href="#">fteCreateTemplate</a>	Create a transfer template for future use
<a href="#">fteCreateTransfer</a>	Create and start a new file transfer
<a href="#">fteDeleteMonitor</a>	Stop and remove an existing resource monitor
<a href="#">fteDeleteScheduledTransfer</a>	Delete a particular file transfer that you have previously scheduled
<a href="#">fteDeleteTemplates</a>	Delete existing file transfer templates

Table 339. Managed File Transfer commands and their purpose (continued)	
Command name	Purpose
<a href="#">fteListAgents</a>	List all of the agents registered against a particular coordination queue manager
<a href="#">fteListMonitors</a>	List all of the resource monitors registered against a particular coordination queue manager
<a href="#">fteListScheduledTransfers</a>	List all of the Managed File Transfer transfers that you previously created using the command line or the IBM MQ Explorer.
<a href="#">fteListTemplates</a>	List all the file transfer templates for a coordination queue manager
<a href="#">ftePingAgent</a>	Pings an agent to determine whether the agent is active and able to process transfers.
 <a href="#">fteSetProductID</a>	Set z/OS SCRT recording product id
<a href="#">fteShowAgentDetails</a>	Display the details of a particular agent
<a href="#">fteShowLoggerDetails</a>	Display the details of a particular logger
<a href="#">fteStartAgent</a>	Start a particular agent before using it to transfer files
<a href="#">fteStartLogger</a>	Start logger
 <a href="#">fteStartMonitor</a>	Start a resource monitor without needing to stop or restart an agent
<a href="#">fteStopAgent</a>	Stop a particular agent
<a href="#">fteStopLogger</a>	Stop logger
 <a href="#">fteStopMonitor</a>	Stop a resource monitor without needing to stop or restart an agent
<b>Command for security:</b>	
<a href="#">fteObfuscate</a>	Encrypt sensitive data in credentials files.
<b>Commands for troubleshooting:</b>	
<a href="#">fteDisplayVersion</a>	Display the product version
<a href="#">fteSetAgentLogLevel</a>	Enable or disable diagnostic information logging for file transfers between a Managed File Transfer protocol bridge agent and FTP/SFTP/FTPS file servers.
<a href="#">fteSetAgentTraceLevel</a>	Set the level of agent trace to run
<a href="#">fteSetLoggerTraceLevel</a>	Set the level of logger trace to run
<a href="#">fteRAS</a>	Run the RAS gathering tool

See [Installed MFT command sets](#) for a table showing which commands are installed with which Managed File Transfer offering.

The syntax for each command and its parameters is presented in the form of a syntax diagram. For an explanation of how to use these diagrams, see [How to read syntax diagrams](#).

### Related concepts

[“Authority to use MFT commands” on page 1986](#)

[MFT object naming conventions](#)

### Related reference

[“Which MFT commands and processes connect to which queue manager” on page 1979](#)

A Managed File Transfer topology consists of a number of different components.

[“\[z/OS\]fteBatch, fteCommon and ftePlatform helper scripts” on page 1989](#)

The **z/OS** fteBatch, fteCommon and ftePlatform are scripts that are provided by Managed File Transfer in the `MQ_INSTALLATION_PATH/bin` directory as helper scripts. The fteBatch script is present on z/OS only.

## Which MFT commands and processes connect to which queue manager

A Managed File Transfer topology consists of a number of different components.

These components are:

- One or more agents, with their associated agent queue manager
- A coordination queue manager
- A command queue manager
- A number of commands that are used to administer the topology, and submit managed transfers
- An optional logger, which collects information about the managed transfers that are performed by the agents in the topology
- The IBM MQ Explorer Managed File Transfer plugin, which can be used to perform some administrative tasks and view information about managed transfers.

Agents, loggers, commands, and the IBM MQ Explorer Managed File Transfer plugin connect to one or more queue managers when they run.




The following tables summarizes which queue manager agents, loggers, commands, and IBM MQ Explorer Managed File Transfer plugin connect to, when they run.

If there are no X characters for a command or process in the table, the command does not connect to any queue manager or process when it runs.

**Note:** Some commands can only be run on specific machines. For more information, see [“Where you can run MFT commands from”](#) on page 1985.

Command name	Agent queue manager	Command queue manager	Coordination queue manager	Logger queue manager
<a href="#">fteAnt</a>				
<a href="#">fteCancelTransfer</a>		X		
<a href="#">fteChangeDefaultConfigurationOptions</a>				
<a href="#">fteCleanAgent</a> “1” on page 1980	X			
<a href="#">fteClearMonitorHistory</a>		X		
<a href="#">fteCreateAgent</a>	X			
<a href="#">fteCreateBridgeAgent</a>	X			
<a href="#">fteCreateCDAgent</a>	X			
<a href="#">fteCreateEnvironment</a>				
<a href="#">fteCreateLogger</a>				
<a href="#">fteCreateMonitor</a>		X		
<a href="#">fteCreateTemplate</a>			X	
<a href="#">fteCreateTransfer</a>		X		
<a href="#">fteDefine</a>				
<a href="#">fteDelete</a>				
<a href="#">fteDeleteAgent</a>	X		X	
<a href="#">fteDeleteLogger</a>				
<a href="#">fteDeleteMonitor</a>		X		

Table 340. Summary of which Managed File Transfer commands connect to which queue manager (continued)

Command name	Agent queue manager	Command queue manager	Coordination queue manager	Logger queue manager
<a href="#">fteDeleteScheduledTransfer</a>		X		
<a href="#">fteDeleteTemplates</a>			X	
<a href="#">fteDisplayVersion</a>				
<a href="#">fteListAgents</a>			X	
<a href="#">fteListMonitors</a>			X	
<a href="#">fteListScheduledTransfers</a>			X	
<a href="#">fteListTemplates</a>			X	
<a href="#">fteModifyAgent</a>				
<a href="#">fteModifyLogger</a>				
<a href="#">fteObfuscate</a>				
<a href="#">ftePingAgent</a>		X		
<a href="#">fteRAS</a>				
<a href="#">fteSetAgentLogLevel</a>				
<a href="#">fteSetAgentTraceLevel</a>				
<a href="#">fteSetLoggerTraceLevel</a>				
 <a href="#">fteSetProductID</a>				
<a href="#">fteSetupCommands</a>				
<a href="#">fteSetupCoordination</a>				
<a href="#">fteShowAgentDetails</a>			X	
<a href="#">fteShowLoggerDetails</a>				
<a href="#">fteStartAgent</a>				
<a href="#">fteStartLogger</a>				
 <a href="#">fteStartMonitor</a>		X		
<a href="#">fteStopAgent</a>		X		
<a href="#">fteStopLogger</a>		X		
 <a href="#">fteStopMonitor</a>		X		

**Note:**

1. The agent must be stopped before you run this command

Table 341. Summary of which Managed File Transfer processes connect to which queue manager

Processes	Agent queue manager	Command queue manager	Coordination queue manager	Logger queue manager
Managed File Transfer agents	X			
Managed File Transfer plug-in for IBM MQ Explorer		X	X	
Managed File Transfer logger			X	X

The file that contains the credentials information that is required to connect to each type of queue manager, that is, agent, command, and coordination queue managers, can be specified in the associated properties file. For example, the coordination queue manager has a `coordination.properties` file. In

this file, you can set the **coordinationQMgrAuthenticationCredentialsFile** property to point to the credentials file.

The commands that connect to the coordination queue manager use the credentials information that is specified in that file. If security is enabled on a queue manager and this property is incorrectly set, MFT commands do not successfully complete. For more information, see [MFT and IBM MQ connection authentication](#).

### Related concepts

[Installed MFT command sets](#)

### **Details of which MFT commands connect to which queue manager**

Further information on which Managed File Transfer commands connect to which queue manager.

This topic expands on the information in [“Which MFT commands and processes connect to which queue manager”](#) on page 1979, together with some illustrations.

### Agent queue managers

Every agent has its own agent queue manager. The agent uses system queues hosted on this queue manager to maintain state information and receive requests for work.

A single queue manager can act as the agent queue manager for multiple agents. Agent queue managers are connected to the coordination queue manager, the command queue manager, and other agent queue managers using sender and receiver channels.

The **fteCreateAgent**, **fteCreateBridgeAgent**, and **fteCreateCDAgent** commands take the agent queue manager name as an argument.

When these commands are run, they connect to the specified queue manager and send a message to the coordination queue manager indicating that the agent has been added to the Managed File Transfer topology.

Similarly, when **fteDeleteAgent** is run, it connects to the agent queue manager and sends a message to the coordination queue manager, informing it that the agent has now been removed from the Managed File Transfer topology.

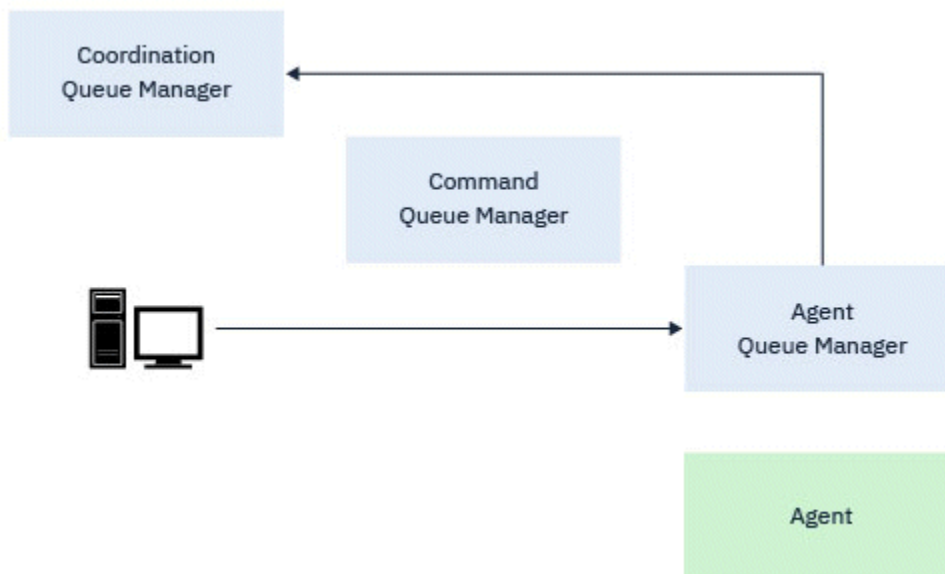


Figure 1. The various **fteCreate** commands, along with the **fteDeleteAgent** command, connect to the agent queue manager and send a message to the coordination queue manager to either register or deregister an agent from the topology.



**Attention: `fteCleanAgent`** connects to the agent queue manager, and removes any state information for that agent from its system queues.

Running this command could have an impact on the whole topology. As such, you should only run this command under guidance from IBM.

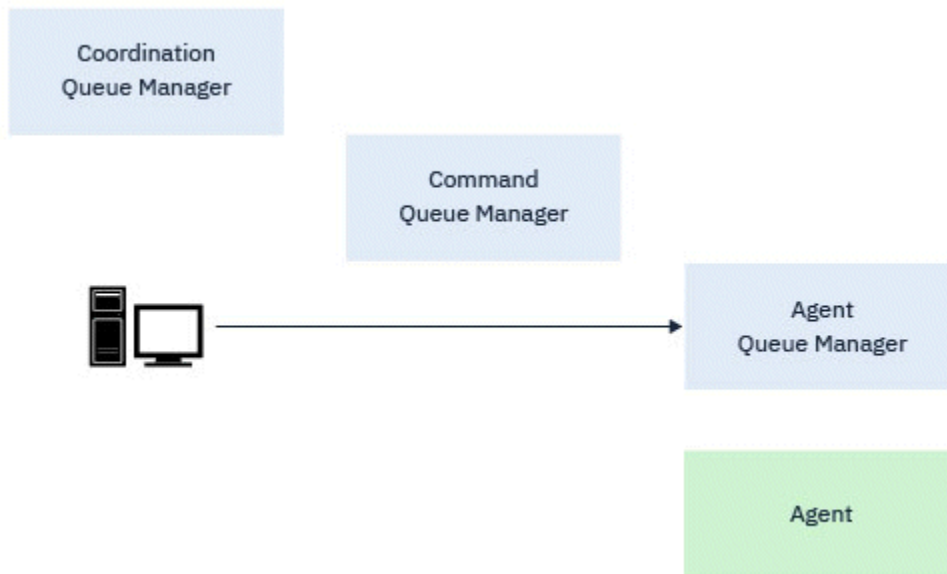


Figure 2. The **`fteCleanAgent`** command connects directly to the agent queue manager, and removes state information from the system queues of the agent

The following commands connect to the agent queue manager:

- [`fteCleanAgent`](#)
- [`fteCreateAgent`](#)
- [`fteCreateBridgeAgent`](#)
- [`fteCreateCDAgent`](#)
- [`fteDeleteAgent`](#)

## Coordination queue managers

The coordination queue manager for a Managed File Transfer topology is a central hub that has knowledge of the entire topology. The coordination queue manager is connected to all of the agent queue managers in a topology through sender and receiver channels. Agents regularly publish status information to the coordination queue manager, and store their transfer templates there.

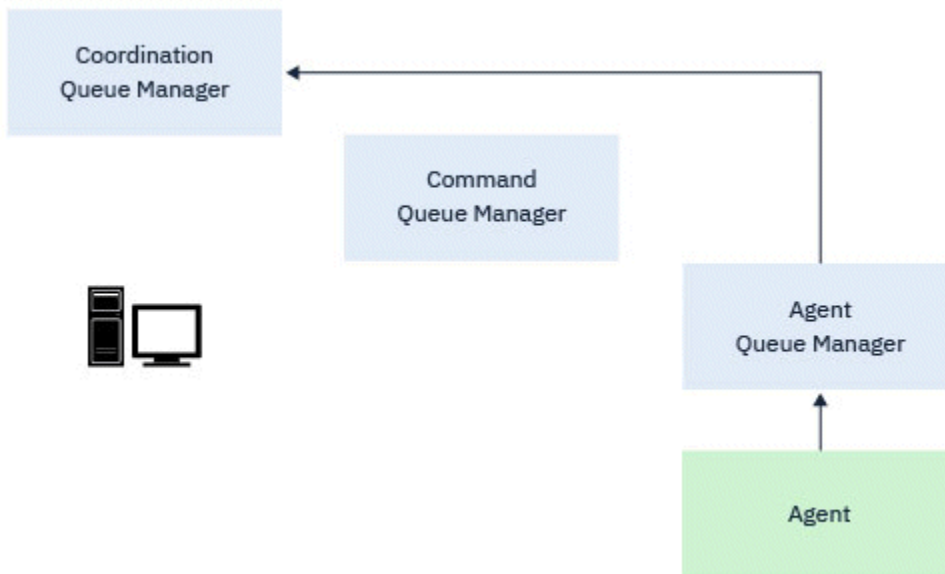


Figure 3. Agents publish status information or store transfer templates on the coordination queue manager

When any of the preceding commands that connect to the coordination queue manager are run, they connect directly to the coordination queue manager and either:

- Create or delete a transfer template.
- Query state information about agents, monitors, or scheduled transfers, and display that information to the user.

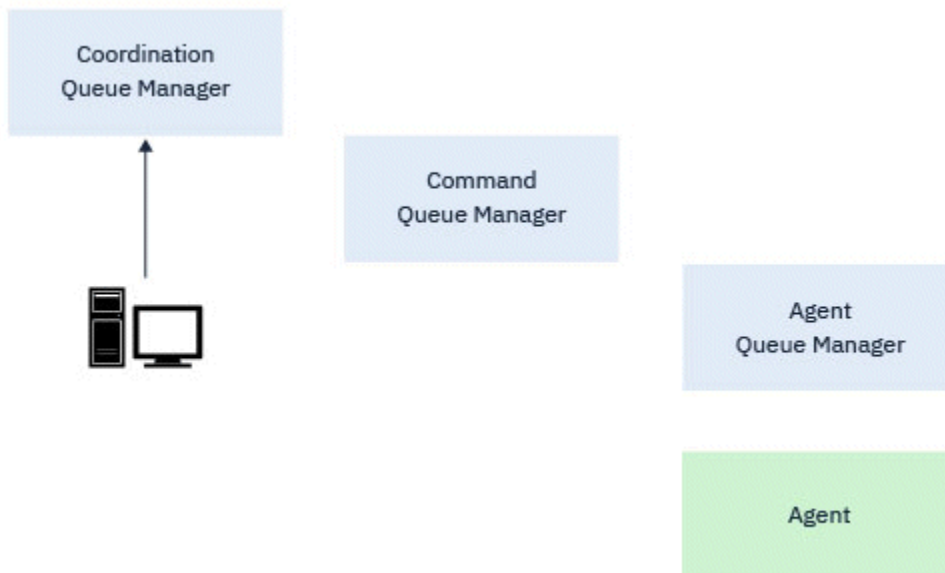


Figure 4. Commands then connect directly to the coordination queue manager to either obtain the appropriate status information or work with transfer templates.

The following commands connect to the coordination queue manager:

- [fteCreateTemplate](#)
- [fteDeleteAgent](#)
- [fteDeleteTemplates](#)

- [fteListAgents](#)
- [fteListMonitors](#)
- [fteListScheduledTransfers](#)
- [fteListTemplates](#)
- [fteShowAgentDetails](#)

## Command queue managers

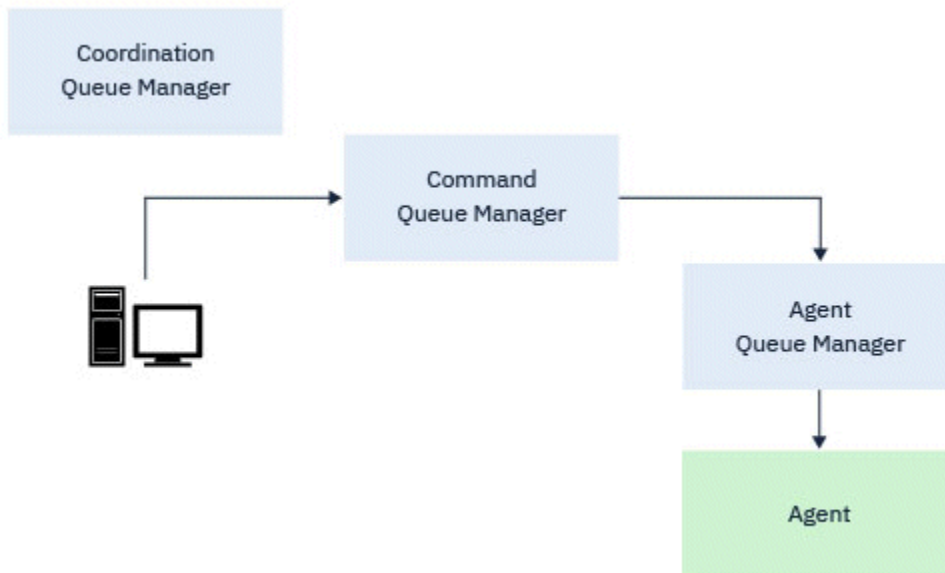
The command queue manager acts as a gateway into an MFT topology. It is connected to the agent queue managers through sender and receiver channels. When one of the commands listed is run, they connect directly to the command queue manager, and send a message to the specified agent. This message is routed through the IBM MQ network to the agent queue manager, where it is picked up by the agent and processed.

When any of the commands that connect to the queue manager are run, they:

- Connect to the command queue manager.
- Create a temporary reply queue.
- Send a message containing the command details to the appropriate agent.

The message is routed through the IBM MQ network to the agent queue manager, where it is picked up by the agent and processed.

After the agent has processed the command, the agent sends a reply back to the command queue manager, where the reply is picked up by the command.



*Figure 5. The commands connect to the command queue manager. The message containing the command is then routed through the IBM MQ network to the correct agent queue manager, where it is picked up by the agent.*

The following commands connect to the command queue manager:

- [fteCancelTransfer](#)
- [fteCreateMonitor](#)
- [fteCreateTransfer](#)
- [fteDeleteMonitor](#)
- [fteDeleteScheduledTransfer](#)



- [ftePingAgent](#)
- [V 9.3.0 fteStartMonitor](#)
- [fteStopAgent](#)
- [V 9.3.0 fteStopMonitor](#)

### Related concepts

[Installed MFT command sets](#)

### Where you can run MFT commands from

The following table summarize whether Managed File Transfer commands can be run locally or remotely.

Where the command is shown as:

- Local only - the commands can only run on the local installation.
- Local to agent - the commands can only run on the local agent installation relevant to that command.




For example, the **fteCreateBridgeAgent** command can connect to the local bridge agent, shown as *X (Bridge)* and the **fteCreateCDAgent** command can connect to the local CD agent, shown as *X (CD)*.

- Local to logger agent- the commands can only run on the local logger installation.
- Local and remote - you can run the command from any system that can connect to the command or coordination queue manager.

If there are no X characters for a command in the table, the command cannot be run in that situation.

Command name	Local	Local to agent	Local to logger agent	Local and remote
<a href="#">fteCancelTransfer</a>				X
<a href="#">fteCleanAgent</a>				X
<a href="#">fteCreateAgent</a>		X		
<a href="#">fteCreateBridgeAgent</a>		X (Bridge)		
<a href="#">fteCreateCDAgent</a>		X (CD)		
<a href="#">fteCreateEnvironment</a>		X		
<a href="#">fteCreateLogger</a>			X	
<a href="#">fteDefine</a>		X		
<a href="#">fteDelete</a>		X		
<a href="#">fteDeleteAgent</a>		X		
<a href="#">fteDeleteLogger</a>			X	
<a href="#">fteDeleteMonitor</a>				X
<a href="#">fteDeleteScheduledTransfer</a>				X
<a href="#">fteDeleteTemplates</a>				X
<a href="#">fteDisplayVersion</a>	X			
<a href="#">fteListAgents</a>				X
<a href="#">fteListMonitors</a>				X
<a href="#">fteListScheduledTransfers</a>				X
<a href="#">fteListTemplates</a>				X
<a href="#">fteModifyAgent</a>		X		
<a href="#">fteModifyLogger</a>			X	

Table 342. Summary of where Managed File Transfer commands can be run from (continued)


Command name	Local	Local to agent	Local to logger agent	Local and remote
<a href="#">fteObfuscate</a>	X			
<a href="#">ftePingAgent</a>				X
<a href="#">fteRAS</a>	X			
<a href="#">fteSetAgentLogLevel</a>		X		
<a href="#">fteSetAgentTraceLevel</a>		X		
<a href="#">fteSetLoggerTraceLevel</a>			X	
 <a href="#">fteSetProductID</a>	X			
<a href="#">fteSetupCommands</a>	X			
<a href="#">fteSetupCoordination</a>	X			
<a href="#">fteShowAgentDetails</a>				X
<a href="#">fteShowLoggerDetails</a>				X
<a href="#">fteStartAgent</a>		X		
<a href="#">fteStartLogger</a>			X	
 <a href="#">fteStartMonitor</a>				X
<a href="#">fteStopAgent</a>		X		
<a href="#">fteStopLogger</a>			X	
 <a href="#">fteStopMonitor</a>				X


### Related concepts

[Installed MFT command sets](#)

## Authority to use MFT commands

Your user ID must be a member of the mqm group if you want to issue Managed File Transfer commands, unless you have already configured IBM MQ to allow users who are not in the mqm group to issue commands.

 For more information about defining an alternative group to mqm on z/OS, see [Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS](#).

For more information about authorization, see [Authority to administer IBM MQ](#).  If you are using IBM i, start with the following topic: [IBM MQ authorities](#).

A subset of the Managed File Transfer commands can be issued using the IBM MQ Explorer.

## Issuing commands from AIX, Linux, and Windows systems

Note the following environment-specific information for issuing commands:

### Managed File Transfer for Windows

All commands can be issued from a command line. Command names are not case-sensitive: You can enter them in uppercase, lowercase, or a combination of uppercase and lowercase. However, arguments to control commands (such as queue names) and parameters (such as **-m** for queue manager name) are case-sensitive.

In the syntax descriptions, the hyphen (-) is used as a flag indicator.

### Managed File Transfer for AIX and Linux systems

All Managed File Transfer commands can be issued from a shell. All commands are case-sensitive.

## Issuing commands from z/OS systems



From IBM MQ for z/OS 9.2, Managed File Transfer is installed in to the `mqft` directory of the z/OS UNIX System Services (z/OS UNIX) Components; for example: `/mqm/V9R2M0/mqft`.

The Managed File Transfer commands are in the `bin` directory underneath the `mqft` directory; for example: `/mqm/V9R2M0/mqft/bin`.

**Important:** This is different from earlier releases, where the `bin` and `mqft` directories were peers of each other.

The commands can be run from either of the following options:

- Directly from the z/OS UNIX environment by specifying the path to the command or including the `bin` subdirectory in the user command path.
- From a PDSE data set of commands configured from the PDSE command template library, for a particular agent or logger. For more information, see [Creating an MFT Agent or Logger command data set](#).

## Issuing commands from the IBM i platform



Note the following environment-specific information for issuing commands on IBM i:

- You can start Managed File Transfer commands using the Qshell interpreter. To start the Qshell interpreter, issue the **STRQSH** command from an IBM i system command line.
- When you run commands in the Qshell environment, command names are not case-sensitive: You can enter them in uppercase, lowercase, or a combination of uppercase and lowercase. However, arguments to control commands (such as queue names) and parameters (such as **-m** for queue manager name) are case-sensitive.

### Related reference

[Return codes for MFT](#)

## Tracing MFT commands

You can trace any of the Managed File Transfer commands to help with problem determination from the command line.

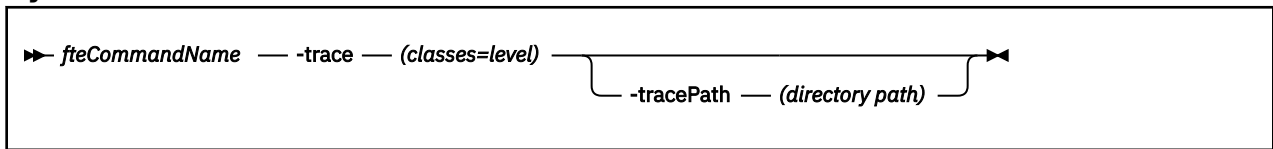
### Purpose

Use the **-trace** parameter for any MFT command to enable trace at a specified level. The trace files produced are located in your current working directory unless the **-tracePath** parameter is included to identify a different directory.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the `agent.properties` file. These properties are described in [Advanced agent properties: Tracing and logging](#).

### Syntax



## Parameters

### **-trace classes=level**

Required. Level to set the trace and which classes to apply the trace to. Specify the following format:

```
classes=level
```

For example:

```
com.ibm.wmqfte=all
```

which traces all Managed File Transfer classes.

Specify a colon-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes.

If *classes* start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

#### **off**

Switches the agent trace off but continues to write information to the log files. This is the default option.

#### **flow**

Captures data for trace points associated with processing flow in the agent.

#### **moderate**

Captures a moderate amount of diagnostic information in the trace.

#### **verbose**


Captures a verbose amount of diagnostic information in the trace.


#### **all**

Sets agent trace to run on all agent classes.

### **-tracePath directory path**

Optional. Specify the directory that you want the trace to be written to. For example, c:\temp.

 If you do not specify this parameter, the value is the directory that the command was issued from. For example, on z/OS:

```
 /u/smith/fte/wmqmft/mqft/logs/MQPV/loggers/BFGLG1/logs/
```

This parameter is valid only when the **-trace** parameter is specified.

## Example

In this example the trace level is set to all, meaning that all of the classes belonging to AGENT.NAME are traced for the **fteStartAgent** command:

**Note:** When the agent is started, the trace goes to *mft\_config/logscoordination\_qmgr/agents/agent*

```
fteStartAgent -trace com.ibm.wmqfte=all -tracePath /u/mft/trace AGENT.NAME
```


In this example the trace level is set to moderate for the com.ibm.wmqfte.common classes for the agent AGENT.NAME. A moderate amount of trace is captured for the **ftePingAgent** command:

```
ftePingAgent -trace com.ibm.wmqfte.common=moderate AGENT.NAME
```

In this example the trace level is set to moderate for the `com.ibm.wmqfte.common` classes for the agent `AGENT.NAME`, and the trace is written to the `c:\$user` directory. A moderate amount of trace is captured for the **ftePingAgent** command:

```
ftePingAgent -trace com.ibm.wmqfte.common=moderate -tracePath c:\$user AGENT.NAME
```


## **fteBatch, fteCommon and ftePlatform helper scripts**

The  `fteBatch`, `fteCommon` and `ftePlatform` are scripts that are provided by Managed File Transfer in the `MQ_INSTALLATION_PATH/bin` directory as helper scripts. The `fteBatch` script is present on z/OS only.

### **fteBatch script (z/OS only)**

#### **z/OS**

`fteBatch` is a helper script for running Managed File Transfer from the JZOS Batch Launcher. `fteBatch` is installed on z/OS only. Typically Managed File Transfer is started using the supplied command shell scripts, which perform some environment configuration before starting the Java class appropriate to that function. When Managed File Transfer is started using the JZOS Batch Launcher, the Java class is started directly from the Launcher. `fteBatch` can be called as part of the launcher setup to place the required class name into an environment variable and performs the setup work that the normal command shell scripts perform before starting Java. This provides a level of isolation between your jobs and the internal class names used by Managed File Transfer.

 **Deprecated** The `fteBatch` command is deprecated for Managed File Transfer in IBM MQ 8.0, as you can run Managed File Transfer through the new PDSE data set of commands. For more information, see [Creating an MFT Agent or Logger command data set](#).

### **fteCommon**

`fteCommon` is a helper script started by the other Managed File Transfer command scripts to perform common setup processing before starting Java.

### **ftePlatform**

`ftePlatform` is a helper script started by the `fteCommon` script to perform platform-specific setup processing.

### **fteCancelTransfer (cancel an MFT transfer)**

Use the **fteCancelTransfer** command to cancel a Managed File Transfer transfer. You can issue this command against either the source or destination agent for the transfer.

### **Purpose**

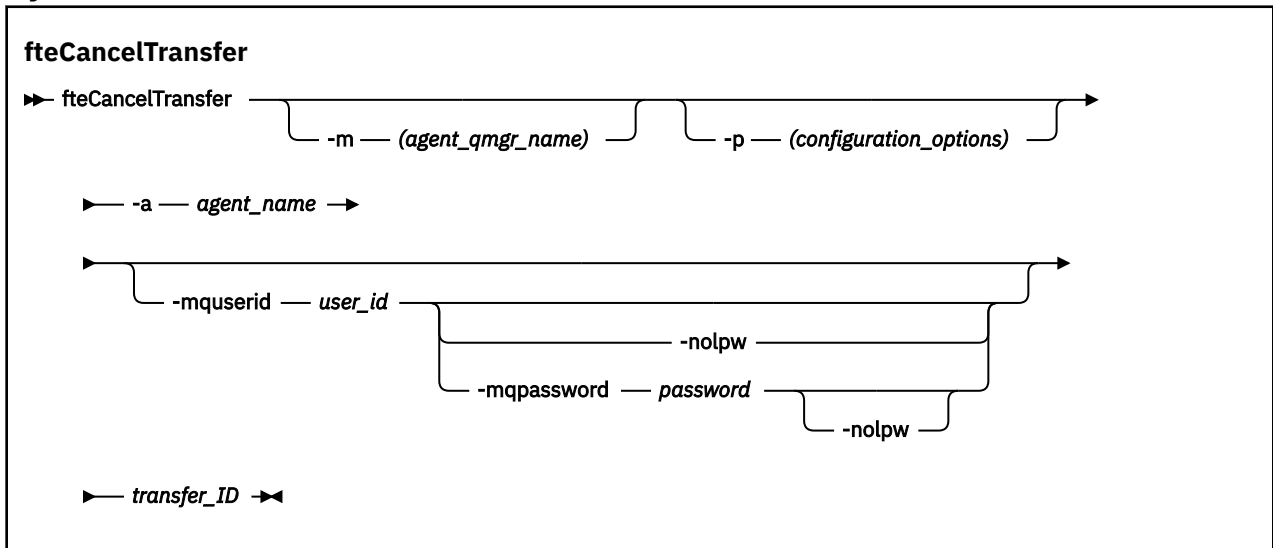
If you issue the **fteCancelTransfer** command while that transfer is currently in progress, any files already transferred as part of that transfer remain on the destination system and are not deleted. Any files partially transferred as part of that transfer are deleted from the destination system. The destination side of the transfer logs that transfer as "cancelled".

If a transfer to a Connect:Direct node is canceled, any files partially transferred as part of the canceled transfer remain on the destination system and are not deleted.

You can run the **fteCancelTransfer** command from any system that can connect to the IBM MQ network and then route to the agent queue manager. Specifically for the command to run, you must have installed Managed File Transfer on this system and you must have configured Managed File Transfer on this system to communicate with the IBM MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

## Syntax



## Parameters

### **-m agent\_qmgr\_name**

Optional. The name of the agent queue manager. This agent must be either the source or destination agent for the transfer you want to cancel. If you do not specify this parameter, the cancel request is sent to the queue manager identified by the set of configuration options you are using.

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### **-a agent\_name**

Required. The name of either the source or destination agent of the transfer that you want to cancel.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using [MQCSP](#) authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-no1pw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### **transfer\_ID**

Required. The ID of the transfer you want to cancel. The transfer ID (also known as the request ID) is displayed at the command line after you issue the **fteCreateTransfer** command. Transfer IDs are also included in file transfer log messages or are displayed in the IBM MQ Explorer Transfer Log panel.

### **-? or -h**

Optional. Displays command syntax.

### **Example**

In this example AGENT1 is the source agent for the transfer to be canceled.

```
fteCancelTransfer -a AGENT1 414d5120514d5f4c4d343336303920201159c54820027102
```

### **Return codes**

#### **0**

Either the command completed successfully or the specified transfer ID is unknown to the agent. If the transfer ID is unknown to the agent, the most likely reason is that the transfer has already completed or has been canceled.

#### **1**

Command ended unsuccessfully.


### **Related reference**


[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

## **fteChangeDefaultConfigurationOptions (change default configuration options for MFT)**

Use the **fteChangeDefaultConfigurationOptions** command to change the default configuration options that you want Managed File Transfer to use. The value of the configuration options defines the group of properties files that Managed File Transfer uses.

**Important:**  On IBM MQ for AIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the command:

- Be a member of the mqm group (if the mqm group is defined on the system).
- Be a member of the group named in the BFG\_GROUP\_NAME environment variable (if one is named).
- Have no value set in the BFG\_GROUP\_NAME environment variable when the command is run.

### **Purpose**

Your default Managed File Transfer configuration options are established the first time you use the [fteSetupCoordination](#) command to configure a queue manager as the coordination queue manager. During the installation of the MFT product, the mqft directory is created under <MQ\_DATA\_PATH> if it does not already exist. Additionally, configuration, installations, and logs directories are created under the mqft directory, if they do not already exist.

By using the **fteChangeDefaultConfigurationOptions** command you can change the default coordination queue manager that is defined in the `installation.properties` file. If you change this coordination queue manager, Managed File Transfer uses the configuration options given by the structured set of directories and property files contained in the directory you used as input for `configuration_options` by default. This directory name is the same as the coordination queue manager used by agents under this configuration.

See [Configuration options](#) for more information about the `installation.properties` file.

## Syntax

### **fteChangeDefaultConfigurationOptions**

```
►► fteChangeDefaultConfigurationOptions — configuration_options ◄◄
```

## Parameters

### **configuration\_options**

Required. This parameter specifies the default configuration options that you want to change to. Use the name of a non-default coordination queue manager as the input for this parameter.

### **-? or -h**

Optional. Displays command syntax.

## Example

In this example, the default configuration options are changed to QM\_COORD2:

```
fteChangeDefaultConfigurationOptions QM_COORD2
```

## Return codes

### **0**

Command completed successfully.

### **1**

Command ended unsuccessfully.

## Related concepts

[Configuration options](#)

## **fteCleanAgent (clean up an MFT Agent)**

Use the **fteCleanAgent** command to clean up the queues that a Managed File Transfer Agent uses, by deleting messages from the persistent and non-persistent queues used by the agent. Use the **fteCleanAgent** command if you are having problems starting an agent, which might be caused by information remaining on the queues used by the agent.

## Purpose

Use the **fteCleanAgent** command to delete messages from the persistent and non-persistent queues used by the agent. Specifically, this command can carry out the following actions:

- Remove any transfers that were in progress to this agent or from this agent before the transfer was stopped. These transfers are not resumed when the agent restarts
- Remove any commands that have already been submitted to the agent, but have not yet been carried out
- Delete all resource monitors stored on the agent



- Delete all scheduled transfers stored on the agent
- Delete all invalid messages stored on the agent

If the agent is a Connect:Direct bridge agent, the **-ms**, **-ss**, and **-ims** parameters are not valid. For Connect:Direct bridge agents the command also carries out the following actions:

- Deletes all files from the directory where the Connect:Direct bridge agent temporarily stores files while they are being transferred. The location of this directory is defined by the **cdTmpDir** parameter
- Displays information about the Connect:Direct processes that are associated with any ongoing transfers

You must, by default, specify which Managed File Transfer state to clear by passing the appropriate parameters to the **fteCleanAgent** command, as well as providing an agent name. This means that, by default, **fteCleanAgent** does not clear all in-progress and pending transfers, resource monitor definitions and scheduled transfer definitions for the agent specified. You can enable or disable this behavior by setting the **failCleanAgentWithNoArguments** property in the `command.properties` file to the appropriate value:

- By default, the value of **failCleanAgentWithNoArguments** is `true`, which means that the **fteCleanAgent** command fails to run if only the **agent\_name** parameter is specified.
- If **failCleanAgentWithNoArguments** is set to `false` and only the **agent\_name** parameter is specified, **fteCleanAgent** behaves in the same way as it does when you specify the **-all** parameter.

You must run the **fteCleanAgent** command on an agent that has been stopped. If you try to run the command on an agent that is currently running, you receive an error. This command does not start the agent. The **fteCleanAgent** command cleans up an agent on the system where you issue the command. You cannot clean up an agent on a remote system. To run the **fteCleanAgent** command you must have write access to the agent lock file, which is located at `MQ_DATA_PATH\mqft\logs\coordination_QMgr_name\agents\agent_name\agent.lock`

The FTEAGENT group must have GET and BROWSE authority on the following queues to run **fteCleanAgent** successfully:

- `SYSTEM.FTE.COMMAND.agent_name`
- `SYSTEM.FTE.EVENT.agent_name`
- `SYSTEM.FTE.STATE.agent_name`

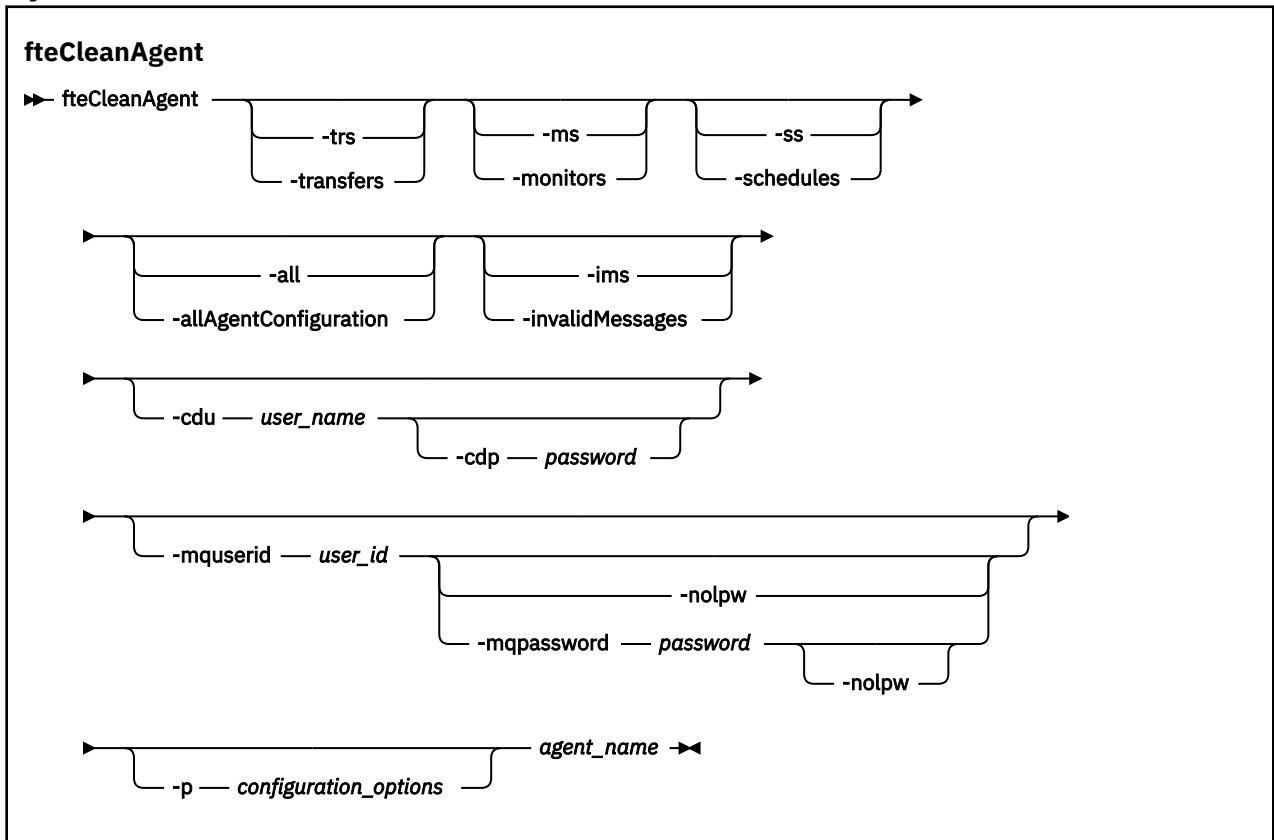
See [Restricting group authorities for MFT-specific resources](#) for further information about the FTEAGENT group and restricting group authorities.

If you are running the **fteCleanAgent** command on an agent that is connected to its queue manager in bindings mode, and the agent has recently stopped running, the **fteCleanAgent** command might report messaging problem: MQRC 2042. This MQRC occurs because a queue handle for the agent still exists in the queue manager. After a short delay the queue manager removes this handle, and you can reissue **fteCleanAgent**.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

**Note:** When cleaning a Connect:Direct bridge agent, the user ID used to run the **fteCleanAgent** command must have read and write access to the Connect:Direct bridge agent temporary directory.

## Syntax



## Parameters

You can use the **fteCleanAgent** command to delete specific artifacts. For example, you can specify the **-trs** command to delete pending transfers but not change any resource monitors and scheduled transfers.

### **-trs or -transfers**

Optional. Specifies that in-progress and pending transfers are to be deleted from the agent. You cannot specify this parameter with **-all** or **-ims** parameters.

### **-ms or -monitors**

Optional. Specifies that all resource monitor definitions are to be deleted from the agent. You cannot specify this parameter with **-all** or **-ims** parameters.

### **-ss or -schedules**

Optional. Specifies that all scheduled transfer definitions are to be deleted from the agent. You cannot specify this parameter with the **-all** or **-ims** parameters.

### **-all or -allAgentConfiguration**

Optional. Specifies that all transfers, resource monitor definitions, and scheduled transfer definitions are to be deleted from the agent. You cannot specify this parameter with the **-trs**, **-ss**, **-ms**, or **-ims** parameters.



**Attention:** You should use the **all** parameter only if no other options are available. The action of deleting transfers, resource monitor definitions, and scheduled transfer definitions can have a significant impact on your enterprise.

### **-ims or -invalidMessages**

Optional. Specifies that all invalid messages are to be deleted from the agent. You cannot specify this parameter with the **-trs**, **-ss**, **-ms**, or **-all** parameters.

### **-cdu user\_name**

Optional. Only valid if the agent being cleaned is a Connect:Direct bridge agent. If this parameter is specified, the command uses the user name provided to make a connection to the Connect:Direct bridge node and retrieve additional information about existing Connect:Direct processes. If you do not specify this parameter, the agent is cleaned but information about Connect:Direct processes is not displayed.

### **-cdp password**

Optional. Valid only if the agent being cleaned is a Connect:Direct bridge agent and you have specified the **-cdu** parameter. If you specify the **-cdp** parameter, the command uses the password provided to make a connection to the Connect:Direct bridge node and retrieve additional information about existing Connect:Direct processes. If you do not specify this parameter, and the **-cdu** parameter has been specified, you are asked to provide the password interactively.


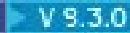
### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the agent queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

  From IBM MQ 9.3 onwards, the command defaults to using [MQCSP](#) authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to clean up an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### **agent\_name**

Required. The name of the Managed File Transfer agent that you want to clean up.

### **-? or -h**

Optional. Displays command syntax.

## **Examples**

In this basic example, all the queues used by AGENT2 are cleaned up:

```
C:\Documents and Settings\Administrator>fteCleanAgent -all AGENT2
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
```

```
All messages will be deleted from all queues
```

```
State Queue Entries:
```

```
Transfer Identifier: 414d5120716d31202020202020202020202020202020202786de4d20485b03
```

```
Source Agent Name:      AGENT2
Destination Agent Name: AGENT3

Transfer Identifier:    414d5120716d31202020202020202020202020202786de4d20487203
Source Agent Name:     AGENT2
Destination Agent Name: AGENT3

Command Queue New Transfer Entries:

Scheduler Queue Schedule Entries:

Directory Monitor Configuration for "MONITOR1" has been cleared from the Agent.

Schedule Identifier:   1
Source Agent Name:     AGENT2
Destination Agent Name: AGENT3

BFGCL0149I: The agent 'AGENT2' has been cleaned.
```

In this example, the invalid messages queue used by AGENT2 are cleaned up:

```
C:\Documents and Settings\Administrator>fteCleanAgent -ims AGENT2
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED

Invalid messages will be deleted from all queues

State Queue Entries:

Warning - Invalid message found on the queue

Command Queue New Transfer Entries:

Warning - Invalid message found on the queue

Scheduler Queue Schedule Entries:

Warning - Invalid message found on the queue

BFGCL0149I: The agent 'AGENT2' has been cleaned.
```

In this example, the transfers queue used by the Connect:Direct bridge agent, AGENT\_CD\_BRIDGE, is cleaned up:

```
C:\Documents and Settings\Administrator>fteCleanAgent -trs -cdu USER1 AGENT_CD_BRIDGE
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Enter Connect:Direct password:

All messages will be deleted from the state and command queues

State Queue Entries:

Transfer Identifier:    414d5120716d31202020202020202020202020202786de4d2048a703
Source Agent Name:     AGENT2
Destination Agent Name: AGENT_CD_BRIDGE
Connect:Direct PNODE Name: CDNODE1
Connect:Direct SNODE Name: CDNODE2
Connect:Direct Current Processes: Name=FA34F8, Number=139

Command Queue New Transfer Entries:

BFGCL0149I: The agent 'AGENT_CD_BRIDGE' has been cleaned.
```

## Return codes

- 0 Command completed successfully.

## 1

Command ended unsuccessfully.

### Related reference

[“fteStopAgent \(stop an MFT agent\)” on page 2146](#)

Use the **fteStopAgent** command to either stop a Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

[“fteDeleteAgent \(delete an MFT agent and its configuration\)” on page 2073](#)

The **fteDeleteAgent** command deletes a Managed File Transfer Agent and its configuration. If the agent is a protocol bridge agent, the user credentials file is left on the file system.

The MFT [command.properties](#) file

## fteClearMonitorHistory (clear resource monitor history)

Use the **fteClearMonitorHistory** command to clear the history of a resource monitor.

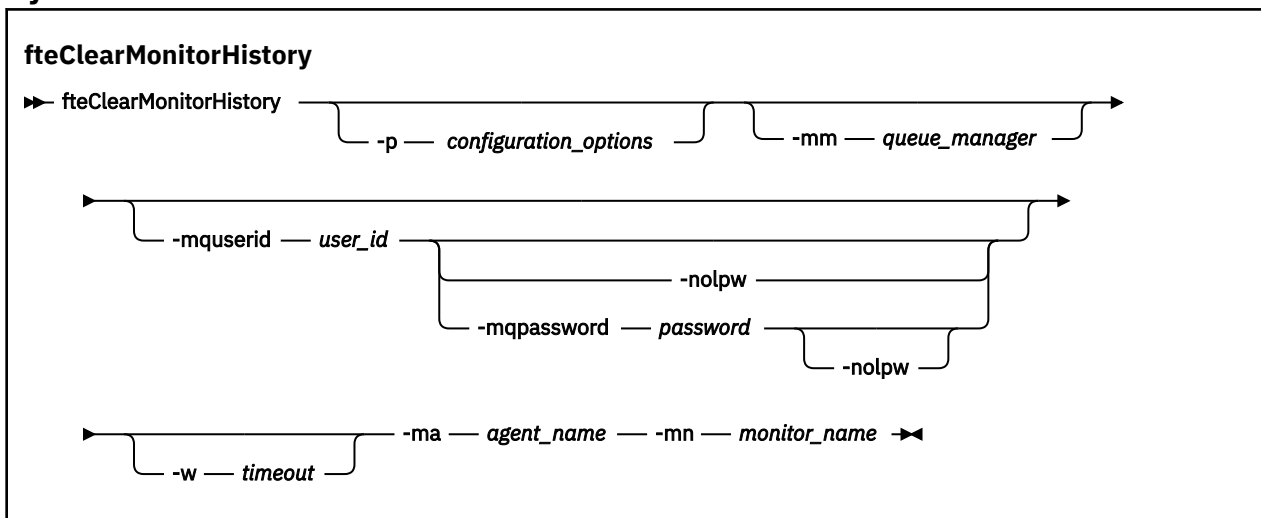
### Purpose

The **fteClearMonitorHistory** command can be run from any system where the MFT commands component is installed. This allows you to clear the history from anywhere, rather than being restricted to the system where the agent that owns the resource monitor is running.

Running the **fteClearMonitorHistory** command puts a Clear Monitor History request XML message on the agent's command queue and wait for a reply on a temporary reply queue. The agent completes the following actions:

- Processes the request message.
- Stops the specified resource monitor
- Clears the history of the specified resource monitor.
- Starts the specified resource monitor.

### Syntax



### Parameters

#### **-ma agent\_name**

Required. The name of the agent running the monitor operation.

#### **-mm queue\_manager**

Optional. The name of the queue manager the agent is connected to.

**-mn *monitor\_name***

Required. The name of the monitor whose history is to be cleared. The characters '\*', '%', and '?' are not allowed in monitor names.

**-p *configuration\_options***

Optional. Determines the set of configuration options that is used to clear the history of the monitor. Use the name of a set of configuration options as the value for the **-p** parameter.

By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

**-w *timeout***

Optional. Specifies to wait for up to *timeout* seconds for the monitor to respond. If you do not specify a timeout, or specify a timeout value of minus one, then the command waits forever for the monitor to respond. If you do not specify this option then the default is to wait up to five seconds for the monitor to respond.



**-mquserid *user\_id***

Optional. Specifies the user ID to authenticate with the command queue manager.

**-mqpassword *password***

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

  From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

  **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**Example**

The following example clears the history of the resource monitor JBSWIFT running in the agent JBAGENT:

```
fteClearMonitorHistory -ma JBAGENT -mn JBSWIFT
```

If the history is cleared successfully, the **fteClearMonitorHistory** command outputs the following messages:

```
BFGCL0780I: A request to clear history of resource monitor 'JBSWIFT' of agent 'JBAGENT' has been issued.
BFGCL0251I: The request has successfully completed.
```

If there is no response from the monitor within the specified timeout period, the **fteClearMonitorHistory** command outputs the following messages:

```
BFGCL0780I: A request to clear history of resource monitor 'JBSWIFT' of agent 'JBAGENT' has been issued.
BFGCL0253W: No acknowledgement to command from agent within timeout.
```

If authority checking is enabled but the user running the **fteClearMonitorHistory** command does not have authority to clear the history (see [Clearing resource monitor history](#)), the command outputs the following messages:

BFGCL0780I: A request to clear history of resource monitor 'JBSWIFT' of agent 'JBAGENT' has been issued.  
BFGCL0267E: This user is not authorized to perform the operation.

## Resource monitor log

The outcome of running the **fteClearMonitorHistory** command is logged in the resource monitor log `resmoneventN.log`, where *N* stands for a number. Here are example log entries:

```
[07/01/2019 16:08:31:144 IST]00000026 F2FM2 Monitor Stopped Resource Monitor Stopped
[07/01/2019 16:08:31:176 IST]00000026 F2FM2 History Cleared Monitor History has been
cleared
[07/01/2019 16:08:31:176 IST]00000026 F2FM2 Monitor Started Resource Monitor Started
```

## Agent event log

The outcome of running the **fteClearMonitorHistory** command is also logged in the agent's `output0.log`, as shown in the following examples.

The **fteClearMonitorHistory** command successfully cleared the resource monitor history:

BFGDM0123I: History of resource of monitor 'JBSWIFT' has been cleared as requested by user 'tjwatson' on host '*hostname*'.

The resource monitor history is empty when the **fteClearMonitorHistory** command is run:

BFGDM0126I: Resource monitor 'JBSWIFT' does not have any items in its history. The request to clear history was submitted by user 'jbusr' on host '*hostname*'.

The **fteClearMonitorHistory** command is issued by the same user who created the monitor but this user does not have the required authority to clear the history (see [Clearing resource monitor history](#)):

BFGDM0124E: User 'jbusr' has requested to clear the history of resource monitor 'JBSWIFT' but does not have either 'Monitor Operations' or 'MONITOR' authorities required to perform this operation.


The **fteClearMonitorHistory** command is issued by a different user from the one who created the resource monitor but this user does not have the Monitor Operations authority to clear the history (see [Clearing resource monitor history](#)).


BFGDM0125E: User 'loggerusr' has requested to clear the history of resource monitor 'JBSWIFT' that belongs to user 'jbusr' but does not have the required authority 'Monitor Operations' to perform this operation.

## fteCreateAgent (create an MFT agent)

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

You can control access to the agent. See [Restricting user authorities on MFT agent actions](#) for further information. You need to use the **-ac** parameter, and give permissions to access some queues.

**Important:**  On IBM MQ for AIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the `mqm` group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the command:

- Be a member of the `mqm` group (if the `mqm` group is defined on the system).
- Be a member of the group named in the `BFG_GROUP_NAME` environment variable (if one is named).
- Have no value set in the `BFG_GROUP_NAME` environment variable when the command is run.

## Purpose

Use the **fteCreateAgent** command to create an agent. This command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.agent\_name
- SYSTEM.FTE.AUTHAGT1.agent\_name
- SYSTEM.FTE.AUTHMON1.agent\_name
- SYSTEM.FTE.AUTHOPS1.agent\_name
- SYSTEM.FTE.AUTHSCH1.agent\_name
- SYSTEM.FTE.AUTHTRN1.agent\_name
- SYSTEM.FTE.COMMAND.agent\_name
- SYSTEM.FTE.DATA.agent\_name
- SYSTEM.FTE.EVENT.agent\_name
- SYSTEM.FTE.REPLY.agent\_name
- SYSTEM.FTE.STATE.agent\_name
- SYSTEM.FTE.HA.agent\_name

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location:

`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_create.mqsc.`

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues used by the agent. The MQSC commands are in a file in the following location:

`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_delete.mqsc.`

Managed File Transfer provides advanced agent properties that help you configure agents. These properties are described in [The agent.properties file](#).

You might need to create a `MQMFTCcredentials.xml` credentials file in order to work with your agent. A sample of this file is located in `MQ_INSTALLATION_PATH/mqft/samples/credentials/`. For more information and examples, see [“MFT credentials file format” on page 2697](#).

### Important:

On AIX and Linux Managed File Transfer commands use socket files to communicate with the agent process running on the same host machine.

These socket files are created in the log directory of the agent and are deleted when an agent stops. In the IBM MQ Managed File Transfer installation, this socket file is created with a file path of: `<MQ_DATA_PATH>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>` where `MQ_DATA_PATH` is `/var/mqm` by default.

For a re-distributable agent, this socket file is created under the directory: `<RE_DISTRIBUTABLE_DIRECTORY>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>`.

For example, if the agent name is SRCAGENT, the agent queue manager name is SRCAGENTQM, the coordination queue manager name is COORDQM, and the redistributable agent is running from the directory `/home/myuser/mqmft-redis`, the full path of this socket file is: `/home/myuser/mqmft-redis/mqft/logs/COORDQM/agents/SRCAGENT/logs/SRCAGENT@SRCAGENTQM`

which is a total file path length of 85 characters.

The maximum path length allowed by these operating systems for a socket file is 107 characters. Therefore, when creating an agent, take care to ensure that the socket file path does not exceed 107



characters. This is particularly important with a redistributable agent where the log directory of the agent can be located in an arbitrary directory location. See the **fteCreateEnvironment** command for details on setting up the configuration directory.

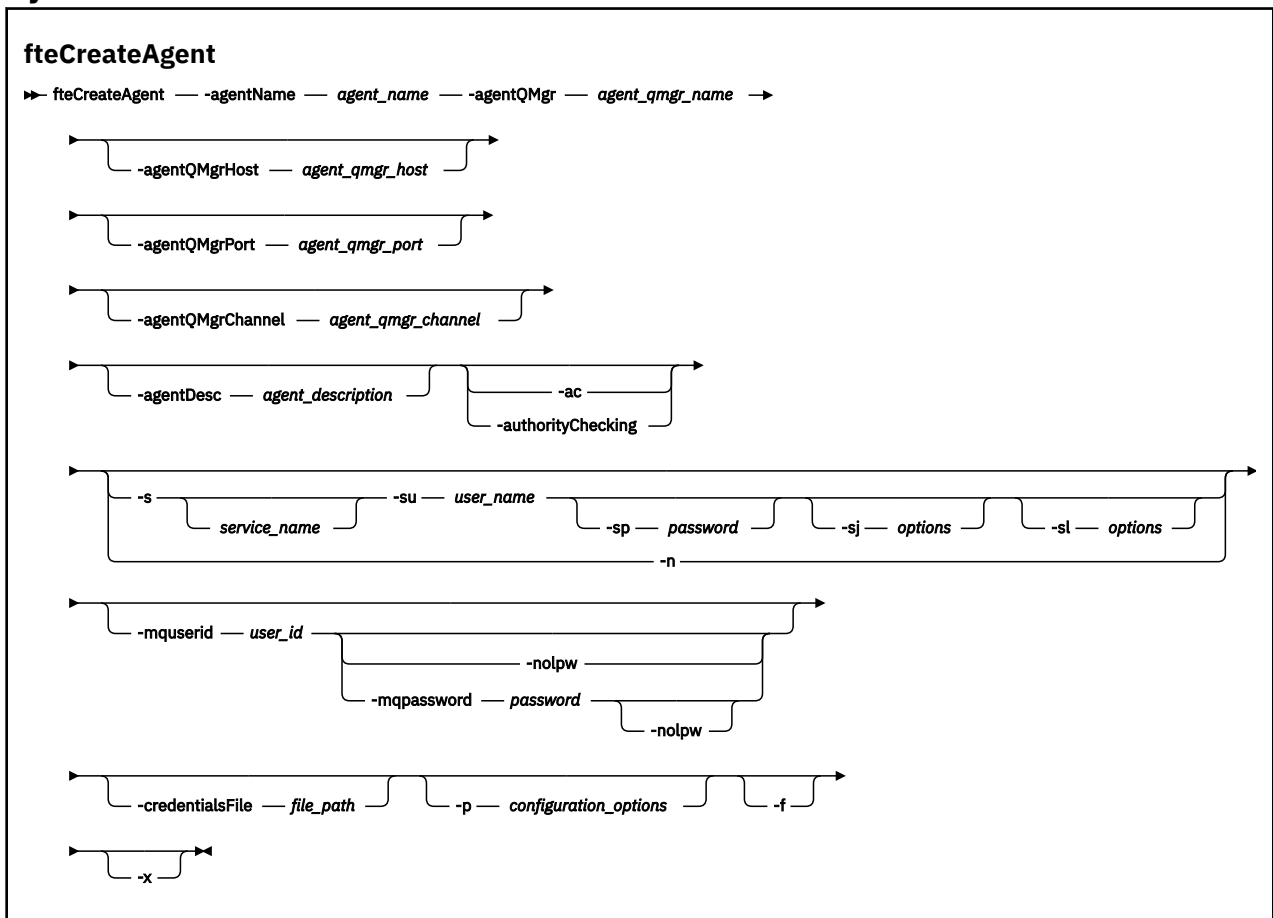
If you start an agent, or other commands that connect to the agent are run, and your path length exceeds 107 characters you receive the following message:

BFGNV0159E: Failed trying to bind to socket file with FFDC

## Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), backslash or forward slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double or single quotation marks or escape the special characters by using the escape sequence of the command shell. When you specify file paths on Windows, ensure the separator character backslash (\) is entered as double backslashes (\\), that is, escaped backslash (\\). Alternatively, you can use a single forward slash (/) character as a separator."

## Syntax



## Parameters

### -agentName *agent\_name*

Required. The name of the agent you want to create. The agent name must be unique to its coordination queue manager.

For more information about naming agents, see [Object naming conventions](#).

**-agentQMgr agent\_qmgr\_name**

Required. The name of the agent queue manager.

**-agentQMgrHost agent\_qmgr\_host**

Optional. The host name or IP address of the agent queue manager.

**-agentQMgrPort agent\_qmgr\_port**

Optional. The port number used for client connections to the agent queue manager.

**-agentQMgrChannel agent\_qmgr\_channel**

Optional. The channel name used to connect to the agent queue manager.

**-agentDesc agent\_description**

Optional. A description of the agent, which is displayed in IBM MQ Explorer.

**-ac or -authorityChecking**

Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see [Restricting user authorities on MFT agent actions](#).

**Windows**

**-s service\_name**

Optional (Windows only). Indicates that the agent is to run as a Windows service, the command must be run from a Windows administrator user ID. If you do not specify *service\_name*, the service is named `mqmftAgentAGENTQMGR`, where *AGENT* is the agent name and *QMGR* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Agent AGENT@QMGR**.

**Note:** If the redistributable agent is going to run as a Windows service, then the **BFG\_DATA** environment variable needs to be set in the system environment for the service to work.

**Windows**

**-su user\_name**

Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user account specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Troubleshooting an MFT agent or logger running as a Windows service](#).

Required when **-s** specified.

**Windows**

**-sp password**

Optional (Windows only).

This parameter is only valid when **-s** is specified. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

**Windows**

**-sj options**

Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of **-D** or **-X** that are passed to the JVM. The options are separated using a number sign (**#**) or semicolon (**;**) character. If you must embed any **#** or semicolon (**;**) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified.

**Windows**

**-sl options**

Optional (Windows only). Sets the Windows service log level. Valid options are: `error`, `info`, `warn`, `debug`. The default is `info`. This option can be useful if you are having problems with the Windows service. Setting it to `debug` gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

## Windows **-n**

Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither one of the **-s** parameter and the **-n** parameter is specified, then the agent is configured as a normal Windows process.

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the coordination queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using [MQCSP](#) authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### **-credentialsFile file\_path**

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

### **-credentialPath credentials\_path.**

This command defines the location to migrate the credential information to. This parameter can be a directory path to an existing credential file, or a directory path to a new credential file. **z/OS**  
On z/OS platforms the credential file can be a pre-existing partitioned data set extended (PDSE). The PDSE can include existing members, or a new member for the credential file. Existing members of the PDSE must be updated to include the credential file. The format of the PDSE must be variable blocked.

### **-f**

Optional. Forces the command to overwrite non-matching existing parameters. Specifying this parameter does not force the replacement of an existing Windows service agent.

### **-? or -h**

Optional. Displays command syntax.

### **-x**

Optional. Creates an agent configuration to run in a high availability mode.

Specifying this parameter adds a new option `highlyAvailable` to the `agent.properties` file.

### Example

In this example, AGENT3 is created with an agent queue manager QM\_NEPTUNE and uses the default coordination queue manager:

```
fteCreateAgent -agentName AGENT3 -agentQMGr QM_NEPTUNE  
-agentQMGrHost myhost.ibm.com -agentQMGrPort 1415 -agentQMGrChannel CHANNEL1
```

In this example, AGHA is created in high availability mode with an agent queue manager QMHA.

```
fteCreateAgent -agentName AGHA -agentQMGr QMHA -x
```

### Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

### Related concepts

[Troubleshooting an MFT agent or logger running as a Windows service](#)

### Related tasks

[Starting an MFT agent as a Windows service](#)

### Related reference

[“fteStartAgent \(start an MFT agent\)” on page 2140](#)


The **fteStartAgent** command starts a Managed File Transfer agent from the command line.


[“fteDeleteAgent \(delete an MFT agent and its configuration\)” on page 2073](#)

The **fteDeleteAgent** command deletes a Managed File Transfer Agent and its configuration. If the agent is a protocol bridge agent, the user credentials file is left on the file system.

## fteCreateBridgeAgent (create and configure an MFT protocol bridge agent)

The **fteCreateBridgeAgent** command creates a Managed File Transfer protocol bridge agent and its associated configuration. Create a protocol bridge agent for each file server that you want to send files to and receive files from.

**Important:**  On IBM MQ for AIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the command:

- Be a member of the mqm group (if the mqm group is defined on the system).
- Be a member of the group named in the BFG\_GROUP\_NAME environment variable (if one is named).
- Have no value set in the BFG\_GROUP\_NAME environment variable when the command is run.

### Purpose

Use the **fteCreateBridgeAgent** command to create a protocol bridge agent. For an overview of how to use the protocol bridge, see [The protocol bridge](#). This **fteCreateBridgeAgent** command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.*agent\_name*

- SYSTEM.FTE.AUTHAGT1.agent\_name
- SYSTEM.FTE.AUTHMON1.agent\_name
- SYSTEM.FTE.AUTHOPS1.agent\_name
- SYSTEM.FTE.AUTHSCH1.agent\_name
- SYSTEM.FTE.AUTHTRN1.agent\_name
- SYSTEM.FTE.COMMAND.agent\_name
- SYSTEM.FTE.DATA.agent\_name
- SYSTEM.FTE.EVENT.agent\_name
- SYSTEM.FTE.REPLY.agent\_name
- SYSTEM.FTE.STATE.agent\_name
- SYSTEM.FTE.HA.agent\_name

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location:

`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_create.mqsc`

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues use by the agent. The MQSC commands are in a file in the following location:

`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_delete.mqsc`.

The **fteCreateBridgeAgent** command creates a `ProtocolBridgeProperties.xml` XML file in the following directory:  
`MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name`.

Users are responsible for manually creating the `ProtocolBridgeCredentials.xml` file, it is no longer created by the **fteCreateBridgeAgent** command.

The `ProtocolBridgeCredentials.xml` file allows you to define user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server and the `ProtocolBridgeProperties.xml` file allows you to define multiple protocol file servers so you can transfer to multiple endpoints.

There is a sample `ProtocolBridgeCredentials.xml` in the `MQ_INSTALLATION_PATH/mqft/samples/credentials/` directory. For more information, see [“Protocol bridge credentials file format”](#) on page 2700 and [“Protocol bridge properties file format”](#) on page 2704.

If you run the **fteCreateBridgeAgent** command and specify a default protocol file server (parameter `-bt`), this default server is contained in the `ProtocolBridgeProperties.xml` file and its host name is used for the server name. With the `-bt` parameter, you need to specify the following parameters:

- `-bh`
- `-btz`
- `-bm`
- `-bsl`
- `-bfe`
- `-bts`

If you do not specify a default server, there are no entries in the `ProtocolBridgeProperties.xml` file; you must add at least one server manually before transfers can take place.

Managed File Transfer provides advanced agent properties that help you configure protocol bridge agents. The properties that relate to the protocol bridge start with `protocol`. These properties are described in [Advanced agent properties: Protocol bridge](#) and [Advanced agent properties: Protocol bridge agent](#)

logging. If you see unexpected behavior in the protocol bridge, review these protocol properties and ensure that you have set these properties correctly for your system.

If you see the following output from the **fteCreateBridgeAgent** command:

```
BFGMQ1007I: The coordination queue manager cannot be contacted or has refused a connection attempt.
The WebSphere MQ reason code was 2058. The agent's presence will not be published.
```

it indicates that the coordination queue manager can not be contacted and provides the IBM MQ reason code for why. This information message can indicate that the coordination queue manager is currently unavailable or that you have defined the configuration incorrectly.

### **Important:**

On AIX and Linux Managed File Transfer commands use socket files to communicate with the agent process running on the same host machine.

These socket files are created in the log directory of the agent and are deleted when an agent stops. In the IBM MQ Managed File Transfer installation, this socket file is created with a file path of: <MQ\_DATA\_PATH>/mqft/logs/<COORDINATION\_QM\_NAME>/agents/<AGENT\_NAME>/logs/<AGENT\_NAME>@<AGENT\_QM\_NAME> where MQ\_DATA\_PATH is /var/mqm by default.

For a re-distributable agent, this socket file is created under the directory: <RE\_DISTRIBUTABLE\_DIRECTORY>/mqft/logs/<COORDINATION\_QM\_NAME>/agents/<AGENT\_NAME>/logs/<AGENT\_NAME>@<AGENT\_QM\_NAME>.

For example, if the agent name is SRCAGENT, the agent queue manager name is SRCAGENTQM, the coordination queue manager name is COORDQM, and the redistributable agent is running from the directory /home/myuser/mqmft-redis, the full path of this socket file is: /home/myuser/mqmft-redis/mqft/logs/COORDQM/agents/SRCAGENT/logs/SRCAGENT@SRCAGENTQM

which is a total file path length of 85 characters.

The maximum path length allowed by these operating systems for a socket file is 107 characters. Therefore, when creating an agent, take care to ensure that the socket file path does not exceed 107 characters. This is particularly important with a redistributable agent where the log directory of the agent can be located in an arbitrary directory location. See the **fteCreateEnvironment** command for details on setting up the configuration directory.

If you start an agent, or other commands that connect to the agent are run, and your path length exceeds 107 characters you receive the following message:

```
BFGNV0159E: Failed trying to bind to socket file with FFDC
```

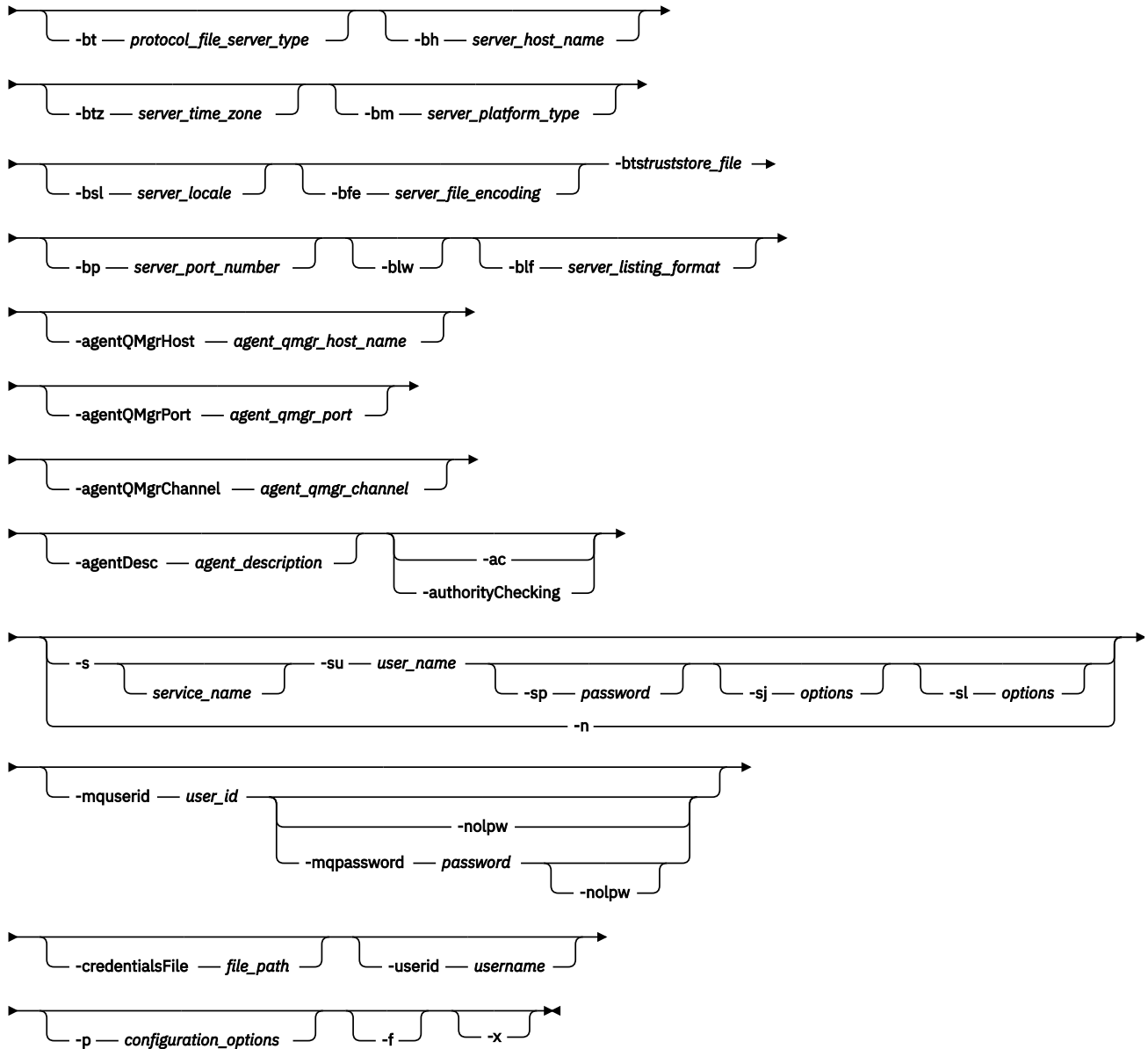
## **Special characters**

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), backslash or forward slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double or single quotation marks or escape the special characters by using the escape sequence of the command shell. When you specify file paths on Windows, ensure the separator character backslash (\) is entered as double backslashes (\\), that is, escaped backslash (\\). Alternatively, you can use a single forward slash (/) character as a separator."

## Syntax

### fteCreateBridgeAgent

► fteCreateBridgeAgent — -agentName *agent\_name* -agentQMgr *agent\_qmgr\_name* →



## Parameters

### **-agentName** *agent\_name*

Required. The name of the agent you want to create. The agent name must be unique in its administrative domain.

For more information about naming agents, see [Object naming conventions](#).

### **-agentQMgr** *agent\_qmgr\_name*

Required. The name of the agent queue manager.

### **-bt** *protocol\_file\_server\_type*

Optional. Specifies that you want to define a default protocol file server. Specify one of the following options:

**FTP**

Standard FTP server

**SFTP**

SSH FTP server

**FTPS**

FTP server secured using SSL or TLS

If you do not specify this parameter, no default protocol server is defined.

**-bh server\_host\_name**

Required only if you also specify a default protocol file server using the **-bt** parameter. The IP host name or IP address of the protocol file server.

**-btz server\_time\_zone**

Required only if you also specify the **-bt** parameter (FTP and FTPS servers only). The time zone of the protocol file server. Specify the time zone in the following format: Area/Location. For example: Europe/London.

You can use the **-htz** parameter to list the possible values for **-btz**. For example:  
`fteCreateBridgeAgent -htz`

**-bm server\_platform**

Required only if you also specify a default protocol file server using the **-bt** parameter. The platform type of the protocol file server. Specify one of the following options:

**UNIX**

Generic UNIX and Linux platform

**WINDOWS**

Generic Windows platform

**OS400**

IBM i platform

**Note:** You must set the both the **bm** parameter to *OS400* and the **blf** parameter to *OS400IFS* if the bridge agent is to communicate with an FTP server running IBM i.

**-bsl server\_locale**

Required only if you also specify the **-bt** parameter (FTP and FTPS servers only). The locale of the protocol file server. Specify the locale in the following format: *xx\_XX*. For example: *en\_GB*.

- *xx* is the ISO Language Code. For a list of valid values, see [Codes for the Representation of Names of Languages](#)
- *XX* is the ISO Country Code. For a list of valid values, see [Country names and code elements](#)

**-bfe server\_file\_encoding**

Required only if you also specify a default protocol file server using the **-bt** parameter. The character encoding format of the files stored on the protocol file server. For example: UTF-8.

You can use the **-hcs** parameter to list the possible values for **-bfe**. For example:  
`fteCreateBridgeAgent -hcs`

**-bts truststore\_file**

Required when you specify the **-bt** parameter (FTPS servers only). Specifies the path to a truststore that is used to validate the certificate presented by the FTPS server.

You can specify the **-bts** parameter only if you have also specified the FTPS option on the **-bt** parameter.

**-bp server\_port**

Optional. The IP port that the protocol file server is connected to. Specify this parameter only if your protocol file server does not use the default port for that protocol. If you do not specify this parameter, Managed File Transfer uses the default port for the protocol type of file server.

**-blw**

Optional. Defines the protocol file server as having limited write abilities. By default, a protocol bridge agent expects the protocol file server to permit file deletion, file renaming, and file opening for append



writing. Specify this parameter to indicate that the protocol file server does not permit these file actions. Instead the file server permits read from and write to file only. If you specify this parameter, any transfers might not be recoverable if they are interrupted and might result in a failure for the file currently being transferred.

### **-blf *server\_listing\_format***

Optional and for FTP and FTPS servers only. Defines the server listing format of the listed file information returned from the default protocol file server. The options are as follows:

#### **UNIX**

Generic UNIX and Linux platform

#### **WINDOWS**

Generic Windows platform

#### **OS400IFS**

Root file system on IBM i platform

#### **Notes:**

1. You must set the both the **bm** parameter to *OS400* and the **blf** parameter to *OS400IFS* if the bridge agent is to communicate with an FTP server running IBM i.
2. You can use Managed File Transfer to send and receive files on the root (/) file system only. Other file systems do not work.

To identify which format to select, use a FTP client program and perform a listing of a directory and select which format is the best fit. For example,

**UNIX** UNIX displays the following type of listing:

```
-rwxr-xr-x 2 userid groupId 4096 2009-07-23 09:36 filename
```

**Windows** Windows displays the following type of listing:

```
437,909 filename
```

**IBM i** IBM i displays the following type of listing:

```
OS400IFS -rwxrwsrwx 3 USERID 0 8192 Mar 7 08:33 filename
```

The default is UNIX, which is the format used by most servers.

### **-agentQMGrHost *agent\_qmgr\_host***

Optional. The host name or IP address of the agent queue manager.

### **-agentQMGrPort *agent\_qmgr\_port***

Optional. The port number used for client connections to the agent queue manager.

### **-agentQMGrChannel *agent\_qmgr\_channel***

Optional. The channel name used to connect to the agent queue manager.

### **-agentDesc *agent\_description***

Optional. A description of the agent, which is displayed in the IBM MQ Explorer.

### **-ac or -authorityChecking**

Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see [Restricting user authorities on MFT agent actions](#).

### **Windows -s *service\_name***

Optional (Windows only). Indicates that the agent is to run as a Windows service. If you do not specify *service\_name*, the service is named `mqmftAgentAGENTQMGR`, where *AGENT* is the agent name and *QMGR* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Agent AGENT@QMGR**.

**Windows -su user\_name**

Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Troubleshooting an MFT agent or logger running as a Windows service](#).

Required when **-s** specified.

**Windows -sp password**

Optional (Windows only). Password for the user account set by the **-su** parameter.

This parameter is only valid when **-s** is specified. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

**Windows -sj options**

Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of **-D** or **-X** that are passed to the JVM. The options are separated using a number sign (**#**) or semicolon (**;**) character. If you must embed any **#** or semicolon (**;**) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified. .

**Windows -sl options**

Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

**Windows -n**

Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither one of the **-s** parameter and the **-n** parameter is specified, then the agent is configured as a normal Windows process.

**-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateBridgeAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in the `installation.properties` file are used. See [Configuration options](#) for more information.

**-f**

Optional. Forces the command to overwrite the existing configuration.

**-htz**

Optional. Displays a list of supported time zones that you can use as input for the **-btz** parameter.

**-hcs**

Optional. Displays a list of supported character sets that you can use as input for the **-bfe** parameter.

Run the **fteCreateBridgeAgent -hcs** command to list the known code pages for the JVM. This information is not available from an external source because the known code pages vary between JVMs.

**-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

**-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using [MQCSP](#) authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

**-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**-credentialsFile file\_path**

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

**-userid username**

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

**-? or -h**

Optional. Displays command syntax.

**-x**

Optional. Creates an agent configuration to run in a high availability mode.

Specifying this parameter adds a new option `highlyAvailable` to the `agent.properties` file.

**Examples**

In this example, a new protocol bridge agent ACCOUNTS1 is created with an agent queue manager QM\_ACCOUNTS and uses the default coordination queue manager. ACCOUNTS1 connects to the FTP server accountshost.ibm.com. This FTP server runs on Windows using a time zone of Europe/Berlin, a locale of de\_DE, and a file encoding of UTF-8. The number of reconnect retries is 4:

```
fteCreateBridgeAgent -agentName ACCOUNTS1 -agentQMgr QM_ACCOUNTS -bt FTP
-bh accountshost.ibm.com -bm WINDOWS -btz Europe/Berlin -bsl de_DE -bfe UTF8
-agentQMgrHost myhost.ibm.com -agentQMgrPort 1415 -agentQMgrChannel CHANNEL1
```

In this example, a new protocol bridge agent ACCOUNTS2 is created with an agent queue manager QM\_ACCOUNTS and uses the default coordination manager. ACCOUNTS2 is created without a default protocol file server.

```
fteCreateBridgeAgent -agentName ACCOUNTS2 -agentQMgr QM_ACCOUNTS
```

**Note:** The above does not apply to Managed File Transfer Agent redistributable.

The scenario here is that the Managed File Transfer Agent is running on a Linux or Windows box but configured to communicate with an FTP server running IBM i. If you require the destination file to be in

the native code page of IB, you must use the **-dce** code page parameter while submitting the transfer request. For example:

```
fteCreateTransfer -rt -1 -sa SRC -sm MFTQM -da OS400FTP -dm MFTQM -dce 37 -sce 1252
-t text -de overwrite -df "<your-domain>:/home/mft/text/uploadwcp.log"
"C:\temp\os400\Text\uploadwcp.log"
```

and, if you require the receiving file in the native code page from IBM i:

```
fteCreateTransfer -rt -1 -da SRC -dm MFTQM -sa OS400FTP -sm MFTQM -sce 37 -dce 1252
-t text -de overwrite -df "C:\temp\os400\Text\downloadwcp.log"
"<your-domain>:/home/mft/text/uploadwcp.log"
```

## Additional customizing

If you used the **-bt** parameter (and the additional parameters that are required) there will be a default server name in the `ProtocolBridgeProperties.xml` file.

If you want to add additional ftp servers, or change the location of the credentials file, see [Defining properties for protocol file servers using the ProtocolBridgeProperties.xml file](#).


## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

Use the **fteStartAgent** command to start your protocol bridge agent. For more information, see [“fteStartAgent \(start an MFT agent\)” on page 2140](#).  See also [Starting an MFT agent on z/OS](#).

## Related reference

[The protocol bridge](#)

[“Protocol bridge credentials file format” on page 2700](#)


The `ProtocolBridgeCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.


[“Protocol bridge properties file format” on page 2704](#)

The `ProtocolBridgeProperties.xml` file in the agent configuration directory defines properties for protocol file servers.

## fteCreateCDAgent (create a Connect:Direct bridge agent)

The `fteCreateCDAgent` command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

**Important:**  On IBM MQ for AIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the `mqm` group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the command:

- Be a member of the `mqm` group (if the `mqm` group is defined on the system).
- Be a member of the group named in the `BFG_GROUP_NAME` environment variable (if one is named).
- Have no value set in the `BFG_GROUP_NAME` environment variable when the command is run.

## Purpose

Use the **fteCreateCDAgent** command to create a Connect:Direct bridge agent. This type of agent is dedicated to transferring files to and from Connect:Direct nodes. For more information, see [The Connect:Direct bridge](#). For details of the supported operating system versions for the Connect:Direct bridge, see the web page [System Requirements for IBM MQ](#).

This command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.*agent\_name*
- SYSTEM.FTE.AUTHAGT1.*agent\_name*
- SYSTEM.FTE.AUTHMON1.*agent\_name*
- SYSTEM.FTE.AUTHOPS1.*agent\_name*
- SYSTEM.FTE.AUTHSCH1.*agent\_name*
- SYSTEM.FTE.AUTHTRN1.*agent\_name*
- SYSTEM.FTE.COMMAND.*agent\_name*
- SYSTEM.FTE.DATA.*agent\_name*
- SYSTEM.FTE.EVENT.*agent\_name*
- SYSTEM.FTE.REPLY.*agent\_name*
- SYSTEM.FTE.STATE.*agent\_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location:

```
MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_create.mqsc.
```

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues belonging to the agent. The MQSC commands are in a file in the following location:

```
MQ_DATA_PATH\mqft\config\coordination_qmgr_name\agents\agent_name\agent_name_delete.mqsc.
```

Managed File Transfer provides advanced agent properties that help you configure agents. These properties are described in [The MFT agent.properties file](#).

The **fteCreateCDAgent** command creates two XML files in the agent properties directory. `ConnectDirectNodeProperties.xml`, which is used to define information about the remote nodes in a transfer, and `ConnectDirectProcessDefinitions.xml`, which is used to specify which user-defined Connect:Direct processes are started by transfers.

To define user names and passwords that the Connect:Direct bridge agent uses to connect to Connect:Direct nodes, you must manually create a `ConnectDirectCredentials.xml` file. Sample XML files are located in `MQ_INSTALLATION_PATH/mqft/samples/credentials/`. For more information and examples, see [“Connect:Direct credentials file format” on page 2714](#).

### Important:

On AIX and Linux Managed File Transfer commands use socket files to communicate with the agent process running on the same host machine.

These socket files are created in the log directory of the agent and are deleted when an agent stops. In the IBM MQ Managed File Transfer installation, this socket file is created with a file path of: `<MQ_DATA_PATH>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>` where `MQ_DATA_PATH` is `/var/mqm` by default.

For a re-distributable agent, this socket file is created under the directory: `<RE_DISTRIBUTABLE_DIRECTORY>/mqft/logs/<COORDINATION_QM_NAME>/agents/<AGENT_NAME>/logs/<AGENT_NAME>@<AGENT_QM_NAME>`.

For example, if the agent name is SRCAGENT, the agent queue manager name is SRCAGENTQM, the coordination queue manager name is COORDQM, and the redistributable agent is running from the directory /home/myuser/mqmftr-dist, the full path of this socket file is: /home/myuser/mqmftr-dist/mqft/logs/COORDQM/agents/SRCAGENT/logs/SRCAGENT@SRCAGENTQM

which is a total file path length of 85 characters.

The maximum path length allowed by these operating systems for a socket file is 107 characters. Therefore, when creating an agent, take care to ensure that the socket file path does not exceed 107 characters. This is particularly important with a redistributable agent where the log directory of the agent can be located in an arbitrary directory location. See the **fteCreateEnvironment** command for details on setting up the configuration directory.

If you start an agent, or other commands that connect to the agent are run, and your path length exceeds 107 characters you receive the following message:

```
BFGNV0159E: Failed trying to bind to socket file with FFDC
```

## Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), backslash or forward slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double or single quotation marks or escape the special characters by using the escape sequence of the command shell. When you specify file paths on Windows, ensure the separator character backslash (\) is entered as double backslashes (\\), that is, escaped backslash (\\). Alternatively, you can use a single forward slash (/) character as a separator."

## fteCreateCDAgent

► fteCreateCDAgent — -agentName — *agent\_name* — -agentQMgr — *agent\_qmgr\_name* — ►

► -cdNode — *cd\_node\_name* — }  
-agentQMgrHost — *agent\_qmgr\_host* — }

► -agentQMgrPort — *agent\_qmgr\_port* — }

► -agentQMgrChannel — *agent\_qmgr\_channel* — }

► -agentDesc — *agent\_description* — }  
-ac — }  
-authorityChecking — }

► -p — *configuration\_options* — }  
-f — }  
-cdNodeHost — *cd\_node\_host* — }

► -cdNodePort — *cd\_node\_port* — }  
-cdTmpDir — *cd\_tmp\_dir* — }

► -s — *service\_name* — }  
-su — *user\_name* — }  
-sp — *password* — }  
-sj — *options* — }  
-sl — *options* — }  
-n — }

► -mquserid — *user\_id* — }  
-nolpw — }  
-mqpassword — *password* — }  
-nolpw — }

► -credentialsFile — *file\_path* — }  
-userid — *username* — }

## Parameters

### -agentName *agent\_name*

Required. The name of the agent you want to create. The agent name must be unique to its coordination queue manager.

For more information about naming agents, see [Object naming conventions](#).

### -agentQMgr *agent\_qmgr\_name*

Required. The name of the agent queue manager.

### -cdNode *cd\_node\_name*

Required. The name of the Connect:Direct node to use to transfer messages from this agent to destination Connect:Direct nodes. The value of this parameter is used for logging and not to specify to the Connect:Direct bridge agent which node to connect to. The values of the **-cdNodeHost** and **-cdNodePort** specify the Connect:Direct node that is part of the Connect:Direct bridge.

### -agentQMgrHost *agent\_qmgr\_host*

Optional. The host name or IP address of the agent queue manager.

### -agentQMgrPort *agent\_qmgr\_port*

Optional. The port number used for client connections to the agent queue manager.

### -agentQMgrChannel *agent\_qmgr\_channel*

Optional. The channel name used to connect to the agent queue manager.

### -agentDesc *agent\_description*

Optional. A description of the agent, which is displayed in IBM MQ Explorer.

**-ac or -authorityChecking**

Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see [Restricting user authorities on MFT agent actions](#).

**-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateCDAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

**-f**

Optional. Forces the command to overwrite non-matching existing parameters. Specifying this parameter does not force the replacement of an existing Windows service agent.

**-cdNodeHost cd\_node\_host\_name**

Optional. The host name or IP address of the system where the Connect:Direct node, specified by the **-cdNode** parameter, is located. If you do not specify the **-cdNodeHost** parameter, a default of the host name or IP address of the local system is used.

In most cases, the Connect:Direct node is on the same system as the Connect:Direct bridge agent. In these cases, the default value of this property, which is the IP address of the local system, is correct. If your system has multiple IP addresses, or your Connect:Direct node is on a different system to your Connect:Direct bridge agent and their systems share a file system, use this property to specify the correct host name for the Connect:Direct node.

**-cdNodePort cd\_node\_port\_name**

Optional. The port number of the Connect:Direct node that client applications use to communicate with the node that is specified by the **-cdNode** parameter. In Connect:Direct product documentation, this port is referred to as the API port. If you do not specify the **-cdNodePort** parameter, a default port number of 1363 is assumed.

**-cdTmpDir cd\_tmp\_directory**

Optional. The directory to be used by this agent to store files temporarily before they are transferred to the destination Connect:Direct node. This parameter specifies the full path of the directory where files are temporarily stored. For example, if **cdTmpDir** is set to /tmp then the files are temporarily placed in the /tmp directory. If you do not specify the **-cdTmpDir** parameter, the files are stored temporarily in a directory named *cdbridge-agent\_name*. This default directory is created in the location that is defined by the value of the `java.io.tmpdir` property.

The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to access the directory specified by this parameter using the same path name. Consider this when planning the installation of your Connect:Direct bridge. If possible, create the agent on the system where the Connect:Direct node that is part of the Connect:Direct bridge is located. If your agent and node are on separate systems, the directory must be on a shared file system and be accessible from both systems using the same path name. For more information about the supported configurations, see [The Connect:Direct bridge](#).

**Note:** If you run the **fteCleanAgent** command, all files in this directory are deleted.

**Windows -s service\_name**

Optional (Windows only). Indicates that the agent is to run as a Windows service, the command must be run from a Windows administrator user ID. If you do not specify *service\_name*, the service is named `mqmftAgentAGENTQMGR`, where *AGENT* is the agent name and *QMGR* is your agent queue manager name.

**Windows -su user\_name**

Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user



account specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain specify the value in the form `UserName`.

**Windows -sp password**

Optional (Windows only).

**Windows -sj options**

Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of `-D` or `-X` that are passed to the JVM. The options are separated using a number sign (`#`) or semicolon (`;`) character. If you must embed any `#` or semicolon (`;`) characters, put them inside single quotation marks.

**Windows -sl options**

Optional (Windows only). Sets the Windows service log level. Valid options are: `error`, `info`, `warn`, `debug`. The default is `info`. This option can be useful if you are having problems with the Windows service. Setting it to `debug` gives more detailed information in the service log file.

**Windows -n**

Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the `-s` option. If neither one of the `-s` parameter and the `-n` parameter is specified, then the agent is configured as a normal Windows process.

**-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

**-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the `-mqpassword` parameter is specified, you must also specify the `-mquserid` parameter. If you specify `-mquserid`, but do not specify `-mqpassword`, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

**V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the `-nolpw` parameter is specified, you must also specify the `-mquserid` parameter. If you specify `-mquserid` and `-nolpw`, but do not specify `-mqpassword`, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the `-nolpw` option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**-credentialsFile file\_path**

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the `-f` force parameter.

**-userid username**

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the `-credentialsFile` parameter.

**Example**

In this example, a new Connect:Direct bridge agent `CD_BRIDGE` is created with an agent queue manager `QM_NEPTUNE`. The agent uses the Connect:Direct node `BRIDGE_NODE` to transfer files to other Connect:Direct nodes. The `BRIDGE_NODE` node is located on the same system as the agent and uses the

default port for client connections. Files that are transferred to or from Connect:Direct are temporarily stored in the directory /tmp/cd-bridge.

```
fteCreateCDAgent -agentName CD_BRIDGE -agentQMgr QM_NEPTUNE  
                -cdNode BRIDGE_NODE -cdTmpDir /tmp/cd-bridge
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related tasks

[Configuring the Connect:Direct bridge](#)

[Transferring a file to a Connect:Direct node](#)

[Transferring a file from a Connect:Direct node](#)

## Related reference

[The Connect:Direct bridge](#)

## fteCreateEnvironment (set up environment for Redistributable Managed File Transfer Agent)

The **fteCreateEnvironment** command sets the environment for the configuration and transfer of files for the Redistributable Managed File Transfer Agent.

## Purpose

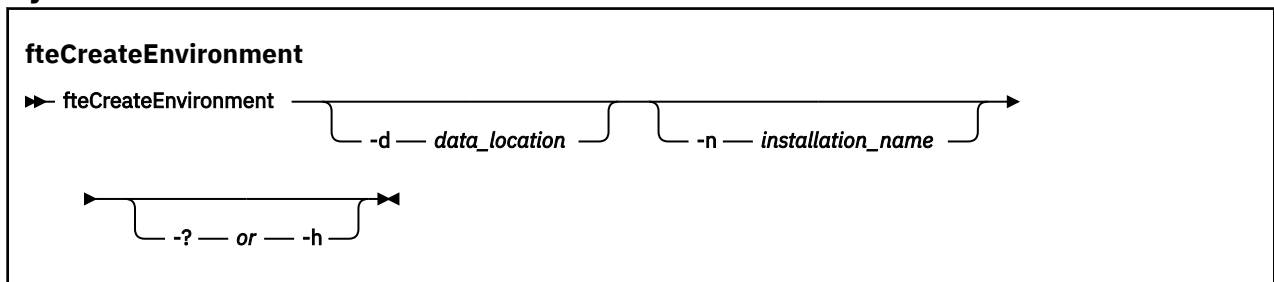
You use the **fteCreateEnvironment** command to set up the environment for using the Redistributable Managed File Transfer Agent.

**V 9.3.0** From IBM MQ 9.3.0, this environment is shared with the Redistributable Managed File Transfer Logger.

You can run this command with the **-d** parameter to specify the location for the MFT Agent data files. If you do not specify the **-d** parameter, the command creates the data files in the Redistributable Managed File Transfer Agent download location and sets the data path.

From IBM MQ 9.2.0, the **-n** parameter gives you the option of specifying an IBM MQ installation name. The value you specify for this option is used for the rest of the MFT commands run from the same console session.

## Syntax



## Parameters

### **-d *data\_location***

Optional. This parameter is used for specifying the location of the data files at the time when the environment is set up.

If you do not specify this parameter, the data directory (if it does not already exist) is created in the location where the Redistributable Managed File Transfer Agent is extracted and the environment variable (BFG\_DATA) is set for this location.

### **-? or -h**

Optional. Displays command syntax.

### **-n *installation\_name***

Optional. This parameter is used for specifying the name of an IBM MQ installation, or a unique name. The unique name must adhere to the same naming standards as the IBM MQ installation name. For more information about the naming standards, see [Installation name on AIX, Linux, and Windows](#).

Examples of situations in which you might want to use this parameter are:

- If you want to quickly test a new function or feature using the redistributable package with the existing configuration where agents have been configured to connect to queue manager in clients mode only. (Note that this parameter does not apply to any agent that is configured connect to a queue manager in bindings mode.)
- If you are migrating from a standard Managed File Transfer installation to a Redistributable Managed File Transfer Agent package, and you want to use the same configuration as the one that was created by the standard installation. This is the case where standard Managed File Transfer has been installed but is connecting to an agent queue manager running on another machine.

The default installation name variable is **BFG\_INSTALLATION\_NAME**.

## Examples

**Windows** In this example, on Windows, the -d parameter specifies the location where the data folders are created:

```
fteCreateEnvironment -d C:\mftRedistributable\mftData
```

**Linux** On Linux, as a prerequisite, the command has to be run on a bash shell. In a bash shell, the command can be run in various ways, and the command file needs to be sourced:

```
source Path_of_MFTZipBin/fteCreateEnvironment
```

An alternative method is:

```
. Path_of_MFTZipBin/fteCreateEnvironment
```

or, if running from the directory where the command file is present:

```
./fteCreateEnvironment
```



**Attention:** Note the space following the first period character (.)

This example creates an environment where you specify both the MFT configuration data path, and installation name environment variables:

```
fteCreateEnvironment -d C:/ProgramData/IBM/mq/mqft -n MFTPROD
```

The output from this command is:

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
```

```
BFG_DATA is C:\ProgramData\IBM\MQ
BFG_INSTALLATION_NAME is MFTPROD
```

Both the **BFG\_INSTALLATION\_NAME** and **BFG\_DATA** environment variables get updated to new values.

This example creates a new environment variable for the installation name only. The data path remains unchanged at C:\ProgramData\IBM\MQ.

```
fteCreateEnvironment -n MFTPROD
```

The output from the command is:

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFG_DATA is C:\ProgramData\IBM\MQ
BFG_INSTALLATION_NAME is MFTPROD
```

The **BFG\_INSTALLATION\_NAME** environment variable is updated to the new value *MFTPROD*.

This example creates a new environment variable for the path of the MFT configuration data only. The installation name remains unchanged at *MFTPROD*:

```
fteCreateEnvironment -d C:/ProgramData/IBM/MQ2
```

The output from the command is:

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFG_DATA is C:\ProgramData\IBM\MQ2
BFG_INSTALLATION_NAME is MFTPROD
```

## Return codes

**0**

Command completed successfully.

**1**


Command ended unsuccessfully.


## Related tasks

[Downloading and configuring Redistributable Managed File Transfer components](#)

## fteCreateLogger (create an MFT file or database logger)

Use the **fteCreateLogger** command to create a Managed File Transfer file or database logger.

**Important:**  On IBM MQ for AIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the command:

- Be a member of the mqm group (if the mqm group is defined on the system).
- Be a member of the group named in the BFG\_GROUP\_NAME environment variable (if one is named).
- Have no value set in the BFG\_GROUP\_NAME environment variable when the command is run.

## Loggers on IBM i



Managed File Transfer loggers are not supported on the IBM i platform.

## Purpose

The **fteCreateLogger** command provides you with the MQSC commands that you must run against your logger command queue manager to create the following logger queues:

- SYSTEM.FTE.LOG.CMD.*logger\_name*
- SYSTEM.FTE.LOG.RJCT.*logger\_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the logger. The MQSC commands to run are also supplied in a file in the following location:

*MQ\_DATA\_PATH*\mqft\config\coordination\_qmgr\loggers\*logger\_name*\*logger\_name\_create.mqsc*

If you later want to delete the logger, use the **fteDeleteLogger** command.

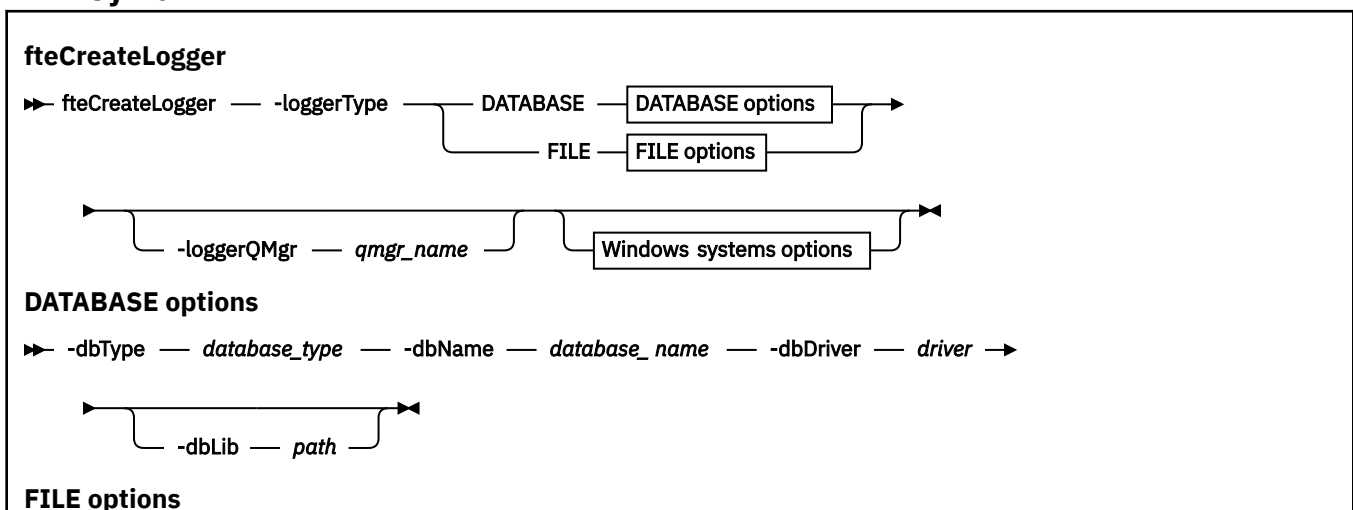
Managed File Transfer provides advanced logger properties that help you configure loggers. See, [MFT logger configuration properties](#)

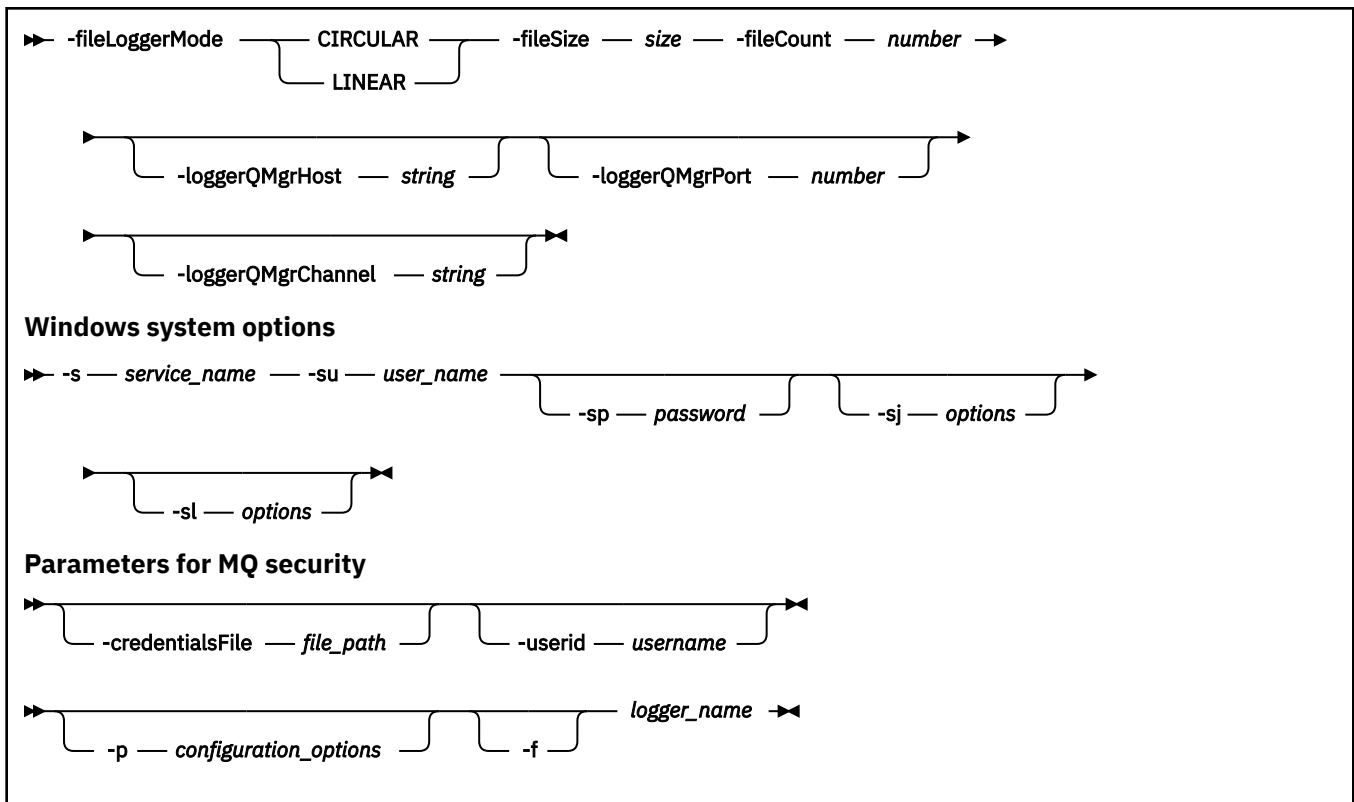
**Note:** If the logger you are creating is a database logger and it is not connecting to a local Db2 database, you will need to manually create a *MQMFTCredentials.xml* file. The file contains the user name and password for connecting to the database. You should use the property file, *wmqfte.database.credentials*, in the *logger.properties* file to specify the path to the *MQMFTCredentials.xml* file. A sample of this credentials file is located in *MQ\_INSTALLATION\_PATH*/mqft/samples/credentials/.

## Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), backslash or forward slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double or single quotation marks or escape the special characters by using the escape sequence of the command shell. When you specify file paths on Windows, ensure the separator character backslash (\) is entered as double backslashes (\\), that is, escaped backslash (\). Alternatively, you can use a single forward slash (/) character as a separator."

## Syntax





## Parameters

### -loggerType *type*

Required. Specifies where managed file transfer information will be logged. The options for type are either DATABASE, if transfer information will be logged to a database, or FILE, if the information will be logged to a file.

### -loggerQMgr *qmgr\_name*

Optional. Determines the queue manager to connect to in order to receive messages containing information about managed file transfers. The queue manager must be on the same system as the logger. If you do not specify the **-loggerQMgr** parameter then the coordination queue manager that is associated with the configuration options set for this logger is used as the default.

From IBM MQ 9.1, if the coordination queue manager connects using a clients mode connection, the logger uses clients mode.



**Attention:** The **loggerQMgrHost**, **loggerQMgrPort**, and **loggerQMgrChannel** parameters are valid on a File logger only. If you attempt to use any one, or more, of these parameters on a Database logger, you receive the following message:

```
BFGCL0456E: The parameter '-loggerQMgrHost' is not valid for the fteCreateLogger
command.
```

### -dbType *database\_type*

Required when `-loggerType` is DATABASE. Specifies the type of database management system in use for storing managed file transfer information. The options are `db2` or `oracle`

**Note:** You need to create tables by using SQL files. The `.sql` files are available from `MQ_INSTALLATION_PATH_/mqft/sql`:

- For Db2 databases: `ftelog_tables_db2.sql`
- For Oracle databases: `ftelog_tables_oracle.sql`

**-dbName *database\_name***

Required when `-loggerType` is DATABASE. The name of the database where managed file transfer information is stored. The database must be configured with the Managed File Transfer log tables.

**-dbDriver *driver***

Required when `-loggerType` is DATABASE. The location of the JDBC driver classes for the database. This is typically the path and file name of a JAR file.

**-dbLib *path***

Optional when `-loggerType` is DATABASE. The location of any native libraries needed by your chosen database driver.

**-fileLoggerMode *mode***

Required when `-loggerType` is FILE. Specifies the type of file system in use for storing managed file transfer information. The options are LINEAR or CIRCULAR.

Option LINEAR means the file logger will write information to a file until that file reaches its maximum size as defined by `-fileSize`. When the maximum size is reached the file logger will start a new file. Previously written files will not be deleted which allows them to be kept as a historical record of log messages. Files are not deleted when running in this mode, so the `-fileCount` will be ignored as there is no upper limit to the number of files that can be created. As there is no upper limit when running in this mode it will be necessary to track the amount of disk space used by the log files in order to avoid running low on disk space.

Option CIRCULAR means the file logger will write information to a file until that file reaches its maximum size as defined by `-fileSize`. When the maximum size is reached the file logger will start a new file. The maximum number of files written in this mode is controlled by the value defined using the `-fileCount`. When this maximum number of files is reached the file logger will delete the first file and re-create it for use as the currently active file. If the value defined in the `-fileSize` is a fixed size byte unit, the upper limit on the disk space used in this mode will equal `fileSize x fileCount`. If the values defined in `-fileSize` are a time unit, the maximum size will depend on the throughput of log message in your system over these time periods.

For more information, see [MFT logger configuration properties](#)

**-fileSize *size***

Required when `-loggerType` is FILE. The maximum size that a log file is allowed to grow to. The value is a positive integer, greater than zero, followed by one of the following units: KB, MB, GB, m (minutes), h (hours), d (days), w (weeks). For example: `-fileSize 5MB` (specifies a maximum size of 5MB), `-fileSize 2d` (specifies a maximum of 2 days worth of data).

**-fileCount *number***

Required when `-loggerType` is FILE and `-fileLoggerMode` is CIRCULAR. The maximum number of log files to create. When the amount of data exceeds the maximum amount that can be stored in this number of files, the oldest file is deleted so that the number of log files never exceeds the value specified in this parameter.

**-loggerQMgrHost**

Host name, or IP address, of the machine where the logger queue manager is running.

The default value is None.

If you do not specify the **-loggerQMgrHost** parameter, the logger is created in bindings mode.

**-loggerQMgrPort**

Port number where the logger queue manager is listening.

The default value is 1414.

**-loggerQMgrChannel**

Name of the channel used for connecting to the logger queue manager.

The default value is SYSTEM.DEF.SVRCONN.

**Windows -s *service\_name***

Optional (Windows systems only). Indicates that the logger is to run as a Windows service. If you do not specify *service\_name*, the service is named `mqmftLogge\LOGGERQMGR`, where `LOGGER` is the logger name and `QMGR` is your logger queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Logger *LOGGER@QMGR***.

**Windows -su user\_name**

Optional (Windows only). When the logger is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the logger using a Windows domain user account specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Troubleshooting an MFT agent or logger running as a Windows service](#).

This parameter is required when **-s** is specified.

**Windows -sp password**

Optional (Windows only). Password for the user account set by the **-su** parameter.

This parameter is only valid when **-s** is specified. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

**Windows -sj options**

Optional (Windows only). When the logger is started as a Windows service, defines a list of options in the form of **-D** or **-X** that are passed to the JVM. The options are separated using a number sign (#) or semicolon (;) character. If you must embed any (#) or semicolon (;) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified.

**Windows -sl options**

Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

**-p configuration\_options**

Optional. Specifies the set of configuration options that is used to create the logger. By convention, this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

**-f**

Optional. Forces the command to overwrite the existing configuration.

**logger\_name**

Required. Name of the logger to create. This is incorporated into Managed File Transfer queue names, and so must contain only letters, numbers, and the periods (.) and underscore characters (\_). It is also limited to a maximum length of 28 characters.

**-credentialsFile file\_path**

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

**-userid username**

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

**-? or -h**

Optional. Displays command syntax.



## Examples

In this example, a circular file logger is created called filelogger1. The file logger will create a maximum of 10 files, each file being 10MB in size, using a maximum of 100MB of disk space in total:

```
fteCreateLogger -loggerType FILE -fileLoggerMode CIRCULAR -fileSize 10MB -fileCount 10
filelogger1
```

In this example, a database logger is created called dblogger1. The database logger connects to a Db2 database called FTEDB:

```
fteCreateLogger -loggerType DATABASE -dbName FTEDB -dbType DB2
-dbDriver "C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc4.jar" dblogger1
```

In this example, a database logger is created called dblogger1. The database logger connects to an Oracle database called FTEDB:

```
fteCreateLogger -loggerType DATABASE -dbName FTEDB -dbType oracle
-dbDriver "C:\app\oracle\product\12.1.0\dbhome_2\jdbc\lib\ojdbc7.jar" dblogger1
```

In this example, a client mode file logger is created, using the host name and default port and channel:

```
fteCreateLogger -loggerType FILE -loggerQMgr CORDQM -loggerQMgrHost cordqm.ibm.com
-fileLoggerMode CIRCULAR -fileSize 10MB -fileCount 10 FL1
```

In this example, a client mode file logger is created, using the host name, port, and channel:

```
fteCreateLogger -loggerType FILE -loggerQMgr CORDQM -loggerQMgrHost cordqm.ibm.com
-loggerQMgrPort 4444 -loggerQMgrChannel LOGGER_CHANNEL -fileLoggerMode CIRCULAR -fileSize 10MB
-fileCount 10 FL1
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related tasks

[Configuring an MFT logger](#)

## Related reference

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 2100](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

[“fteStartLogger \(start an MFT logger\)” on page 2142](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

[“fteStopLogger \(stop an MFT logger\)” on page 2148](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

[“fteDeleteLogger \(delete an MFT logger and its configuration\)” on page 2076](#)

Use the **fteDeleteLogger** command to delete a Managed File Transfer logger and its configuration. Existing log files associated with the logger can either be retained or deleted.

[MFT logger error handling and rejection](#)

[MFT logger configuration properties](#)

## fteCreateMonitor (create an MFT resource monitor)

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

### Purpose

Use the **fteCreateMonitor** command to create and then start a new resource monitor by using a Managed File Transfer agent. For example, you can use a resource monitor in the following way: An external application puts one or more files in a known directory and when processing is complete, the external application places a trigger file in a monitored directory. The trigger file is then detected and a defined file transfer starts and copies the files from the known directory to a destination agent.

You can use the **-ox** and **-ix** parameters to export and import a resource monitor configuration to an XML file. Importing this file with the **fteCreateMonitor** command creates a new resource monitor with the same parameters as the resource monitor given in the **fteCreateMonitor** command to export to the XML file. Additionally, you can use the **-f** and **-c** parameters to overwrite a monitor configuration dynamically.

### Notes:

- There is no restriction on the number of resource monitors that can be created on an agent, and all run with the same priority. Consider the implications of overlapping monitored resources, conflicting trigger conditions and how frequently the resources are polled. For more information, see [MFT resource monitoring concepts](#).
- You cannot create a resource monitor with a task definition that contains scheduled transfers. If you try to create a resource monitor with a transfer definition that points to a transfer that is scheduled to run, and repeat, at a specific time, the following message is displayed: Task definition file contains a scheduled transfer. A scheduled transfer can not be used with a resource monitor.
- The **fteCreateMonitor** command is not supported on protocol bridge agents.

**Tip:** You can also use the **fteListMonitors** command to export resource monitor configurations to an XML file:

- Using the **fteListMonitors** command with the **-ox** exports the definition for a single resource monitor.
- From IBM MQ 9.1.0, using the **fteListMonitors** command with the **-od** exports multiple resource monitor definitions to a specified directory. You can also use the **-od** option to export a single resource monitor definition to a specified directory.

For more information about the **fteListMonitors** command, see [“fteListMonitors \(list MFT resource monitors\)” on page 2089](#).

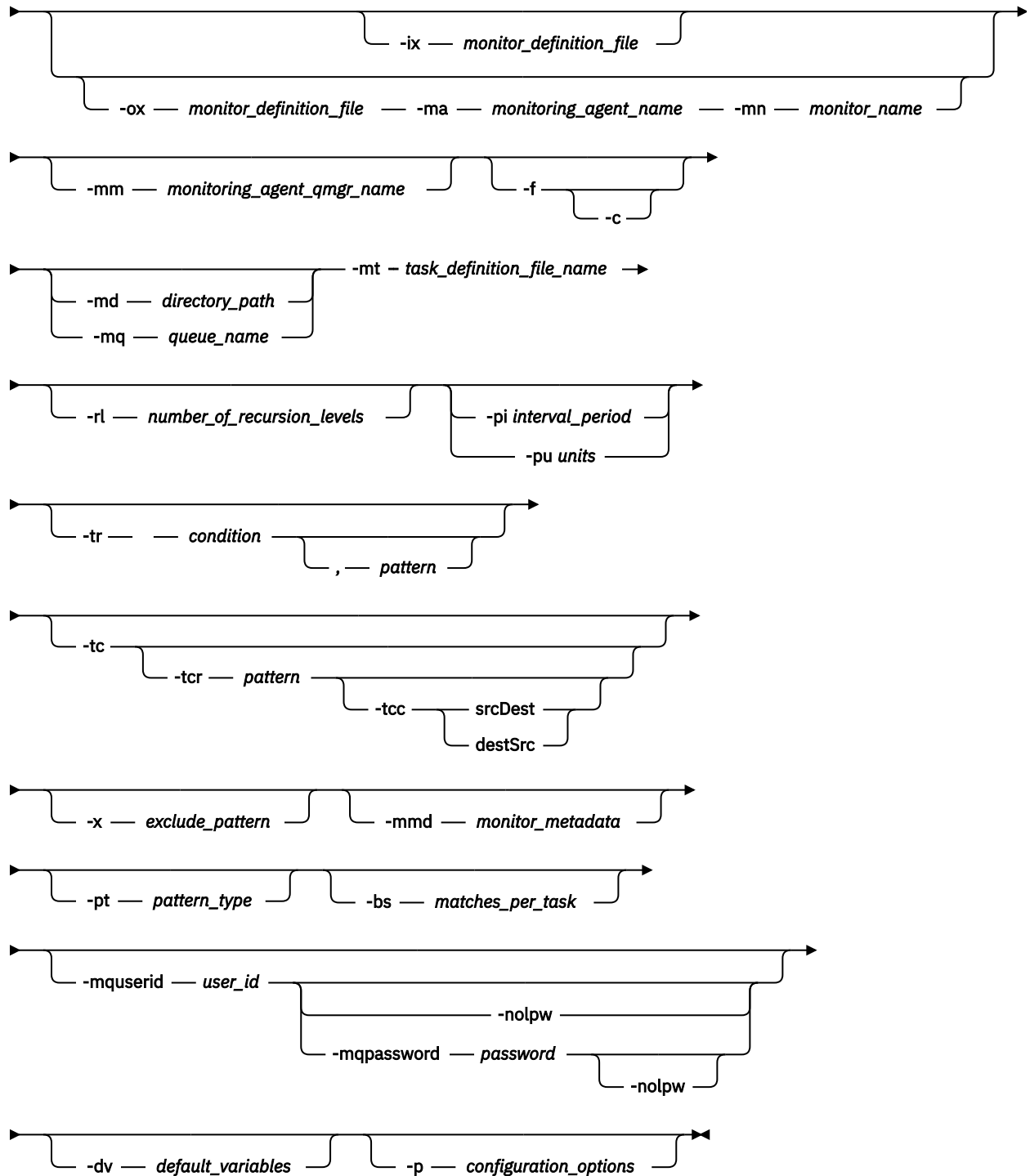
### Special characters

Take care when you use parameter values that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified file paths and names that contains such characters as space, quotation mark (single or double), backslash or forward slash characters, might be interpreted by the command shell rather than being passed through directly to the command itself. To avoid characters being interpreted by the command shell, enclose the entire parameter in double or single quotation marks or escape the special characters by using the escape sequence of the command shell. When you specify file paths on Windows, ensure the separator character backslash (\) is entered as double backslashes (\\), that is, escaped backslash (\\). Alternatively, you can use a single forward slash (/) character as a separator."

## Syntax

### fteCreateMonitor

► fteCreateMonitor ►



## Parameters

### `-ix` *xml\_filename*

Optional. Imports the resource monitor configuration from an XML file.

**-ox xml\_filename**

Optional. This parameter must be specified with the **-ma** and **-mn** parameters and can be used with the **-f** parameter. Exports the resource monitor configuration to an XML file.

**-mn monitor\_name**

Required. The name that you assign to this monitor. The monitor name must be unique to the monitoring agent. However, you can delete a monitor and then create a monitor with the same name.

The maximum length for a resource monitor name is 256 characters. Resource monitor names are not case-sensitive. Resource monitor names that are entered in lowercase or mixed case are converted to uppercase. Resource monitor names must not contain asterisk (\*), percent (%), or question mark (?) characters.

**-ma monitoring\_agent\_name**

Required. The name of the agent to perform the resource monitoring. This monitoring agent must be the source agent for the monitor task that you want to trigger.

**-mm monitoring\_agent\_qmgr\_name**

The name of the queue manager that the monitoring agent is connected to. Because the monitoring agent and the source agent must be same, this queue manager is also your source agent queue manager.

**Note:** The **fteCreateMonitor** command connects to the command queue manager for a Managed File Transfer topology. If the command queue manager is also the agent queue manager for the monitoring agent, then this parameter is optional. Otherwise, the parameter is required.

**-f**

Optional. Use this parameter to overwrite a resource monitor configuration. For example, when the resource monitor name you choose already exists on the resource monitoring agent and you want to update it rather than delete and re-create a monitor with the same name. Using this parameter causes the agent to restart the monitor process.

**-c**

Optional. This parameter clears the history of an updated resource monitor, which causes the resource monitor to check the trigger conditions again. You can use this parameter with the **-f** parameter only.

**-md directory\_path**

Optional. The absolute name of the directory path that you want to monitor. Unless you are using the **-ix** or **-ox** parameters you must specify one of the **-md** or **-mq** parameters.

**-mq queue\_name**

Optional. The name of the queue that you want to monitor. This queue must be on the monitoring agent queue manager. Unless you are using the **-ix** or **-ox** parameters you must specify one of the **-md** or **-mq** parameters.

**-mt task\_definition\_file\_name**

Required. The name of the XML document that contains the task definition that you want to carry out when the trigger condition is satisfied. For more information, see [Using transfer definition files](#). The path to the transfer definition XML document must be on the local file system that you run the **fteCreateMonitor** command from. If you do not specify a path to the file, the command looks for it in the current working directory. Unless you are using the **-ix** or **-ox** parameters, **-mt** is a required parameter.

You can use the **-gt** parameter on the [fteCreateTransfer](#) command to generate a template XML document that contains your file transfer request. The monitor uses the transfer template as its task definition.

You can also use the transfer recovery timeout, **-rt** parameter, along with the **-gt** parameter, when you run the **fteCreateMonitor** command. You can set the amount of time in seconds during which the source agent keeps retrying to recover a transfer that is stalled. The recovery timeout parameter is then included in the XML document with the transfer definition that the monitor uses. For more information on how to set this parameter, see [fteCreateTransfer](#) command.

**z/OS** On z/OS, you must store the task definition document in a UNIX file on z/OS UNIX System Services. You cannot store task definition documents in z/OS sequential files or PDS members.

**IBM i** On IBM i, you must store the task definition document in the integrated file system.

**-rl *number\_of\_recursion\_levels***

Optional. The level of monitoring recursion of the root monitoring directory that is how many levels of subdirectory to go down into. For example, in a directory structure like the following example with C:/wmqfte/monitor set as the root monitoring directory

```
C:/wmqfte/monitor
C:/wmqfte/monitor/reports
C:/wmqfte/monitor/reports/2009
C:/wmqfte/monitor/reports/2009/April
```

If you specify `-rl 2`, Managed File Transfer searches only as far down as the C:/wmqfte/monitor/reports/2009 directory and its sibling directories. The C:/wmqfte/monitor/reports/2009/April directory is ignored. By default, recursion is set to none.

**-pi *interval\_period***

Optional. The interval period between each monitor of a directory. The poll interval must be a positive integer value. The default value for `-pi` is 1.

**-pu *units***

Optional. The time units for the monitor poll interval. If you specify the `-pu` parameter, you must also specify the `-pi` parameter. The default value for `-pu` is minutes. Specify one of the following options:

**seconds**

**minutes**

**hours**

**days**

**-tr**

Optional. Specifies the trigger condition that must be satisfied for the defined task to take place. If the condition is not satisfied, according to the source agent, the monitor task (for example the file transfer) is not started. A trigger condition consists of two optional parts, condition and pattern, separated by a comma. Specify one of the following formats:

- `condition,pattern`

where *condition* is one of the following values:

**match**

For each trigger that is satisfied, the defined task is performed. `match` is the default value.

For example, if the match is `*.go` and the files `LONDON.go` and `MANCHESTER.go` are present, the task is performed for `LONDON.go` and another task is performed for `MANCHESTER.go`.

If the same trigger file is present from a previous poll (that is, the file has not been modified), this file has a not satisfied trigger condition. That is, the match trigger file must be new and must have been modified since last the poll before the defined task is performed.

**noMatch**

No files in the monitored directory match the pattern. That is, if *any* of the files in the monitored directory do not exist, the condition is satisfied. If no files match the trigger condition at the time the monitor is created, the monitor starts instantly, but does not start again until a file match is found, and then removed.

**noSizeChange=*n***

A minimum of one of the files in the directory matches the pattern and has a file size that does not change for *n* polling intervals. The value of *n* is a positive integer.

**fileSize>=size**

A minimum of one of the files in the directory matches the pattern and has a minimum file size greater or equal to *size*. The value *size* is a combination of an integer with an optional size unit of B, KB, MB, or GB. For example, `fileSize">"=10KB`. If you do not specify a size unit, the default size that is used is bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `fileSize` option on the command line, as shown in this example.

The pattern is a file pattern match sequence in wildcard or Java regular expression format. The default value for the pattern is `*`, or match any file, and the default format is wildcard format. Use the **-pt** to specify the format of the pattern.

For example, the following trigger condition is satisfied when a file exists in the monitored directory with the suffix `.go`.

```
-tr match,*.go
```

The following trigger condition is satisfied when there are no files in the monitored directory that have the suffix `.stop`.

```
-tr noMatch,*.stop
```

You can specify *condition,pattern* only if you also specify the **-md** parameter.

- *condition*

where *condition* is one of the following values:

**queueNotEmpty**

The monitored queue is not empty. That is, if there are *any* IBM MQ messages on the monitored queue, the condition is satisfied. A single task is run for all of the messages on the queue.

**completeGroups**

There is a complete group on the monitored queue. That is, if *any* of the IBM MQ message groups on the monitored queue are complete, the condition is satisfied. An individual task is run for each complete group on the queue.

If a single message that is not in a group is put on the queue, it is treated as if it is a complete group and a task is run for the single message.

You can specify *condition* only if you also specify the **-mq** parameter.

For each monitor that you create, you can specify the **-tr** parameter once only.

**-tc**

Optional. Indicates that the triggered file contains one or more file paths to generate a transfer request. The default format of the trigger file's contents is one file entry on each line. Specify the file paths either as *source file path* or *source file path,destination file path*. This parameter is available only for directory monitor triggers match and noSizeChange.

**-tcr pattern**

Optional. Specifies a replacement regular expression for parsing trigger files. If you specify the **-tcr** parameter, you must also specify the **-tc** parameter.

Design the pattern to parse each line entry completely with one or two capture groups. Group one defines the source file path and the optional group two defines the destination file path. This is the default behavior, which you can change using the **-tcc** parameter.

For more information and examples, see [Using a trigger file](#).

**-tcc**

Optional. Defines the regular expression capture group order.

**srcDest**

The default value where group one is the source file path and group two is the destination file path.

**destSrc**

The reverse of `srcDest`. Group one is the destination file path and group two is the source file path. Ensure that the regular expression for `destSrc` has two capture groups.

If you specify the **-tcc** parameter, you must also specify the **-tcx** parameter.

**-x exclude\_pattern**

Optional. Specifies files that are excluded from the trigger pattern match. The trigger pattern is specified by the **-tr** parameter.

The pattern is a file pattern match sequence in wildcard or Java regular expression format. The default format is wildcard format. Use the **-pt** parameter to specify the format of the pattern.

**-mmd monitor\_metadata**

Optional. Specifies the user-defined metadata that is passed to the monitor's exit points. The parameter can take one or more name pairs that are separated by commas. Each name pair consists of a *name=value*. You can use the **-mmd** parameter more than once in a command.

**-pt pattern\_type**

Optional. The type of pattern that is used by the **-tr** and **-x** parameters. Valid values are:

**wildcard**

The patterns are evaluated as wildcard patterns. An asterisk (\*) matches zero or more characters and a question mark (?) matches exactly one character. This is the default.

**regex**

The patterns are evaluated as Java regular expressions. For more information, see [“Regular expressions used by MFT” on page 2523](#).

**-bs matches\_per\_task**

Optional. The maximum number of trigger matches to include in a single task. For example, if a value of 5 is specified for *matches\_per\_task* and nine trigger matches occur in a single poll interval, two tasks are performed. The first task corresponds to triggers 1-5 inclusive, and the second task corresponds to triggers 6-9. The default value of *matches\_per\_task* is 1.

The **-bs** parameter is supported only when the task definition XML that you supply to the **-mt** parameter is a managedTransfer. A managedCall is not supported with the **-bs** parameter.



**-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

**-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

  From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

  **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### **-dv default\_variables**

Optional. A comma-separated list of default variables that can be used in variable substitution when monitoring a queue. The values are in the format of a key-value pair. For example:

```
-dv size=medium,color=blue
```

For more information about variable substitution, see [Customizing MFT tasks with variable substitution](#). You can only specify the **-dv** parameter if you have also specified the **-mq** parameter.

### **-? or -h**

Optional. Displays command syntax.

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a nondefault coordination queue manager as the input for this parameter. The command then uses the set of properties files that are associated with this nondefault coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

## **Examples**

In this example, a new resource monitor is created called MYMONITOR using the monitoring agent MYAGENT. Provided the trigger condition that a file larger than 5 MB is present in the directory C:/wmqfte/monitors, the file transfer that is defined in the file C:/templates/transfer\_reports.xml is started. MYAGENT is also the source agent for the file transfer that is defined in C:/templates/transfer\_reports.xml:

```
fteCreateMonitor -ma MYAGENT -md C:/wmqfte/monitors -mn MYMONITOR -mt C:/templates/transfer_reports.xml -tr fileSize">"=5MB,*.*go
```

In this example, a resource monitor called MONITOR1 using the agent AGENT1 is created to transfer files greater than 5 MB and is exported to the XML file monitor.xml.

```
fteCreateMonitor -ox monitor.xml -ma AGENT1 -mn MONITOR1 -mt task.xml -tr "fileSize>=5MB,*.*zip"
```

Then the XML file is imported and changed to exclude any files greater than 10MB.

```
fteCreateMonitor -ix monitor.xml -x "fileSize>=10MB,*.*zip" -f
```

In this example, a new resource monitor is created called MYMONITOR using the agent MYAGENT.

```
fteCreateMonitor -ma MYAGENT -md c:/wmqfte -mn MYMONITOR -mt c:/templates/transfer_reports.xml -tr "fileSize>=5MB,*.*go"
```

However the trigger is initially incorrectly set to monitor c:/wmqfte rather than c:/wmqfte/monitors. The **fteCreateMonitor** request is immediately reissued with the monitor directory corrected and the **-f** (overwrite) and **-c** (clear history) parameters used to update the monitor.

```
fteCreateMonitor -ma MYAGENT -md c:/wmqfte/monitors -mn MYMONITOR -mt c:/templates/transfer_reports.xml -tr "fileSize>=5MB,*.*go" -f -c
```

## **Return codes**

<i>Table 343. Return code names and descriptions</i>	
<b>Return code</b>	<b>Description</b>
0	Command completed successfully.
1	Command ended unsuccessfully.



## Related concepts

[Timeout option for file transfers in recovery](#)

## Related tasks

[Monitoring MFT resources](#)

[Configuring MFT monitor tasks to start commands and scripts](#)

[Customizing MFT tasks with variable substitution](#)

[Backing up and restoring MFT resource monitors](#)

## Related reference

[“fteDeleteMonitor \(delete an MFT resource monitor\)” on page 2078](#)

Use the **fteDeleteMonitor** command to stop and delete an existing Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

## fteCreateTemplate (create new file transfer template)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn *template\_name*** parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

## Purpose

Use the **fteCreateTemplate** command to create a file transfer template that stores your transfer details until you want to use them at a later date. Use transfer templates to store common file transfer settings for repeated or complex transfers. After you have created a transfer template, submit the template using the IBM MQ Explorer. You cannot submit a transfer template from the command line.

The transfer template that you create using the **fteCreateTemplate** command is not the same as the XML message that you create using the **-gt** parameter on the **fteCreateTransfer** command. You cannot use the two different types of template interchangeably.

You can run the **fteCreateTemplate** command from any system that can connect to the IBM MQ network and then route to the coordination queue manager. Specifically for the command to run, you must have installed Managed File Transfer on this system and you must have configured the Managed File Transfer component on this system to communicate with the IBM MQ network.

**V 9.3.0** From IBM MQ 9.3.0, this command uses the `coordination.properties` file to connect to the coordination queue manager for the Managed File Transfer topology. If the `coordination.properties` file contains the **coordinationQMgrHost** property, then the command connects to the coordination queue manager using the CLIENT transport. Otherwise, the command connects to the coordination queue manager using the BINDINGS transport. For more information, see [The MFT coordination.properties file](#).


You can specify multiple source files for a file transfer but only one destination agent; transferring one file to multiple destination agents is not supported. However, you can transfer multiple source files to multiple destination files on a single destination agent.

For guidance about how to transfer files, see [“Guidelines for transferring files” on page 2491](#).

## Special characters

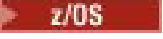
Take care when you use parameters that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. **z/OS** For example, fully qualified data set names that contain single quotation marks and source specifications that contain asterisk characters might be interpreted by the command shell rather than being passed through in the transfer request. To avoid characters being interpreted by the command shell, enclose the entire parameter in double quotation marks as shown in the final two examples [“Examples” on page 2047](#), or escape the special characters using the escape sequence of the command shell.

## Relative paths


The **fteCreateTemplate** command supports the use of relative file paths. On distributed systems  and z/OS UNIX System Services by default paths are considered to be relative to the home directory of the user that the agent is running as. To change the directory that path names are evaluated relative to, set the `transferRoot` property in the `agent.properties` file. This file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr/agents/agent_name` directory. Add the following line to the file:

```
transferRoot=directory_name
```

You must escape Windows paths or write them in UNIX format. For example, specify `C:\TransferRoot` as `C:\\TransferRoot` or `C:/TransferRoot`.

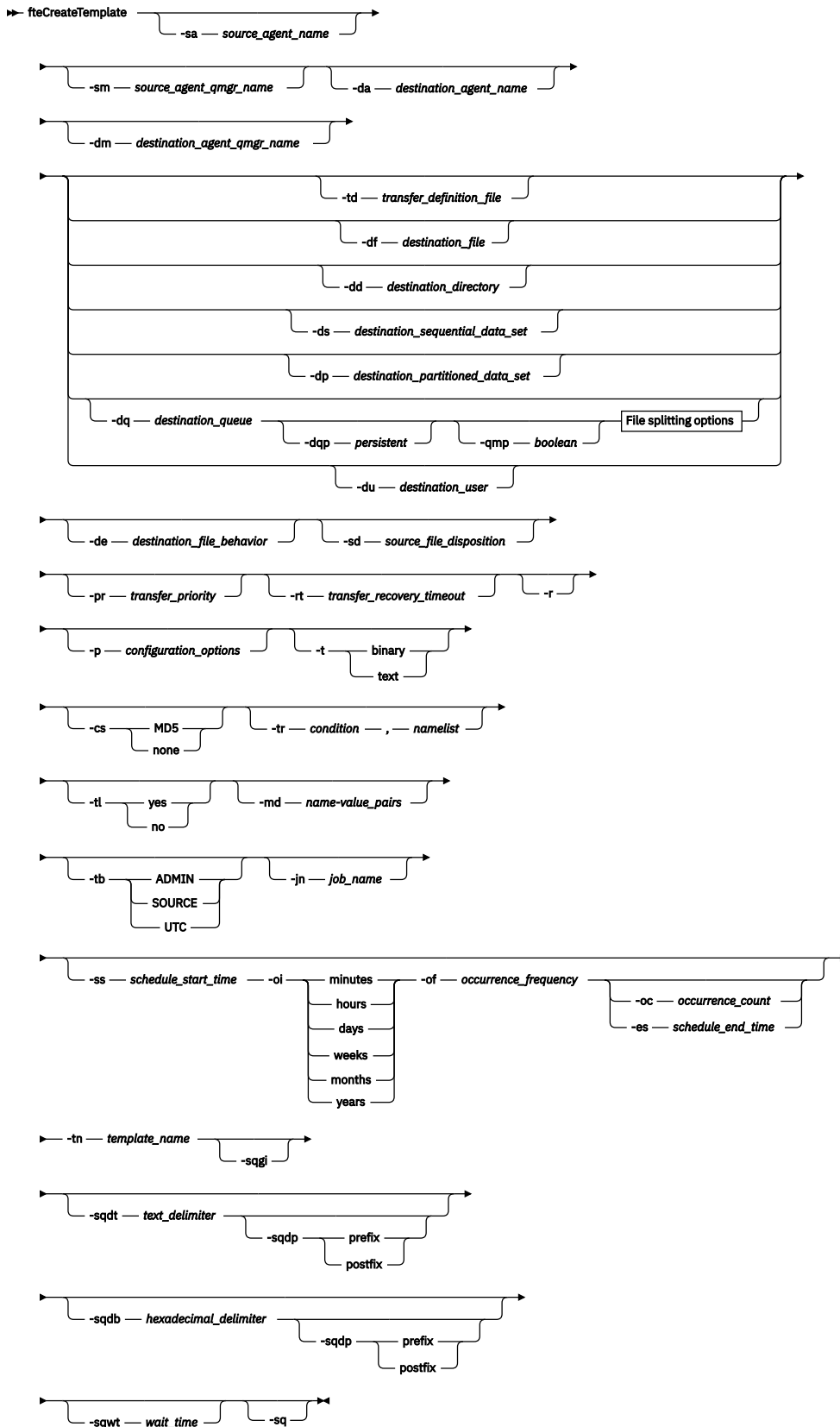
 On z/OS, by default the user name that the agent is currently running under is added as a high-level qualifier prefix to data set specifications that have not been fully qualified. For example: `//ABC.DEF`. To change the value that is added as a prefix to the data set name, set the `transferRootHLQ` property in the `agent.properties` file. This file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr/agents/agent_name` directory. Add the following line to the file:

```
transferRootHLQ=prepend_value
```

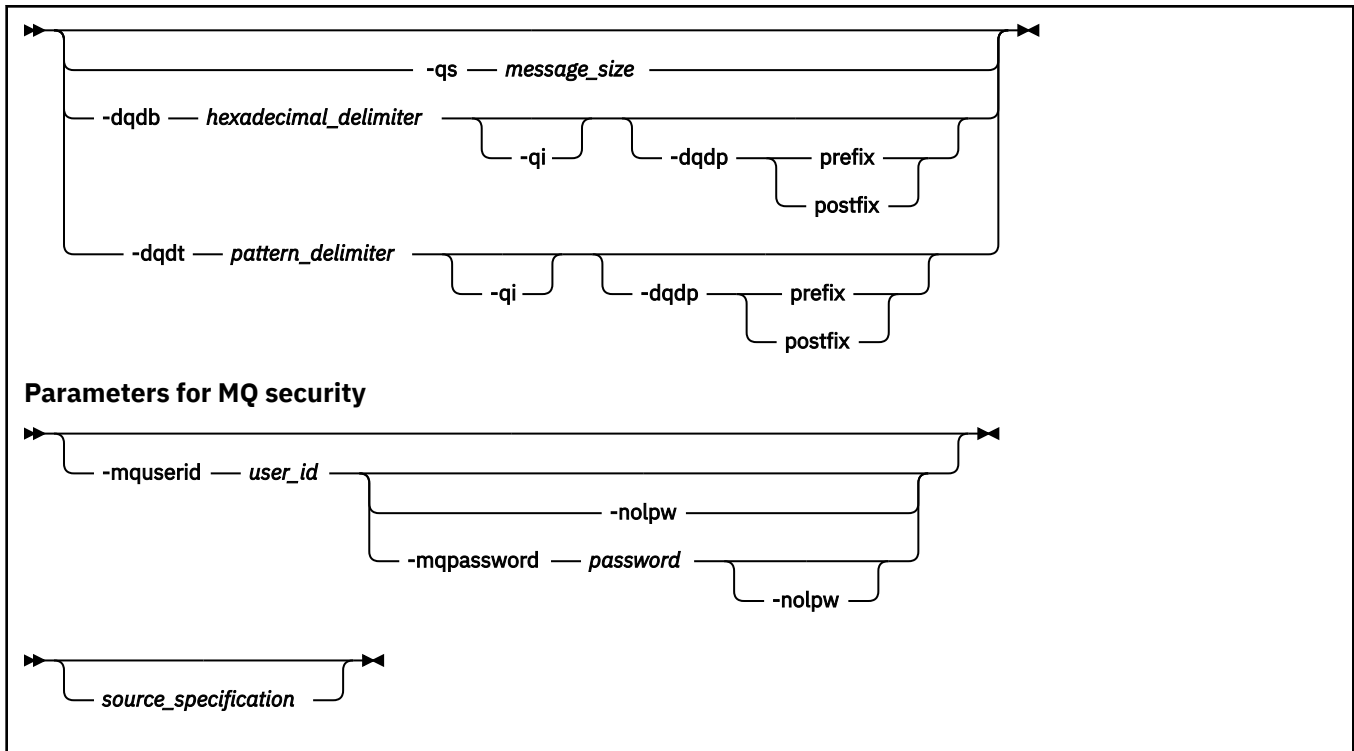
 However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name.

# Syntax

## fteCreateTemplate



### File splitting options



## Parameters

### **-sa source\_agent\_name**

Optional. The name of the agent that the source file is transferred from. If you do not specify this agent name when you create the template, you must specify the source agent name when you use the template.

### **-sm source\_agent\_qmgr\_name**

Optional. The name of the queue manager that the source agent is connected to.

If you do not specify the **-sm** parameter, the queue manager used is determined by the set of configuration options in use, based on the source agent name. If the queue manager name cannot be determined using these options, the transfer template creation fails. For example, the template creation fails if the `agent.properties` file for the source agent cannot be found.

### **-da destination\_agent\_name**

Optional. The name of the agent that the file is transferred to. If you do not specify the destination agent name when you create the template, you must specify the destination agent name when you use the template.

### **-dm destination\_agent\_qmgr\_name**

Optional. The name of the queue manager that the destination agent is connected to.

If you do not specify the **-dm** parameter, the queue manager used is determined by the set of configuration options in use, based on the destination agent name. If the queue manager name cannot be determined using these options, the transfer template creation fails. For example, the template creation fails if the `agent.properties` file for the destination agent cannot be found.

### **-td transfer\_definition\_file**

Optional. The name of the XML document that defines one or more source and destination file specifications for the transfer.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-td** parameter, you cannot specify source files, or specify the **-df**, **-dd**, **-ds**, **-dp**, **-dq**, **-du**, **-sd**, **-r**, **-de**, **-t**, or **-cs** parameters.

The **ftcCreateTemplate** command locates the transfer definition file in relation to your current directory. If you cannot use relative path notation to specify the location of the transfer definition file, use the fully qualified path and file name of the transfer definition file instead.

Alternatively, this parameter can be the name of the XML document that contains a managed transfer request. An XML task definition is created with the **-gt** parameter of the `fteCreateTransfer` command.

**z/OS** On z/OS, you must store the transfer definition file in a UNIX file on z/OS UNIX System Services. You cannot store transfer definition files in z/OS sequential files or PDS members.

**IBM i** On IBM i, you must store the transfer definition file in the integrated file system.

For more information, see [Using transfer definition files](#).

#### **-df destination\_file**

Optional. The name of the destination file. Specify a file name that is valid on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination file is specified in the format `connect_direct_node_name:file_path`. The Connect:Direct bridge agent accepts only file paths that are specified in this format. **z/OS** If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-df** parameter, you cannot specify the **-td**, **-dd**, **-dp**, **-dq**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

#### **-dd destination\_directory**

Optional. The name of the directory the file is transferred to. Specify a directory name that is valid on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination directory is specified in the format `connect_direct_node_name:directory_path`. If the destination agent is a Connect:Direct bridge agent and the destination is a PDS, you must also specify the **-de** parameter with a value of overwrite.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-dd** parameter, you cannot specify the **-td**, **-df**, **-dp**, **-dq**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

#### **z/OS -ds destination\_sequential\_data\_set**

z/OS only. Optional. The name of the sequential data set or PDS member that files are transferred into. Specify a sequential data set name or a partitioned data set member.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-ds** parameter, you cannot specify the **-td**, **-dd**, **-df**, **-dq**, **-du**, or **-dp** parameters because these parameters are mutually exclusive.

The syntax for the data set name is as follows:

```
//data_set_name{;attribute;...;attribute}
```

or

```
//pds_data_set_name(member_name){;attribute;...;attribute}
```

That is, a data set name specifier prefixed with // and optionally followed by a number of attributes separated by semicolons.


If the data set is located at a Connect:Direct node, you must prefix the data set name with the node name. For example:

```
CD_NODE1: //'OBJECT.LIB';RECFM(F,B);BLKSIZE(800);LRECL(80)
```

If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite. For more information about data set

transfers to or from Connect:Direct nodes, see [“Transferring data sets to and from Connect:Direct nodes”](#) on page 2500.

For transfers that only involve Managed File Transfer agents, if the data set name part is enclosed by single quotation mark characters, it specifies a fully qualified data set name. If the data set name is not enclosed by single quotation mark characters, the system adds the default high-level qualifier for the destination agent (either the value for the `transferRootHLQ` agent property or the user ID that the agent runs under, if you have not set `transferRootHLQ`).

**Note:**  However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name. This is the case even if the data set name is enclosed by single quotation mark characters.

The data set attributes are used either to create a data set or to ensure that an existing data set is compatible. The specification of data set attributes is in a form suitable for BPXWDYN (see [Requesting dynamic allocation](#) for more information). When the agent is to create a destination data set, the following BPXWDYN attributes are automatically specified: `DSN(data_set_name) NEW CATALOG MSG(numeric_file_descriptor)`, where *numeric\_file\_descriptor* is a file descriptor generated by Managed File Transfer. For a data set to data set transfer, the attributes of RECFM, LRECL, and BLKSIZE from the source are selected for a new destination data set. Note the SPACE setting for a new destination data set is not set by Managed File Transfer and system defaults are used. Therefore, you are recommended to specify the SPACE attribute when a new data set is to be created. You can use the **`bpxwdynAllocAdditionalProperties`** property in the agent `.properties` file to set BPXWDYN options that apply to all transfers. For more information, see [The MFT agent.properties file](#).

Some BPXWDYN options must not be specified when using the **`fteCreateTemplate`** command, the **`fteCreateTransfer`** command or the **`bpxwdynAllocAdditionalOptions`** property in the agent `.properties` file. For a list of these properties, see [BPXWDYN properties you must not use with MFT](#).

When you transfer a file or data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes passed in the transfer definition. If no attributes are specified, attributes are set to the same as the source data set or to the default values when the source is a file. The attributes of an existing tape data set are ignored.

The **`-ds`** parameter is not supported when the destination agent is a protocol bridge agent.

 **`-dp destination_partitioned_data_set`**

z/OS only. Optional. The name of the destination PDS that files are transferred into. Specify a partitioned data set name. If a PDS is created as a result of the transfer, this PDS is created as a PDSE by default. You can override the default by specifying `DSNTYPE=PDS`.

One of the **`-td`**, **`-df`**, **`-dd`**, **`-ds`**, **`-dq`**, **`-du`**, and **`-dp`** parameters is required. If you specify the **`-dp`** parameter, you cannot specify the **`-td`**, **`-dd`**, **`-df`**, **`-dq`**, **`-du`**, or **`-ds`** parameters because these parameters are mutually exclusive.

The syntax for the PDS data set name is as follows:

```
//pds_data_set_name{;attribute;..;attribute}
```

The syntax for the data set name is the same as described for the **`-ds destination_sequential_data_set`** parameter. All the syntax details for specifying data sets that are located on Connect:Direct nodes also apply to the **`-dp`** parameter. If the destination agent is a Connect:Direct bridge agent, you must also specify the **`-de`** parameter with a value of `overwrite`.

The **`-dp`** parameter is not supported when the destination agent is a protocol bridge agent.

**`-du destination_user`**

Optional. The name of the user whose destination file space the files are transferred into.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dp**, **-du**, and **-dq** parameters is required. If you specify the **-du** parameter, you cannot specify the **-td**, **-dd**, **-df**, **-dp**, **-dq**, or **-ds** parameters because these parameters are mutually exclusive.

The **-du** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent.

**-dq destination\_queue**

Optional. The name of a destination queue that files are transferred onto. You can optionally include a queue manager name in this specification, using the format QUEUE@QUEUEMANAGER. If you do not specify a queue manager name, the destination agent queue manager name is used if you have not set the enableClusterQueueInputOutput agent property to true. If you have set the enableClusterQueueInputOutput agent property to true, the destination agent uses standard IBM MQ resolution procedures to determine where the queue is located. You must specify a valid queue name that exists on the queue manager.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dp**, **-du**, and **-dq** parameters is required. If you specify the **-dq** parameter, you cannot specify the **-td**, **-dd**, **-df**, **-dp**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

The **-dq** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent, or when the source specification is a queue.

**-dqp persistent**

Optional. Specifies whether messages written to the destination queue are persistent. The valid options are as follows:

**true**

Writes persistent messages to the destination queue. This is the default value.

**false**

Writes non-persistent messages to the destination queue.

**qdef**

The persistence value is taken from the DefPersistence attribute of the destination queue.

You can only specify the **-dqp** parameter if you have also specified the **-dq** parameter.

**-qmp boolean**

Optional. Specifies whether the first message written to the destination queue by the transfer has IBM MQ message properties set. The valid options are as follows:

**true**

Sets message properties on the first message created by the transfer.

**false**

Does not set message properties on the first message created by the transfer. This is the default value.

You can only specify the **-qmp** parameter if you have also specified the **-dq** parameter. For more information, see [“MQ message properties set by MFT on messages written to destination queues” on page 2548](#)

**-qs message\_size**

Optional. Specifies whether to split the file into multiple fixed-length messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ LAST\_MSG\_IN\_GROUP flag set. The size of the messages is specified by the value of *message\_size*. The format of *message\_size* is *lengthunits*, where *length* is a positive integer value and *units* is one of the following values:

**B**

Bytes. The minimum value allowed is two times the maximum bytes-per-character value of the code page of the destination messages.

**K**

This is equivalent to 1024 bytes.

## M

This is equivalent to 1048576 bytes.

If you specify the value `text` for the **-t** parameter and the file is in a double byte character set or multibyte character set, the file is split into messages on the closest character boundary to the specified message size.

You can only specify the **-qs** parameter if you have also specified the **-dq** parameter. You can only specify one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

### **-dqdb hexadecimal\_delimiter**

Optional. Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The format for specifying a hexadecimal byte as a delimiter is `xNN`, where `N` is a character in the range `0-9` or `a-f`. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes, for example: `x3e,x20,x20,xbf`.

You can only specify the **-dqdb** parameter if you have also specified the **-dq** parameter and the transfer is in binary mode. You can only specify one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

### **-dqdt pattern**

Optional. Specifies the regular expression to use when splitting a text file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The format for specifying a regular expression as a delimiter is a regular expression enclosed in parentheses, (*regular\_expression*). The value of this parameter is evaluated as a Java regular expression. For more information, see [“Regular expressions used by MFT” on page 2523](#).

By default, the length of the string that the regular expression can match is limited by the destination agent to five characters. You can change this behavior using the `maxDelimiterMatchLength` agent property. For more information, see [Advanced agent properties](#).

You can only specify the **-dqdt** parameter if you have also specified the **-dq** parameter and the value `text` for the **-t** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

### **-dqdp**

Optional. Specifies the expected position of destination text and binary delimiters when splitting files. You can only specify the **-dqdp** parameter if you have also specified one of the **-dqdt** and **-dqdb** parameters.

Specify one of the following options:

#### **prefix**

The delimiters are expected at the beginning of each line.

#### **postfix**

The delimiters are expected at the end of each line. This is the default option.

### **-qi**

Optional. Specifies whether to include the delimiter that is used to split the file into multiple messages in the messages. If **-qi** is specified, the delimiter is included at the end of the message that contains the file data preceding the delimiter. By default the delimiter is not included in the messages.

You can only specify the **-qi** parameter if you have also specified one of the **-dqdt** and **-dqdb** parameters.

### **-de destination\_file\_behavior**

Optional. Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:

#### **error**

Reports an error and the file is not transferred. This is the default value.

#### **overwrite**

Overwrites the existing destination file.



If you specify the **-de** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive.

### **-sd source\_file\_disposition**


Optional. Specifies the action that is taken on a source file when that source file has successfully been transferred to its destination. The valid options are as follows:

#### **leave**

The source files are left unchanged. This is the default value.

#### **delete**

The source file is deleted from the source system after the source file is successfully transferred.

 On z/OS, if the source is a tape data set and you specify the delete option, the tape is remounted to delete the data set. This behavior is because of the behavior of the system environment.

If the source is a queue and you specify the leave option, the command returns an error and a transfer is not requested.

If the source agent is a Connect:Direct bridge agent and you specify the delete option, the behavior is different to the usual source disposition behavior. One of the following cases occurs:

- If Connect:Direct uses a process that is generated by Managed File Transfer to move the file or data set from the source, specifying the delete option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see [Submitting a user-defined Connect:Direct process from a file transfer request](#).
- If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

If you specify the **-sd** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify source disposition behavior in the transfer definition file.

### **-pr transfer\_priority**

Optional. Specifies the priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

This value matches the message priority value used by IBM MQ, see [Getting messages from a queue: priority](#) for more information. Message traffic for file transfer data defaults to a priority level of 0, which allows your IBM MQ message traffic to take priority.

### **-rt transfer\_recovery\_timeout**

Optional. Sets the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. Specify one of the following options:

#### **-1**

The agent continues to attempt to recover the stalled transfer until the transfer is complete. Using this option is the equivalent of the default behavior of the agent when the property is not set.

#### **0**

The agent stops the file transfer as soon as it enters recovery.

#### **>0**

The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. For example,

```
-rt 21600
```

indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. The maximum value for this parameter is 999999999.

Specifying the transfer recovery timeout value in this way sets it on a per transfer basis. To set a global value for all transfers in a Managed File Transfer network, you can add a [transferRecoveryTimeout](#) property to the `agent.properties` file.

**-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to create the transfer template. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

**-r**

Optional. Recursively transfer files in subdirectories when *source\_specification* contains wildcard characters. When Managed File Transfer is presented with a wildcard character as a *source\_specification*, any directories that match the wildcard character are transferred only if you have specified the **-r** parameter. When *source\_specification* matches a subdirectory, all files in that directory and its subdirectories (including hidden files) are always transferred.

For more information about how Managed File Transfer handles wildcard characters, see [“Using wildcard characters with MFT”](#) on page 2518

If you specify the **-r** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify recursive behavior in the transfer definition file.

**-t**


Optional. Specifies the type of file transfer: binary mode or text mode.


**binary**

The data in the file is transferred without any conversion. This is the default value.

**text**

The code page and end-of-line characters of the file are converted. The exact conversions performed depend on the operating systems of the source agent and destination agent.

 For example, a file transferred from Windows to z/OS has its code page converted from ASCII to EBCDIC. When a file is converted from ASCII to EBCDIC, the end-of-line characters are converted from ASCII carriage return (CR) and line feed (LF) character pairs to an EBCDIC new line (NL) character.

 For more information about how z/OS data sets are transferred, see [“Transferring files and data sets between z/OS and distributed systems”](#) on page 2492 and [“Transferring between data sets on z/OS”](#) on page 2493.

If you specify the **-t** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify transfer mode behavior in the transfer definition file.

**-cs**

Optional. Specifies whether a checksum algorithm is run on the file transfer data to check the integrity of the transferred files. Specify one of the following options:

**MD5**

Computes an MD5 checksum for the data. The resulting checksum for the source and destination files is written to the transfer log for validation purposes. By default, Managed File Transfer computes MD5 checksums for all file transfers.

**none**

No MD5 checksum is computed for the file transfer data. The transfer log records that checksum was set to none and the value for the checksum is blank. For example:

```
<checksum method="none"></checksum>
```

If you use the none option, you might improve file transfer performance, depending on your environment. However, selecting this option means that there is no validation of the source or destination files.

If you specify the **-cs** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify checksum behavior in the transfer definition file.

#### **-tr**

Optional. Specifies a condition that must be true for this file transfer to take place. If the condition is not true, according to the source agent, the file transfer is discarded and no transfer takes place. Specify the following format:

```
condition, namelist
```

where *condition* is one of the following values:

##### **file=exist**

A minimum of one of the files in the namelist exists. That is, if *any* of the files in the namelist exists, the condition is true.

##### **file!=exist**

A minimum of one of the files in the namelist does not exist. That is, if *any* of the files in the namelist do not exist, the condition is true.

##### **filesize>=size**

A minimum of one of the files in the namelist exists and has a minimum size as specified by *size*. The value of *size* is an integer with an optional size unit of KB, MB, or GB. For example, `filesize">="=10KB`. If you do not specify a size unit, the size is assumed to be bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `filesize` option on the command line, as shown in this example.

And where *namelist* is a comma-separated list of file names located on the source system. Depending on your operating system, if you want to use path names or file names in a namelist that contain spaces, you might have to enclose the path names and file names in double quotation marks. You can specify more than one trigger condition by using the **-tr** parameter more than once. However in that case, every separate trigger condition must be true for the file transfer to take place.

**Note:** To continually monitor a resource for a trigger condition to be true, you are recommended to use [resource monitoring](#). You can create a resource monitor using the `fteCreateMonitor` command.

In the following example, the file `file1.doc` is transferred from AGENT1 to AGENT2, on condition that either file `A.txt`, or file `B.txt`, or both files exist on AGENT1 *and* that either file `A.txt`, or file `B.txt`, or both files are equal to or larger than 1 GB:

```
fteCreateTemplate -tn JUPITER_AGENT_TRIGGER_TEST_TEMPLATE -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tr file=exist,C:\export\A.txt,C:\export\B.txt
-tr filesize">="=1GB,C:\export\A.txt,C:\export\B.txt
-df C:\import\file1.doc C:\export\file1.doc
```

You can combine triggering parameters with scheduling parameters. If you do specify both types of parameters, the trigger conditions are applied to the file transfer created by the scheduling parameters.

#### **-tl**

Optional. Specifies whether trigger failures are logged. Specify one of the following options:

##### **yes**

Log entries are created for failed triggered transfers. This is the default behavior even if you do not specify the **-tl** parameter.

##### **no**

No log entries are created for failed triggered transfers.

#### **-md**

Optional. Specifies the user-defined metadata that is passed to the exit points of the agent. The **-md** parameter can take one or more name-value pairs separated by commas. Each name pair consists of *name=value*. You can use the **-md** parameter more than once in a command.

**z/OS** On z/OS, spaces represent delimiters so you must use underscores to separate values. For example, use `kw=text1_text2_text3` rather than `kw="text1 text2 text3"`

**-tb**

Optional. Specifies the time base you want to use for the scheduled file transfer. That is, whether you want to use a system time or Coordinated Universal Time (UTC). You must use this parameter with the **-ss** parameter only. Specify one of the following options:

**admin**

The start and end times used for the scheduled transfer are based on the time and date of the system used by the administrator. This is the default value.

**source**

The start and end times used for the scheduled transfer are based on the time and date of the system where the source agent is located.

**UTC**

The start and end times used for the scheduled transfer are based on Coordinated Universal Time (UTC).

**-jn *job\_name***

Optional. A user-defined job name identifier that is added to the log message when the transfer has started.

**-ss *schedule\_start\_time***

Optional. Specifies the time and date that you want the scheduled transfer to take place. Use one of the following formats to specify the time and date. Specify the time using the 24-hour clock:

```
yyyy-MM-ddThh:mm  
hh:mm
```

Scheduled file transfers start within a minute of the schedule start time, if there are no problems that might affect the transfer. For example, there might be issues with your network or agent that prevent the scheduled transfer starting.

**-oi**

Optional. Specifies the interval that the scheduled transfer occurs at. You must use this parameter with the **-ss** parameter only. Specify one of the following options:

**minutes**

**hours**

**days**

**weeks**

**months**

**years**

**-of *occurrence\_frequency***

Optional. Specifies the frequency that the scheduled transfer occurs at. For example, every **5** weeks or every **2** months. You must specify this parameter with the **-oi** and **-ss** parameters only. If you do not specify this parameter, a default value of 1 is used.

**-oc *occurrence\_count***

Optional. Specifies how many times you want this scheduled transfer to occur. After the occurrence count has been met, the scheduled transfer is deleted.

Specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-oc** parameter, you cannot specify the **-es** parameter because these parameters are mutually exclusive.

You can omit both the **-oc** and **-es** parameters to create a transfer that repeats indefinitely.

**-es *schedule\_end\_time***

Optional. The time and date that a repeating scheduled transfer ends.

You must specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-es** parameter, you cannot specify the **-oc** parameter because these parameters are mutually exclusive.

You can omit both the **-es** and **-oc** parameters to create a transfer that repeats indefinitely.

Use one of the following formats to specify the end time and date. Specify the time using the 24-hour clock:

```
yyyy-MM-ddThh:mm
```

```
hh:mm
```

**-tn *template\_name***

Required. The name of the template that you want to create. Use a descriptive string that allows you to select the correct template for transfers at a later date. There is no specific limit to the length of this string, but be aware that excessively long names might not be displayed properly in some user interfaces.

Do not create multiple templates with the same name.

**-sqgi**

Optional. Specifies that the messages are grouped by IBM MQ group ID. The first complete group is written to the destination file. If this parameter is not specified, all messages on the source queue are written to the destination file.

You can only specify the **-sqgi** parameter if you have also specified the **-sq** parameter.

**-sqdt *text\_delimiter***

Optional. Specifies a sequence of text to insert as the delimiter when appending multiple messages to a text file. You can include Java escape sequences for String literals in the delimiter. For example, `-sqdt \u007d\n`.

You can only specify the **-sqdt** parameter if you have also specified the **-sq** parameter and the value `text` for the **-t** parameter.

**-sqdb *hexadecimal\_delimiter***

Optional. Specifies one or more byte values to insert as the delimiter when appending multiple messages to a binary file. Each value must be specified as two hexadecimal digits in the range 00-FF, prefixed by `x`. Multiple bytes must be comma-separated. For example, `-sqdb x08,xA4`.

You can only specify the **-sqdb** parameter if you have also specified the **-sq** parameter. You cannot specify the **-sqdb** parameter if you have also specified the value `text` for the **-t** parameter.

**-sqdp**

Optional. Specifies the position of insertion of source text and binary delimiters. You can only specify the **-sqdp** parameter if you have also specified one of the **-sqdt** and **-sqdb** parameters.

Specify one of the following options:

**prefix**

The delimiters are inserted at the start of each message

**postfix**

The delimiters are inserted at the end of each message. This is the default option.

**-sqwt *wait\_time***

Optional. Specifies the time, in seconds, to wait for one of the following conditions to be met:

- For a new message to be put on the queue
- If the **-sqgi** parameter was specified, for a complete group to be put on the queue

If neither of these conditions are met within the time specified by *wait\_time*, the source agent stops reading from the queue and completes the transfer. If the **-sqwt** parameter is not specified, the source agent stops reading from the source queue immediately if the source queue is empty or, in the case where the **-sqgi** parameter is specified, if there is no complete group on the queue.

You can only specify the **-sqwt** parameter if you have also specified the **-sq** parameter.

**-sq**

Optional. Specifies that the source of a transfer is a queue.



**-mquserid *user\_id***

Optional. Specifies the user ID to authenticate with the coordination queue manager.

**-mqpassword *password***

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

  From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

  **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**source\_specification**

Required if you have specified one of the **-df**, **-dd**, **-dp**, **-dp**, or **-ds** parameters. If you specify the **-td** parameter, do not specify *source\_specification*.

- If you have not specified the **-sq** parameter, *source\_specification* is one or more file specifications that determine the source, or sources, for the file transfer. File specifications are space delimited. File specifications can take one of five forms and can include wildcard characters. For more information about wildcard characters in WMQFTE, see [“Using wildcard characters with MFT” on page 2518](#). You can escape asterisks that are part of the file specification by using two asterisk characters (\*\*) in the file specification.

To transfer files containing spaces in their file names, place double quotation marks around the file names that contain spaces. For example to transfer file a b.txt to file c d.txt specify the following text as part of the **fteCreateTemplate** command:

```
-df "c d.txt" "a b.txt"
```

Each file specification must be in one of the following formats:

**File names**

The name of a file, expressed using the appropriate notation for the system where the source agent is running. When a file name is specified as a source file specification, the contents of the file are copied.

**Directories**

The name of a directory, expressed using the appropriate notation for the system where the source agent is running. When a directory is specified as a source file specification, the contents of the directory are copied. More precisely, all files in the directory and in all its subdirectories, including hidden files, are copied.

For example, to copy the contents of DIR1 to DIR2 only, specify DIR1/\* DIR2

 **Sequential data set**

(z/OS only). The name of a sequential data set or partitioned data set member. Denote data sets by preceding the data set name with two forward slash characters (//).

## Partitioned data set

(z/OS only). The name of a partitioned data set. Denote data set names by preceding the data set name with two forward slash characters (//).

### File name or directory at a Connect:Direct node

(Connect:Direct bridge agent only). The name of a Connect:Direct node, a colon character (:), and a file or directory path on the system that is hosting the Connect:Direct node. For example, `connect_direct_node_name:file_path`.

If the source agent is a Connect:Direct bridge agent, it will only accept source specifications in this form.

**Note:** Wildcard characters are not supported in file paths when the source agent is a Connect:Direct bridge agent.

- If you have specified the **-sq** parameter, *source\_specification* is the name of a local queue on the source agent queue manager. You can specify only one source queue. The source queue is specified in the format:

```
QUEUE_NAME
```

The queue manager name is not included in the source queue specification, because the queue manager must be the same as the source agent queue manager.

### -? or -h

Optional. Displays command syntax.

### Examples

In this example, a transfer template called `payroll accounts monthly report template` is created. When submitted, this template transfers any file with the extension `.xls` from the agent `PAYROLL1` to the agent `ACCOUNTS` in the directories specified:


```
fteCreateTemplate -tn "payroll accounts monthly report template" -sa PAYROLL -sm QM_PAYROLL1 -da ACCOUNTS -dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

In this example, a transfer template called `jupiter_neptune_sched_template` is created. When submitted, the template transfers the file `originalfile.txt` from the system where `QM_JUPITER` is located to the system where `QM_NEPTUNE` is located. The file transfer is scheduled to take place at 09:00 based on the system time of the system where the source agent is located and occurs every two hours four times:

```
fteCreateTemplate -tn jupiter_neptune_sched_template -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE -tb source -ss 09:00 -oi hours -of 2 -oc 4 -df C:\import\transferredfile.txt C:\export\originalfile.txt
```


In this example, a transfer template called `jupiter neptune trigger template` is created. When the template is submitted, the file `originalfile.txt` is transferred from `AGENT1` to `AGENT2`, on condition that the file `A.txt` exists on `AGENT1`:

```
fteCreateTemplate -tn "jupiter neptune trigger template" -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE -tr file=exist,C:\export\A.txt -df C:\import\transferredfile.txt C:\export\originalfile.txt
```

 In this example, a template called `ascii_ebcdic_template` is created. When the template is submitted, the file `originalfile.txt` is transferred from the system where `AGENT1` is

located to a data set //'USERID.TRANS.FILE.TXT' on the system where AGENT2 is located. Text mode has been selected to convert data from ASCII to EBCDIC.

```
fteCreateTemplate -tn ascii_ebcidic_template -t text -sa AGENT1 -da AGENT2
-ds "//TRANS.FILE.TXT;RECFM(V,B);BLKSIZE(6144);LRECL(1028);
SPACE(5,1)" C:\export\originalfile.txt
```

 In this example, a template called `ebcidic_ascii_template` is created. When the template is submitted, a member of a fully qualified data set on the system where AGENT1 is located is transferred to a file on the system where AGENT2 is located. Text mode has been selected to convert the file from EBCDIC to ASCII.

```
fteCreateTemplate -tn ebcidic_ascii_template -t text -sa AGENT1 -da AGENT2 -df /tmp/IEEUJV.txt
"//'SYS1.SAMPLIB(IEEUJV)'"
```

## Return codes

Return code	Description
0	Command completed successfully.
1	Command ended unsuccessfully.

### Related concepts

[Working with file transfer templates](#)

[Timeout option for file transfers in recovery](#)

### Related tasks

[Creating a file transfer template using IBM MQ Explorer](#)

[Backing up a file transfer template definition](#)

### Related reference

[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[“fteListTemplates \(list available MFT transfer templates\)” on page 2095](#)

Use the **fteListTemplates** command to list the available Managed File Transfer transfer templates on a coordination queue manager.

[“fteDeleteTemplates \(delete an MFT template\)” on page 2081](#)

Use the **fteDeleteTemplates** command to delete an existing Managed File Transfer template from a coordination queue manager.

## fteCreateTransfer (start a new file transfer)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

### Purpose

Use the **fteCreateTransfer** command to create and then start a new file transfer from a Managed File Transfer agent.

**Note:** File transfers can only take place between agents within the same Managed File Transfer topology.

For guidance about how to transfer files, see [“Guidelines for transferring files” on page 2491](#). For the z/OS platform, you can transfer text files, data sets, and generation data groups (GDGs).



You can run the **fteCreateTransfer** command from any system that can connect to the IBM MQ network and then route to the source agent queue manager. Specifically, for the command to run, you must install a Managed File Transfer component (either Service or Agent) on this system and configure the Managed File Transfer component on this system to communicate with the IBM MQ network.

This command uses a properties file called `command.properties` to connect to the IBM MQ network. If the `command.properties` file does not contain property information, a bindings mode connection is made to the default queue manager on the local system. If the `command.properties` file does not exist, an error is generated. For more information, see [The MFT command.properties file](#).

You can specify multiple source files for a file transfer but they must originate from a single source agent and terminate at a single destination agent. Transferring a single source file to multiple destination files on the same agent or multiple different agents is not supported within a single transfer. Ant scripting can be used to send the same source file to multiple destinations at one or more agents. For more information, see [Using Apache Ant with MFT](#).

## Special characters

Take care when you use parameters that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified data set names that contain single quotation marks, and source specifications that contain asterisk characters, might be interpreted by the command shell rather than being passed through in the transfer request. To avoid characters being interpreted by the command shell, enclose the entire parameter in double quotation marks or escape the special characters by using the escape sequence of the command shell.

## Relative paths

The **fteCreateTransfer** command supports the use of relative file paths. For the following platforms, by default, paths are considered to be relative to the home directory of the user that the agent is running as:

- **Multi** Multiplatforms
- **z/OS** z/OS UNIX System Services

To change the directory that path names are evaluated relative to, set the `transferRoot` property in the `agent.properties` file. This file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr/agents/agent_name` directory. Add the following line to the file:

```
transferRoot=directory_name
```

**Windows** For example, specify `C:\TransferRoot` as `C:\\TransferRoot` or `C:/TransferRoot`.

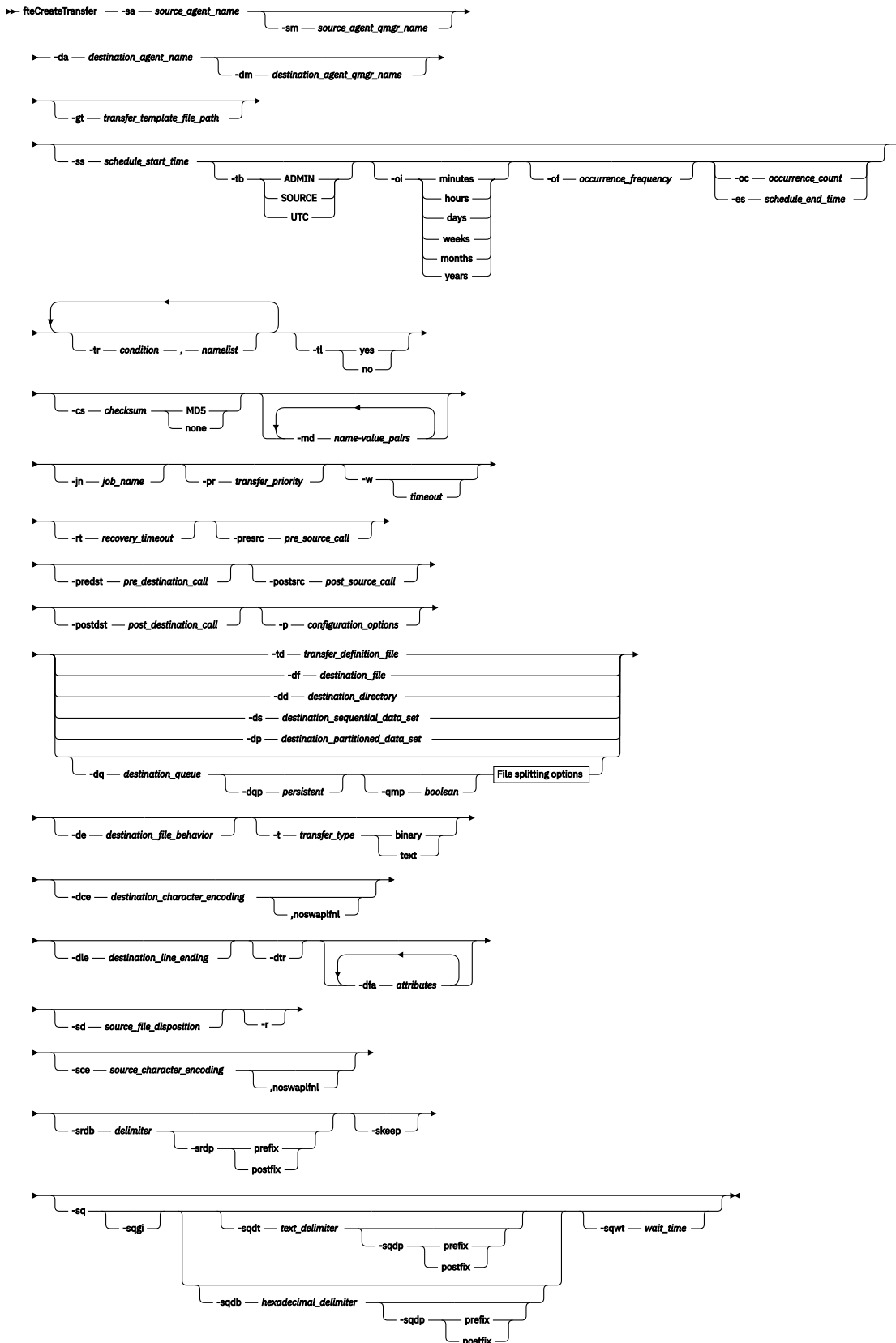
**z/OS** On z/OS, by default the user name that the agent is running under is added as a high-level qualifier prefix to data set specifications that have not been fully qualified. For example: `//ABC.DEF`. To change the value that is added as a prefix to the data set name, set the `transferRootHLQ` property in the `agent.properties` file. This file is located in the `MQ_DATA_PATH/mqft/config/coordination_qmgr/agents/agent_name` directory. Add the following line to the file:

```
transferRootHLQ=prepend_value
```

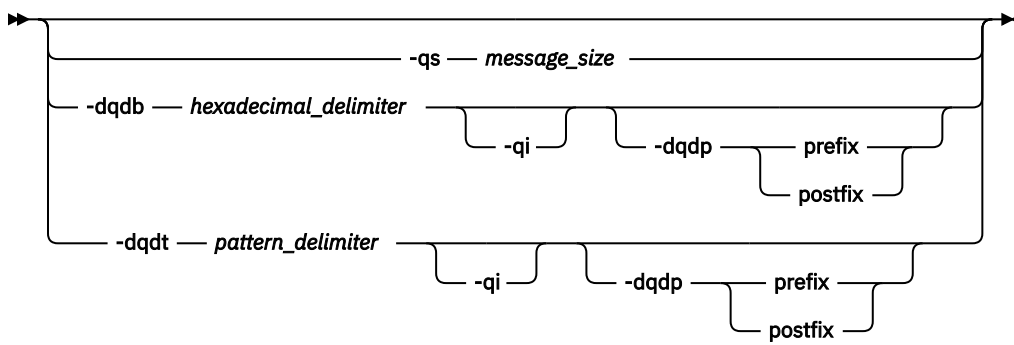
**z/OS** However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name.

# Syntax

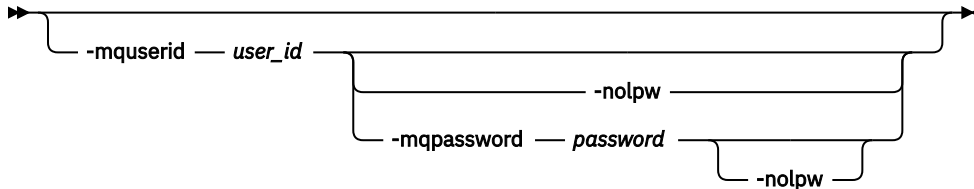
## fteCreateTransfer



### File splitting options



### Parameters for MQ security



► source\_specification ◄

## Parameters for agent specification

### **-sa source\_agent\_name**

Required. The name of the agent that the source files are transferred from.

**z/OS** If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

If you specify the **-td** parameter and the transfer definition file contains the source agent that you want to use for the transfer, do not specify the **-sa** parameter.

### **-sm source\_agent\_qmgr\_name**

Optional. The name of the queue manager that the source agent is connected to.

If you do not specify the **-sm** parameter, the queue manager that is used is determined by the set of configuration options in use, which is based on the source agent name. If the `agent.properties` file for the source agent cannot be found, the file transfer fails.

### **-da destination\_agent\_name**

Required. The name of the agent that the files are transferred to.

If you specify the **-td** parameter and the transfer definition file contains the destination agent that you want to use for the transfer, do not specify the **-da** parameter.

### **-dm destination\_agent\_qmgr\_name**

Optional. The name of the queue manager that the destination agent is connected to.

If you do not specify the **-dm** parameter, the queue manager that is used is determined by the set of configuration options in use, which is based on the destination agent name. If the `agent.properties` file for the destination agent cannot be found, the file transfer fails.

## Parameters for generating transfer templates

### **-gt transfer\_template\_file\_path**

Optional. Generates a transfer template XML message and writes this message to a file. If you specify this parameter, no transfer request is sent to Managed File Transfer. Instead, the contents of the transfer request message are written to the named XML document. You can then use this XML document to define the task for resource monitoring. See [fteCreateMonitor command](#) for information

about how to create a resource monitor. If you do not specify this parameter, the default behavior takes place and an actual transfer request is carried out.

You must provide the full path and name of an XML output file as input for this parameter, for example `C:\templates\transfer_reports.xml`

**z/OS** On z/OS, you must store the transfer template document in a UNIX file on z/OS UNIX System Services. You cannot store transfer template documents in z/OS sequential files or PDS members.

**IBM i** On IBM i, you must store the transfer template document in the integrated file system.

The transfer template XML message that you create by using the **-gt** parameter is not the same as the transfer you create by using the **fteCreateTemplate** command, which means you cannot use the two different types of template interchangeably.

**Note:** If you want to generate a transfer template XML document by running the **fteCreateTransfer** command with the **-gt** parameter, and then provide that transfer template XML document as input to the **fteCreateTransfer** command using the **-td** parameter, you must ensure that the transfer template XML document was generated specifying those parameters that are mutually exclusive with the **-td** option.

The parameters mutually exclusive to the **-td** option are:

- **-dd** *destination\_directory*
- *Source path*
- **-df** *destination\_file*
- **-cs** *checksum*
- **-de** *destination\_file\_behavior*
- **-dq** *destination\_queue*
- **-t** *transfer\_type*
- **-sd** *source\_file\_disposition*

For example, it is not possible to specify both the **-td** and **-t** parameters (indicating whether the transfer is a binary or text transfer) on the **fteCreateTransfer** command. This means that if you want to pass in a transfer template XML document to the command and specify that the transfer should be a text transfer, you should create the XML document by specifying the **-gt** and **-t** text parameters.

This parameter is not supported in the REST API.

## Parameters for scheduling transfers

### **-ss** *schedule\_start\_time*

Optional. Specifies the time and date that you want the scheduled transfer to take place. Use one of the following formats to specify the time and date. Specify the time by using the 24-hour clock:

```
yyyy-MM-ddThh:mm  
hh:mm
```

Scheduled file transfers start within a minute of the schedule start time, if there are no problems that might affect the transfer. For example, there might be issues with your network or agent that prevent the scheduled transfer starting.

### **-tb**

Optional. Specifies the time base you want to use for the scheduled file transfer. That is, whether you want to use a system time or Coordinated Universal Time (UTC). You must use this parameter with the **-ss** parameter only. Specify one of the following options:

**admin**

The start and end times used for the scheduled transfer are based on the time and date of the system used by the local administrator. This is the default value.

**source**

The start and end times used for the scheduled transfer are based on the time and date of the system where the source agent is located.

**UTC**

The start and end times used for the scheduled transfer are based on Coordinated Universal Time (UTC).

**-oi**

Optional. Specifies the interval that the scheduled transfer occurs at. You must use this parameter with the **-ss** parameter only. Specify one of the following options:

**minutes**

**hours**

**days**

**weeks**

**months**

**years**

**-of occurrence\_frequency**

Optional. Specifies the frequency that the scheduled transfer occurs at. For example, every **5** weeks or every **2** months. You must specify this parameter with the **-oi** and **-ss** parameters only. If you do not specify this parameter, a default value of 1 is used.

**-oc occurrence\_count**

Optional. Specifies how many times you want this scheduled transfer to occur. After the occurrence count is met, the scheduled transfer is deleted.

Specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-oc** parameter, you cannot specify the **-es** parameter because these parameters are mutually exclusive.

You can omit both the **-oc** and **-es** parameters to create a transfer that repeats indefinitely.

**-es schedule\_end\_time**

Optional. The time and date that a repeating scheduled transfer ends.

You must specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-es** parameter, you cannot specify the **-oc** parameter because these parameters are mutually exclusive.

You can omit both the **-es** and **-oc** parameters to create a transfer that repeats indefinitely.

Use one of the following formats to specify the end time and date. Specify the time by using the 24-hour clock:

```
yyyy-MM-ddThh:mm
```

```
hh:mm
```

## Parameters for triggering transfers

**-tr**

Optional. Specifies a condition that must be true for this file transfer to take place. If the condition is not true, according to the source agent, the file transfer is discarded and no transfer takes place. Specify the following format:

*condition,namelist*

where *condition* is one of the following values:

**file=exist**

A minimum of one of the files in the namelist exists. That is, if *any* of the files in the namelist exists, the condition is true.

**file!=exist**

A minimum of one of the files in the namelist does not exist. That is, if *any* of the files in the namelist do not exist, the condition is true.

**filesize>=size**

A minimum of one of the files in the namelist exists and has a minimum size as specified by *size*. *size* is an integer with an optional size unit of KB, MB, or GB. For example, `filesize">"=10KB`. If you do not specify a size unit, the size is assumed to be bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `filesize` option on the command line, as shown in this example.

And where *namelist* is a comma-separated list of file names located on the same system as the source agent. Depending on your operating system, if you want to use path names or file names in a namelist that contain spaces, you might have to enclose the path names and file names in double quotation marks.

You can specify more than one trigger condition by using the **-tr** parameter more than once. However in that case, every separate trigger condition must be true for the file transfer to take place.

**Note:** To continually monitor a resource for a trigger condition to be true, you are strongly recommended to use [resource monitoring](#). You can create a resource monitor by using the `fteCreateMonitor` command.

In the following example, the file `file1.doc` is transferred from AGENT1 to AGENT2, on condition that either file `A.txt`, or file `B.txt`, or both files exist on AGENT1 *and* that either file `A.txt`, or file `B.txt`, or both files are equal to or larger than 1 GB:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tr file=exist,C:/export/A.txt,C:/export/B.txt
-tr filesize">"=1GB,C:/export/A.txt,C:/export/B.txt
-df C:/import/file1.doc C:/export/file1.doc
```

You can combine triggering parameters with scheduling parameters. If you do specify both types of parameters, the trigger conditions are applied to the file transfer created by the scheduling parameters.

The **-tr** parameter is not supported on protocol bridge agents, or in the CreateTransfer REST API.

**-tl**

Optional. Specifies whether trigger failures are written to the transfer log. Specify one of the following options:

**yes**

Transfer log entries are created for failed triggered transfers. This is the default behavior even if you do not specify the **-tl** parameter.

**no**

No transfer log entries are created for failed triggered transfers.

## Parameters for specifying transfer options

**-jn *job\_name***

Optional. A user-defined job name identifier that is added to the transfer log message when the transfer starts.

## **-md**

Optional. Specifies the user-defined metadata that is passed to the exit points run by the agent. The **-md** parameter can take one or more name-value pairs that are separated by commas. Each name pair consists of *name=value*. You can use the **-md** parameter more than once in a command.

When the agent property **enableUserMetadataOptions** is set to a value of *true*, certain user-defined metadata keys provide more options to the transfer. For more information about the user-defined metadata keys that are currently supported, see [enableUserMetadataOptions: Supported MFT user-defined metadata keys](#). When the **enableUserMetadataOptions** property is set to *true*, key names starting with `com.ibm.wmqfte.` are not supported for user-defined use.

Any user metadata provided on the **fteCreateTransfer** command is made available as an environment variable to a process called through the **presrc**, **postsrc**, **predst**, and **postdst** parameters.

For example, the following transfer results in an environment variable called **procname** being set to *compress* (**procname=compress**) and is available to the `proc.sh` script:

```
fteCreateTransfer -sa ESBPA1 -sm ESBP10 -da INFOPA1
-dm INFOP1 -md procname=compress -df /home/mqm/hosts.out /etc/hosts -de overwrite
-postdst /home/mqm/proc.sh
```

## **-cs checksum**

Optional. Specifies whether a checksum algorithm is run on the file transfer data to check the integrity of the transferred files. Specify one of the following options:

### **MD5**

Computes an MD5 checksum for the data. The resulting checksum for the source and destination files is written to the transfer log for validation purposes. By default, Managed File Transfer computes MD5 checksums for all file transfers.

### **none**

No MD5 checksum is computed for the file transfer data. The transfer log records that checksum was set to none and the value for the checksum is blank. For example:

```
<checksum method="none"></checksum>
```

If you use the **none** option, you might improve file transfer performance, depending on your environment. However, selecting this option means that there is no validation of the source or destination files.

If you specify the **-cs** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify checksum behavior in the transfer definition file.

## **-pr transfer\_priority**

Optional. Specifies the priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is the priority level of the source agent.

This value matches the message priority value of IBM MQ, see [Getting messages from a queue: priority](#) for more information. Message traffic for file transfer data defaults to a priority level of 0, which allows your IBM MQ message traffic to take priority.

## **-qmp boolean**

Optional. Specifies whether the first message written to the destination queue by the transfer has IBM MQ message properties set. The valid options are as follows:

### **true**

Sets message properties on the first message that is created by the transfer.

### **false**

Does not set message properties on the first message that is created by the transfer. This is the default value.

You can specify the **-qmp** parameter only if you also specify the **-dq** parameter. For more information, see “MQ message properties set by MFT on messages written to destination queues” on page 2548

### **-qs message\_size**

Optional. Specifies whether to split the file into multiple fixed-length messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ `LAST_MSG_IN_GROUP` flag set. The size of the messages is specified by the value of *message\_size*. The format of *message\_size* is *lengthunits*, where *length* is a positive integer value and *units* is one of the following values:

#### **B**

Bytes. The minimum value that is allowed is two times the maximum bytes-per-character value of the code page of the destination messages.

#### **K**

This is equivalent to 1024 bytes.

#### **M**

This is equivalent to 1048576 bytes.

If the file is transferred in text mode, and is in a double-byte character set or multibyte character set, the file is split into messages on the closest character boundary to the specified message size.

You can specify the **-qs** parameter only if you also specify the **-dq** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

### **-qi**

Optional. Using this option includes the delimiter that is used to split the file into multiple messages in the messages. The delimiter is included at the beginning or at the end of the message, depending on the **-dqdp** parameter (which specifies prefix or postfix). By default the delimiter is not included in the messages.

You can specify the **-qi** parameter only if you also specify one of the **-dqdt** and **-dqdb** parameters.

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to create the file transfer. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files that are associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options that are based on the default coordination queue manager is used.

This parameter is not supported in the REST API interface.

### **-w timeout**

Optional. Specifying the **-w** parameter causes the **fteCreateTransfer** command to wait for a response from the agent before returning. If you do not specify this parameter, the **fteCreateTransfer** command waits a maximum of five seconds to receive an acknowledgment from the source agent for the transfer that the agent has received the transfer request. If no acknowledgment is received during the five-second wait, the **fteCreateTransfer** command returns the following warning message:

```
BFGCL0253W: No acknowledgment to command from agent within timeout.
```

The return code will be 0, unless you used the **-w** option on the command line.

The *timeout* argument is optional. If you specify *timeout*, the **fteCreateTransfer** command waits for up to *timeout* seconds for the agent to respond. If the agent does not respond before the time limit is reached, the command produces a warning and ends with a return code of 2 or 3. If you do not specify a *timeout* value, or you specify a *timeout* value of -1, then the command waits until the agent responds.



The REST service does not provide an equivalent option for this parameter, as ideal wait time is not recommended in a REST service implementation.

### **-rt *recovery\_timeout***

Optional. Sets the amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. Specify one of the following options:

#### **-1**

The agent continues to attempt to recover the stalled transfer until the transfer is complete. Using this option is the equivalent of the default behavior of the agent when the property is not set.

#### **0**

The agent stops the file transfer as soon as it enters recovery.

#### **>0**

The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. For example,

```
-rt 21600
```

indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. The maximum value for this parameter is 999999999.

Specifying the transfer recovery timeout value in this way sets it on a per transfer basis. To set a global value for all transfers in a Managed File Transfer network, you can add a property to the [The agent.properties file](#).

## **Parameters for invoking programs**

For more information about how you can start a program from Managed File Transfer, see [Specifying programs to run with MFT](#). For examples of specifying a program to invoke using the parameters that are described here, see [Examples of using fteCreateTransfer to start programs](#).

### **-presrc *pre\_source\_call***


Optional. Specifies a program to invoke at the source agent before the transfer starts. Use the following format for *pre\_source\_call*:

```
[type:]commandspec [, [retrycount] [, [retrywait] [, [successrc]]]
```

In this syntax, the variables are:

#### **type**

Optional. Valid values are **executable**, **antscript**, and **jcl**. The default value is **executable**.

 The **jcl** value is only applicable when targeted at an agent in a z/OS environment. In this case, the command refers to either a ZFS file, or a QSAM-readable dataset, or a member of a PDS. The contents should be JCL that can be submitted.

#### **commandspec**

Required. The command specification. Use one of the following formats:

- Type **executable**: *command* [(*arg1*, *arg2*, ...)]

If arguments contain variable substitutions, like `${FilePath}` or `${FileName}`, which are valid only if the substitution is initiated by a resource monitor, the variables are substituted with the first item in the transfer request.

For example, if a transfer request consists of files "reports01.csv, reports02.csv, reports03.csv", and the destination directory is "/output", the following transfer request:

```
fteCreateTransfer -sa 1 -da 2 -presrc "executable:archive(${FileName})"  
-dd TargetDir "${FilePath}" -gt task.xml
```

is replaced with

```
fteCreateTransfer -sa 1 -da 2 -presrc "executable:archive(reports01.csv)"
-dd TargetDir "/ouptut" -gt task.xml
```

- Type **antscript**: *command* [(*name1=var1* | *target1*, *name2=var2* | *target2*, ...)]
- Type **jcl**: *command*

where:

**command**

Required. The name of the program to call.

The **jcl** value is only applicable when targeted at an agent in a z/OS environment.

Arguments in brackets ( [ ] ) are optional and syntax depends on command type. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. **Windows** File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/).

**retrycount**

Optional. The number of times to retry calling the program if the program does not return a successful return code. Default value is 0.

**retrywait**

Optional. The time to wait, in seconds, before trying the program invocation again. Default value is 0 (no wait between retries).

**successrc**

Optional. Expression that is used to determine when the program invocation successfully runs. This expression can be composed of one or more expressions. Combine these expressions with a vertical bar character (|) to represent Boolean OR, or an ampersand (&) character to represent Boolean AND. Each expression is of the following form:

```
[>|<|!]value
```

where

- > Optional. A greater than test of the *value*.
- < Optional. A less than test of the *value*.
- ! Optional. A not equal test of the *value*.

**value**

Required. A valid integer.

If you do not specify this parameter, a default value of 0 is used.

**-predst *pre\_destination\_call***

Optional. Specifies a program to invoke at the destination agent before the transfer starts. *pre\_destination\_call* has the same format as *pre\_source\_call*.

**-postsrc *post\_source\_call***

Optional. Specifies a program to invoke at the source agent after the transfer has completed. *post\_source\_call* has the same format as *pre\_source\_call*.

**-postdst *post\_destination\_call***

Optional. Specifies a program to invoke at the destination agent after the transfer has completed. *post\_destination\_call* has the same format as *pre\_source\_call*.

## Parameters for specifying the destination

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, and **-dp** parameters is required. You cannot specify more than one of these parameters in a transfer request; they are mutually exclusive.

### **-td** *transfer\_definition\_file*

Optional. The name of the XML document that defines one or more source and destination file specifications for the transfer. Alternatively, the name of the XML document that contains a managed transfer request (which might have been generated by the **-gt** parameter). If you specify the **-td** parameter and also specify any other parameters on the command line, these other parameters override the corresponding value from the transfer definition file.

The **ftCreateTransfer** command locates the transfer definition file in relation to your current directory. If you cannot use relative path notation to specify the location of the transfer definition file, use the fully qualified path and file name of the transfer definition file instead.

**z/OS** On z/OS, you must store the transfer definition file in a UNIX file on z/OS UNIX System Services. You cannot store transfer definition files in z/OS sequential files or PDS members.

**IBM i** On IBM i, you must store the transfer definition file in the integrated file system.

For more information, see [Using transfer definition files](#).

### **-df** *destination\_file*

Optional. The name of the destination file.

If the destination agent is a Connect:Direct bridge agent, the destination file is specified in the format *connect\_direct\_node\_name:file\_path*. The Connect:Direct bridge agent accepts only file paths that are specified in this format. **z/OS** If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of `overwrite`.

Note the following information:

- If the destination agent is a protocol bridge agent and you want to specify an endpoint for a file, use the following format:

```
protocol_server:file_path
```

where *protocol\_server* is the name of the protocol server (which is optional) and where *file\_path* is the path to the file on the protocol server system. If you do not specify a protocol server, the default protocol server is used.

- If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against the destination agent, you can use the **-df** parameter in a transfer.
- **z/OS** When the destination agent is on z/OS, if the file specified starts with `//`, it is assumed to be a partitioned z/OS data set.

### **-dd** *destination\_directory*

Optional. The name of the directory the file is transferred to. Specify a valid directory name on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination directory is specified in the format *connect\_direct\_node\_name:directory\_path*. If the destination agent is a Connect:Direct bridge agent and the destination is a PDS, you must also specify the **-de** parameter with a value of `overwrite`.

Note the following information:

- If the destination agent is a protocol bridge agent and you want to specify a directory at a particular endpoint, use the following format:

```
protocol_server:directory_path
```

where *protocol\_server* is the name of the protocol server (which is optional) and where *directory\_path* is the path to the directory on the protocol server system. If you do not specify a protocol server, the default protocol server is used.

- If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against the destination agent, you can use the **-dd** parameter in a transfer.
- **z/OS** When the destination agent is on z/OS, if the file specified starts with //, it is assumed to be a z/OS partitioned data set.

### **z/OS** -ds destination\_sequential\_data\_set

z/OS only. Optional. The name of the sequential data set or PDS member that files are transferred into. Specify a sequential data set name or a partitioned data set member. For information about transferring data sets, see [“Guidelines for transferring files” on page 2491](#).

The syntax for the data set name is as follows:

```
//data_set_name{;attribute(value);...;attribute(value)}
```

or

```
//pds_data_set_name(member_name){;attribute(value);...;attribute(value)}
```

That is, a data set name specifier prefixed with // and optionally followed by a number of attributes that are separated by semicolons.

#### **For example:**

```
//'TEST.FILE.NAME';DSNTYPE(PDS);RECFM(F,B);BLKSIZE(800);LRECL(80);CYL;SPACE(2,2)
```

If the data set is located at a Connect:Direct node, you must prefix the data set name with the node name. For example:

```
CD_NODE1://'OBJECT.LIB';RECFM(F,B);BLKSIZE(800);LRECL(80)
```

If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite. For more information about data set transfers to or from Connect:Direct nodes, see [“Transferring data sets to and from Connect:Direct nodes” on page 2500](#).

For transfers that only involve Managed File Transfer agents, if the data set name part is enclosed by single quotation mark characters, it specifies a fully qualified data set name. If the data set name is not enclosed by single quotation mark characters, the system adds the default high-level qualifier for the destination agent (either the value for the transferRootHLQ agent property or the user ID that the agent runs under, if you have not set transferRootHLQ).

**Note:** **z/OS** However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name. This is the case even if the data set name is enclosed by single quotation mark characters.

When you transfer a file or data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes that are passed in the transfer definition. If no attributes are specified, attributes are set to the same as the source data set or to the default values when the source is a file. The attributes of an existing tape data set are ignored.

The data set attributes are used either to create a data set or to ensure that an existing data set is compatible. The specification of data set attributes is in a form suitable for BPXWDYN (see

Requesting [dynamic allocation](#) for more information). When the agent is to create a destination data set, the following BPXWDYN attributes are automatically specified: DSN(*data\_set\_name*) NEW CATALOG MSG(*numeric\_file\_descriptor*). The value of *numeric\_file\_descriptor* is generated by Managed File Transfer. For a data set to data set transfer, the attributes of RECFM, LRECL, and BLKSIZE from the source are selected for a new destination data set. The SPACE setting for a new destination data set is not set by Managed File Transfer and system defaults are used. Therefore, you are recommended to specify the SPACE attribute when a new data set is to be created. You can use the **bpxwdynAllocAdditionalProperties** property in the agent .properties file to set BPXWDYN options that apply to all transfers. For more information, see [The MFT agent .properties file](#).

**z/OS** Some BPXWDYN options must not be specified when using the **fteCreateTemplate** command, the **fteCreateTransfer** command or the **bpxwdynAllocAdditionalProperties** property in the agent .properties file. For a list of these properties, see [“BPXWDYN properties you must not use with MFT” on page 2507](#).

The **-ds** parameter is not supported when the destination agent is a protocol bridge agent.

If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-ds** parameter in a transfer. Using the **-ds** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard Managed File Transfer I/O is used instead.

#### **z/OS** **-dp destination\_partitioned\_data\_set**

z/OS only. Optional. The name of the destination PDS that files are transferred into. Specify a partitioned data set name. If a PDS is created as a result of the transfer, this PDS is created as a PDSE by default. You can override the default by specifying DSNTYPE=PDS.

The syntax for the PDS data set name is as follows:

```
//pds_data_set_name{;attribute;...;attribute}
```

The syntax for the data set name is the same as described for the **-ds destination\_sequential\_data\_set** parameter. All the syntax details for specifying data sets that are located on Connect:Direct nodes also apply to the **-dp** parameter. If the destination agent is a Connect:Direct bridge agent, you must also specify the **-de** parameter with a value of overwrite.

The **-dp** parameter is not supported when the destination agent is a protocol bridge agent.

If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-dp** parameter in a transfer. Using the **-dp** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard Managed File Transfer I/O is used instead.

#### **-dq destination\_queue**

Optional. The name of a destination queue that files are transferred onto. You can optionally include a queue manager name in this specification, by using the format QUEUE@QUEUEMANAGER. If you do not specify a queue manager name the destination agent queue manager name is used. You must specify a valid queue name that exists on the queue manager.

The **-dq** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent, or when the source specification is a queue.

If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-dq** parameter in a transfer. Using the **-dq** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard Managed File Transfer I/O is used instead.

#### **-dqp persistent**

Optional. Specifies whether messages written to the destination queue are persistent. The valid options are as follows:

**true**

Writes persistent messages to the destination queue. This is the default value.

**false**

Writes non-persistent messages to the destination queue.

**qdef**

The persistence value is taken from the DefPersistence attribute of the destination queue.

You can specify the **-dqpb** parameter only if you also specify the **-dq** parameter.

**-dqdb hexadecimal\_delimiter**

Optional. Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ LAST\_MSG\_IN\_GROUP flag set. The format for specifying a hexadecimal byte as a delimiter is xNN, where N is a character in the range 0-9 or a-f. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes, for example: x3e, x20, x20, xbf.

You can specify the **-dqdb** parameter only if you also specify the **-dq** parameter and the transfer is in binary mode. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

**-dqdt pattern**

Optional. Specifies the Java regular expression to use when splitting a text file into multiple messages. All the messages have the same IBM MQ group ID; the last message in the group has the IBM MQ LAST\_MSG\_IN\_GROUP flag set. The format for specifying a regular expression as a delimiter is a regular expression that is enclosed in parentheses, (*regular\_expression*), or enclosed in double quotation marks, "*regular\_expression*". For more information, see [“Regular expressions used by MFT” on page 2523](#).

By default, the length of the string that the regular expression can match is limited by the destination agent to five characters. You can change this behavior by editing the **maxDelimiterMatchLength** agent property. For more information, see [Advanced agent properties](#).

You can specify the **-dqdt** parameter only if you also specify the **-dq** parameter and the value text for the **-t** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

**-dqdp position**

Optional. Specifies the expected position of destination text and binary delimiters when splitting files. You can specify the **-dqdp** parameter only if you also specify one of the **-dqdt** and **-dqdb** parameters.

Specify one of the following options:

**prefix**

The delimiters are expected at the beginning of each line.

**postfix**

The delimiters are expected at the end of each line. This is the default option.

**-de destination\_file\_behavior**

Optional. Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:

**error**

Reports an error and the file is not transferred. This is the default value.

**overwrite**

Overwrites the existing destination file.

If you specify the **-de** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify destination file exists behavior in the transfer definition file.

### **-t transfer type**

Optional. Specifies the type of file transfer: binary mode or text mode.

#### **binary**

The data in the file is transferred without any conversion. This is the default value.

#### **text**

The code page and end-of-line characters of the file are converted. You can specify which code page and line ending to use for the conversion with the **-sce**, **-dce** or **-dle** parameters. If you do not specify the **-sce**, **-dce** or **-dle** parameters, the exact conversions performed depend on the operating system of the source agent and destination agent.

**z/OS** For example, a file that is transferred from Windows to z/OS has its code page converted from ASCII to EBCDIC. When a file is converted from ASCII to EBCDIC, the end-of-line characters are converted from ASCII carriage return (CR) and line feed (LF) character pairs to an EBCDIC new line (NL) character.

**z/OS** For more information about how z/OS data sets are transferred, see “[Transferring files and data sets between z/OS and distributed systems](#)” on page 2492 and “[Transferring between data sets on z/OS](#)” on page 2493.

If you specify the **-t** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify transfer mode behavior in the transfer definition file.

### **-dce destination\_character\_encoding**

Optional. Specifies which character encoding to use to write the file at the destination. This option is only applicable to text files and so **-t text** must also be specified. The code pages available for conversion depend on the platform of the destination agent. For a list of available code pages, see [Available code pages for MFT](#).

#### **noswaplfnl**

By default Managed File Transfer uses swaplfnl with supported EBCDIC character sets. Using swaplfnl changes the behavior of the character set mapping from and to the EBCDIC LF 0x25 character. However, this can sometimes result in a mapping that is not what you want. Use noswaplfnl to override this behavior.

### **-dle destination\_line\_ending**

Optional. Specifies the end-of-line characters that are used when the file is written at the destination. This option is applicable to text files only and so you must also specify the **-t text** parameter. The valid options are:

#### **LF**

Line feed. This is the default for the following platforms:

- **Linux** **AIX** AIX and Linux platforms
- **z/OS** z/OS UNIX System Services files

When you use the standard EBCDIC code pages that are supplied with Managed File Transfer for EBCDIC files, the end-of-line characters are mapped to a NL character (0x15) and not to a LF character (0x25).

#### **CRLF**

Carriage return followed by line feed. **Windows** This is the default for Windows.

**z/OS** If the destination of the transfer is a z/OS data set, this option is ignored.

#### **z/OS -dtr**

Optional. Specifies that destination records longer than the LRECL data set attribute are truncated. If this parameter is not specified, the records are wrapped. This parameter is valid only for text mode transfers where the destination is a data set.

### **-dfa attributes**

Optional. When transferring to an IBM MQ 8.0 Managed File Transfer agent running on 4690, this parameter is used to specify a semicolon separated list of file attributes that are associated with the destination files in the transfer. The **-dfa** parameter can be specified with or without a value. For example, without a value:

```
-dfa ATTRIBUTE1;ATTRIBUTE2
```

For example, with a value:

```
-dfa ATTRIBUTE1(VALUE);ATTRIBUTE2(VALUE)
```

For example, one attribute with a value and one without:

```
-dfa ATTRIBUTE1;ATTRIBUTE2(VALUE)
```

You can use the **-dfa** parameter more than once in a command.

For more information about file attributes on 4690, see [File distribution attributes](#) in the IBM MQ 8.0 documentation.

## **Parameters for security**



### **-mquserid *user\_id***

Optional. Specifies the user ID to authenticate with the command queue manager.

### **-mqpassword *password***

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

 **V 9.3.0**  **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using [MQCSP](#) authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

## **Parameters for specifying the source**

### **-sd *source\_file\_disposition***

Optional. Specifies the action that is taken on a source file in file-to-file or file-to-message transfers when that source file is successfully transferred to its destination. The valid options are as follows:

#### **leave**

The source files are left unchanged. This is the default value.

#### **delete**

The source files are deleted from the source system after the source files are successfully transferred.



**Note:** For message-to-file transfers, the messages on the source queue are always deleted once they have been successfully transferred. This means that if the **-sd** parameter is set to leave for a message-to-file transfer, the value is ignored.

**z/OS** On z/OS, if the source is a tape data set and you specify the delete option, the tape is remounted to delete the data set. This behavior is because of the behavior of the system environment.

If the source is a queue and you specify the leave option, the command returns an error and a transfer is not requested.

If the source agent is a Connect:Direct bridge agent and you specify the delete option, the behavior is different to the usual source disposition behavior. One of the following cases occurs:

- If Connect:Direct uses a process that is generated by Managed File Transfer to move the file or data set from the source, specifying the delete option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see [Submitting a user-defined Connect:Direct process from a file transfer request](#).
- If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

If you specify the **-sd** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify source disposition behavior in the transfer definition file.

**-r**

Optional. Recursively transfer files in subdirectories when *source\_specification* contains wildcard characters. When Managed File Transfer is presented with a wildcard character as a *source\_specification*, any subdirectories that match the wildcard character are transferred only if you specify the **-r** parameter. When *source\_specification* matches a subdirectory, all files in that directory and its subdirectories (including hidden files) are always transferred.

For more information about how Managed File Transfer handles wildcard characters, see [“Using wildcard characters with MFT”](#) on page 2518

If you specify the **-r** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify recursive behavior in the transfer definition file.

**-sce source\_character\_encoding**

Optional. Specifies which character encoding to use to read the source file when performing character conversion. This option is only applicable to text files and so **-t text** must also be specified. The code pages available for conversion depend on the platform of the destination agent, because the conversion is performed on the destination system. For a list of available code pages, see [“Available code pages for MFT”](#) on page 2554.

**noswaplfnl**

By default Managed File Transfer uses swaplfnl with supported EBCDIC character sets. Using swaplfnl changes the behavior of the character set mapping from and to the EBCDIC LF 0x25 character. However, this can sometimes result in a mapping that is not what you want. Use noswaplfnl to override this behavior.

**z/OS -skeep**

Optional. Specifies that trailing spaces are kept on source records that are read from a fixed-length-format record-oriented file (for example, a z/OS data set) as part of a text mode transfer. If you do not specify this parameter, trailing spaces are stripped from source records.

**z/OS -srd delimiter**

Optional. For source files that are record oriented (for example, z/OS data sets), specifies one or more byte values to insert as the delimiter when appending records into a binary file. You must specify

each value as two hexadecimal digits in the range 00-FF, prefixed by x. Separate multiple bytes with commas. For example:

```
-srdp x0A
```

or

```
-srdp x0D,x0A
```

You must configure the transfer in binary mode.

### **-srdp position**

Optional. Specifies the position to insert source record delimiters. You can specify the **-srdp** parameter only if you also specify the **-srdp** parameter.

Specify one of the following options:

#### **prefix**

The delimiters are inserted at the start of each record.

#### **postfix**

The delimiters are inserted at the end of each record. This is the default option.

### **-sq**

Optional. Specifies that the source of a transfer is a queue.

If you want to invoke any of the Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-sq** parameter in a transfer. Using the **-sq** parameter prevents the transfer I/O user exits from being invoked for the source and means that the standard Managed File Transfer I/O is used instead.

### **-sqgi**

Optional. Specifies that the messages are grouped by IBM MQ group ID. The first complete group is written to the destination file. If this parameter is not specified, all messages on the source queue are written to the destination file.

You can specify the **-sqgi** parameter only if you also specify the **-sq** parameter.

### **-sqdt text\_delimiter**

Optional. Specifies a sequence of text to insert as the delimiter when appending multiple messages to a text file. You can include Java escape sequences for String literals in the delimiter. For example, `-sqdt \u007d\n`.

The text delimiter is encoded to binary format by using the source encoding of the transfer. Each message is read in binary format. The encoded delimiter is prepended or appended in binary format to the message (as specified by the **-sqdp** parameter) and the result is transferred in binary format to the destination agent. If the source agent code page includes shift-in and shift-out states, the agent assumes that each message is in the shift-out state at the end of the message. At the destination agent the binary data is converted in the same way as a file to file text transfer.

You can specify the **-sqdt** parameter only if you also specify the **-sq** parameter and the value `text` for the **-t** parameter.

### **-sqdb hexadecimal\_delimiter**

Optional. Specifies one or more byte values to insert as the delimiter when appending multiple messages to a binary file. Each value must be specified as two hexadecimal digits in the range 00-FF, prefixed by x. Multiple bytes must be comma-separated. For example, `-sqdb x08,xA4`.

You can specify the **-sqdb** parameter only if you also specify the **-sq** parameter. You cannot specify the **-sqdb** parameter if you also specify the value `text` for the **-t** parameter.

### **-sqdp position**

Optional. Specifies the position of insertion of source text and binary delimiters. You can specify the **-sqdp** parameter only if you have also specified one of the **-sqdt** and **-sqdb** parameters.

Specify one of the following options:

#### **prefix**

The delimiters are inserted at the start of each message

#### **postfix**

The delimiters are inserted at the end of each message. This is the default option.

### **-sqwt wait\_time**

Optional. Specifies the time, in seconds, to wait for one of the following conditions to be met:

- For a new message to appear on the queue
- If the **-sqgi** parameter was specified, for a complete group to appear on the queue

If neither of these conditions is met within the time that is specified by *wait\_time*, the source agent stops reading from the queue and completes the transfer. If the **-sqwt** parameter is not specified, the source agent stops reading from the source queue immediately if the source queue is empty or, in the case where the **-sqgi** parameter is specified, if there is no complete group on the queue.

For information about using the **-sqwt** parameter, see [“Guidance for specifying a wait time on a message-to-file transfer” on page 2553](#).

You can specify the **-sqwt** parameter only if you also specify the **-sq** parameter.

### **source\_specification**

One or more file specifications that determine the source, or sources, for the file transfer.

Required if you specify one of the **-df**, **-dd**, **-dp**, **-dq**, or **-ds** parameters. If you specify the **-td** parameter, do not specify *source\_specification*.

- If you have not specified the **-sq** parameter, *source\_specification* is one or more file specifications that determine the source, or sources, for the file transfer. File specifications can take one of five forms and can include wildcard characters. For more information about wildcard characters, see [“Using wildcard characters with MFT” on page 2518](#). You can escape asterisks that are part of the file specification by using two asterisk characters (\*\*) in the file specification.

You can specify multiple source file specifications separated by the space character. However, if you specify multiple source specifications for the **-df** or **-ds** parameters and also specify **-de overwrite**, the destination will contain only the data for the source file that you specified last. If you do not specify **-de overwrite** the transfer can only be partially successful. If the destination file did not previously exist, it will contain the data for the source file that you specified first.

To transfer files that contain spaces in their file names, for example a b.txt to file c d.txt, place double quotation marks around the file names that contain spaces. Specify the following text as part of the **fteCreateTransfer** command:

```
-df "c d.txt" "a b.txt"
```

Each file specification must be in one of the following categories:

#### **File names**

The name of a file, expressed in the appropriate notation for the system where the source agent is running. When a file name is specified as a source file specification, the contents of the file are copied.

#### **Directories**

The name of a directory, expressed in the appropriate notation for the system where the source agent is running. When a directory is specified as a source file specification, the contents of the directory are copied. More precisely, all files in the directory and in all its subdirectories, including hidden files, are copied.

For example, to copy the contents of DIR1 to DIR2 only, specify `fteCreateTransfer ... -dd DIR2 DIR1/*`

### **z/OS Sequential data set**

The name of a sequential data set or partitioned data set member. Denote data sets by preceding the data set name with two forward slash characters (`//`).

If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

### **z/OS Partitioned data set**

The name of a partitioned data set. Denote data set names by preceding the data set name with two forward slash characters (`//`).

If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

### **File name or directory at a Connect:Direct node**

(Connect:Direct bridge agent only). The name of a Connect:Direct node, a colon character (`:`), and a file or directory path on the system that is hosting the Connect:Direct node. For example, `connect_direct_node_name:file_path`.

If the source agent is a Connect:Direct bridge agent, it will only accept source specifications in this form.

**Note:** Wildcard characters are not supported in file paths when the source agent is a Connect:Direct bridge agent.

### **File name or directory on a protocol file server**

The name of a protocol file server, a colon character (`:`), and a file or directory path on the protocol server system. For example, `protocol_server:file_path`.

If you do not specify a protocol server, the default protocol server is used.

- If you specify the `-sq` parameter, `source_specification` is the name of a local queue on the source agent queue manager. You can specify only one source queue. The source queue is specified in the format:

```
QUEUE_NAME
```

The queue manager name is not included in the source queue specification, because the queue manager must be the same as the source agent queue manager.

- **z/OS** If the source agent is on z/OS, source files that start with `//` are assumed to be z/OS partitioned data sets.

## **Other parameters**

### **-? or -h**

Optional. Displays command syntax.

## **Examples**

In this basic example, the file `originalfile.txt` is transferred from AGENT1 to AGENT2 on the same system and renamed to `transferredfile.txt`

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -df C:/import/transferredfile.txt C:/export/originalfile.txt
```

In this example, the files `originalfile.txt` and `originalfile2.txt` are transferred from AGENT1 to AGENT2 on the same system, to the directory `C:/import`

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:/import C:/export/originalfile.txt C:/export/originalfile2.txt
```

In this example, the file `originalfile.txt` is transferred from AGENT1's system to AGENT2's system. The file transfer is scheduled to take place at 09:00 based on the system time of the source agent's system and occurs every two hours four times:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tb source -ss 09:00 -oi hours -of 2 -oc 4
-df C:/import/transferredfile.txt C:/export/originalfile.txt
```

In this example, the file `originalfile.txt` is transferred from AGENT1 to AGENT2, on condition that the file `A.txt` exists on AGENT1:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tr file=exist,C:/export/A.txt -df C:/import/transferredfile.txt C:/export/originalfile.txt
```

**z/OS** In this example, the file `originalfile.txt` is transferred from AGENT1's system to a data set `//'USERID.TRANS.FILE.TXT'` on AGENT2's system. Text mode is selected to convert data from ASCII to EBCDIC.

```
fteCreateTransfer -t text -sa AGENT1 -da AGENT2
-ds "//TRANS.FILE.TXT;RECFM(V,B);BLKSIZE(6144);LRECL(1028);
SPACE(5,1)" C:/export/originalfile.txt
```

**z/OS** In this example, a member of a fully qualified data set on AGENT1's system is transferred to a file on AGENT2's system. Text mode is selected to convert the file from EBCDIC to the default code page of AGENT2's system.

```
fteCreateTransfer -t text -sa AGENT1 -da AGENT2 -df /tmp/IEEUJV.txt "'SYS1.SAMPLIB(IEEUJV)'"
```

In this example, a file that is called `file.bin` on agent AGENT1 is transferred to a destination file called `file.bin` on the protocol file server `accountshost.ibm.com` by using the destination agent BRIDGE1.

```
fteCreateTransfer -sa AGENT1 -da BRIDGE1 -df accountshost.ibm.com:/tmp/file.bin /tmp/file.bin
```

In this example, a wildcard is used without quotation marks. All files in AGENT1's current working directory that end in `.txt` are transferred to directory `C:/import` on AGENT2. The file names remain unchanged.

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:/import *.txt
```

In this example, a wildcard is used with double quotation marks. All files in AGENT1's transfer root directory that end in `.txt` are transferred to directory `C:/import` on AGENT2. The file names remain unchanged.

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:/import "*.txt"
```

## Return codes

<i>Table 345. Return code names and descriptions</i>	
<b>Return code</b>	<b>Description</b>
0	Command completed successfully.
1	Command ended unsuccessfully.
2	Command ended with a timeout. The command sent a message to the agent, but the agent did not respond within the time specified.

Table 345. Return code names and descriptions (continued)

Return code	Description
3	Command ended with a timeout. The command was waiting for an acknowledgment from the agent, but did not receive one within the timeout period.
20	Command completed with partial success and some files were transferred.
21	The queue manager that the <b>fteCreateTransfer</b> command was connected to was stopped before the transfer result was determined.
40	Failed. None of the files specified were transferred.
41	The transfer was canceled.
42	The transfer did not take place because the transfer was conditional and the required condition was not met.
43	The transfer request message was malformed.
44	The source agent did not have sufficient capacity to carry out the transfer.
45	The destination agent did not have sufficient capacity to carry out the transfer.
46	The number of files that are being transferred exceeded the source agent's limit.
47	The number of files transferred exceeds the destination agent's limit.

**Note:** The return code will always be 0 or 1, unless the **-w** parameter is used on the command line.

#### Related concepts

[Timeout option for file transfers in recovery](#)

#### Related tasks

[Starting a new file transfer](#)

[Using transfer definition files](#)

[Creating a scheduled file transfer](#)

[Triggering a file transfer](#)

## fteDefine (generate configuration scripts)

Use the **fteDefine** command to generate the configuration scripts necessary to define the specified Agent Queue Manager objects.

### Purpose

You would expect to use the **fteDefine** command when some configuration steps need to be run on a system that is remote to the one containing the configuration data. For example, configuring the queues for an agent on a queue manager to be accessed over a client connection.

### Syntax

<p><b>fteDefine</b></p> <pre> &gt;&gt; fteDefine -t type -d output_dir name &lt;&lt; </pre>
---

### Parameters

#### -t type

Required. The type of object to be defined. The options for type are agent.

**-d output\_dir**

Optional. A directory path in which the scripts are written. If not provided, the scripts are written to the standard output stream.

**name**

Required. One or more names of the objects to be defined. To specify names for more than one object, separate them with a space. For example, *name1 name2 . . .*

**-? or -h**

Optional. Displays command syntax.

**Examples**

In this example, the **fteDefine** command is specified with the **-t agent** parameter and a single agent name. The output is written to a file.

```
fteDefine -t agent EXAMPLE.AGENT >EXAMPLE.AGENT_create.mqsc
```

The output that is generated from this command are the MQSC command scripts to be run against the agent queue manager to create the necessary agent queues:

```
$ fteDefine -t agent EXAMPLE.AGENT
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
DEFINE QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT) +
  DEFPRTY(0) +
  DEFSOPT(SHARED) +
  GET(ENABLED) +
  MAXDEPTH(5000) +
  MAXMSGL(4194304) +
  MSGDLVSQ(PRIORITY) +
  PUT(ENABLED) +
  RETINTVL(999999999) +
  SHARE +
  NOTRIGGER +
  USAGE(NORMAL) +
  REPLACE
DEFINE QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT) +
  DEFPRTY(0) +
  DEFSOPT(SHARED) +
  GET(ENABLED) +
  MAXDEPTH(5000) +
  MAXMSGL(4194304) +
  MSGDLVSQ(PRIORITY) +
  PUT(ENABLED) +
  RETINTVL(999999999) +
  SHARE +
  NOTRIGGER +
  USAGE(NORMAL) +
  REPLACE
...
etc.
```

In this example, the **fteDefine** command is specified with the **-d outputDir** parameter and several agent names.

```
fteDefine -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
```

The output that is generated from this command are the absolute file paths to the locations of the MQSC command scripts:

```
$ fteDefine -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent
EXAMPLE.AGENT.1.
The file can be found here: '/tmp/EXAMPLE.AGENT.1_create.mqsc'.
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent
EXAMPLE.AGENT.2.
The file can be found here: '/tmp/EXAMPLE.AGENT.2_create.mqsc'.
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent
```

EXAMPLE.AGENT.3.  
The file can be found here: '/tmp/EXAMPLE.AGENT.3\_create.mqsc'.

## Return codes

- 0** Command completed successfully.
- 1** Command ended unsuccessfully.

## Related reference

[“fteDelete \(generate scripts to remove configuration\)” on page 2072](#)

Use the **fteDelete** command to generate the configuration scripts necessary to remove the specified Agent Queue Manager objects.

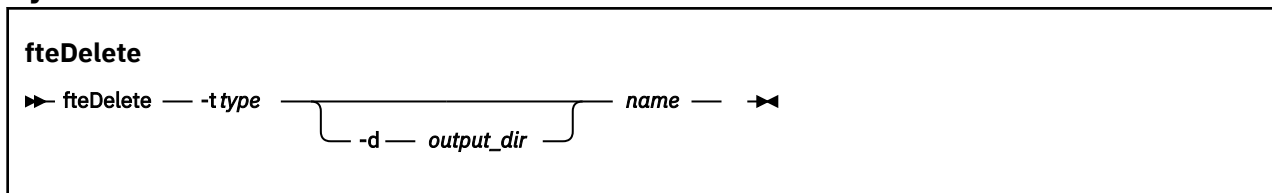
## fteDelete (generate scripts to remove configuration)

Use the **fteDelete** command to generate the configuration scripts necessary to remove the specified Agent Queue Manager objects.

## Purpose

You would expect to use the **fteDelete** command when some configuration steps need to be run on a system that is remote to the one containing the configuration data. For example, removing the queues for a remote client agent on a local queue manager.

## Syntax



## Parameters

### -t type

Required. The type of object to be delete. The options for type are agent.

### -d output\_dir

Optional. A directory path in which the scripts are written. If not provided, the scripts are written to the standard output stream.

### name

Required. One or more names of the objects to be delete. To specify names for more than one object, separate them with a space. For example, *name1 name2 . . .*

### -? or -h

Optional. Displays command syntax.

## Examples

In this example, the **fteDelete** command is specified with the **-t agent** parameter and a single agent name. The output is written to a file.

```
fteDelete -t agent EXAMPLE.AGENT >EXAMPLE.AGENT_delete.mqsc
```

The output that is generated from this command are the MQSC command scripts to be run against the agent queue manager to delete the agent queues:



```

$ fteDelete -t agent EXAMPLE.AGENT
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
CLEAR QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.REPLY.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.REPLY.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.STATE.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.STATE.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.EVENT.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.EVENT.EXAMPLE.AGENT)
...
etc.

```

In this example, the **fteDelete** command is specified with the **-d outputDir** parameter and several agent names.

```
fteDelete -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
```

The output that is generated from this command are the absolute file paths to the locations of the MQSC command scripts:

```

$ fteDelete -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent
EXAMPLE.AGENT.1.
The file can be found here: '/tmp/EXAMPLE.AGENT.1_delete.mqsc'.
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent
EXAMPLE.AGENT.2.
The file can be found here: '/tmp/EXAMPLE.AGENT.2_delete.mqsc'.
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent
EXAMPLE.AGENT.3.
The file can be found here: '/tmp/EXAMPLE.AGENT.3_delete.mqsc'.

```

## Return codes

- 0** Command completed successfully.
- 1** Command ended unsuccessfully.

## Related reference

[“fteDefine \(generate configuration scripts\)” on page 2070](#)

Use the **fteDefine** command to generate the configuration scripts necessary to define the specified Agent Queue Manager objects.

## fteDeleteAgent (delete an MFT agent and its configuration)

The **fteDeleteAgent** command deletes a Managed File Transfer Agent and its configuration. If the agent is a protocol bridge agent, the user credentials file is left on the file system.

## Purpose

Stop the agent with the [fteStopAgent](#) command before running the **fteDeleteAgent** command.

If you have configured your agent to run as a Windows service, running the **fteDeleteAgent** command deletes the service definition.

From IBM MQ 9.1, any resource monitor and scheduled transfers are removed when the agent is deleted.

Only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive an error message and the command will not run.

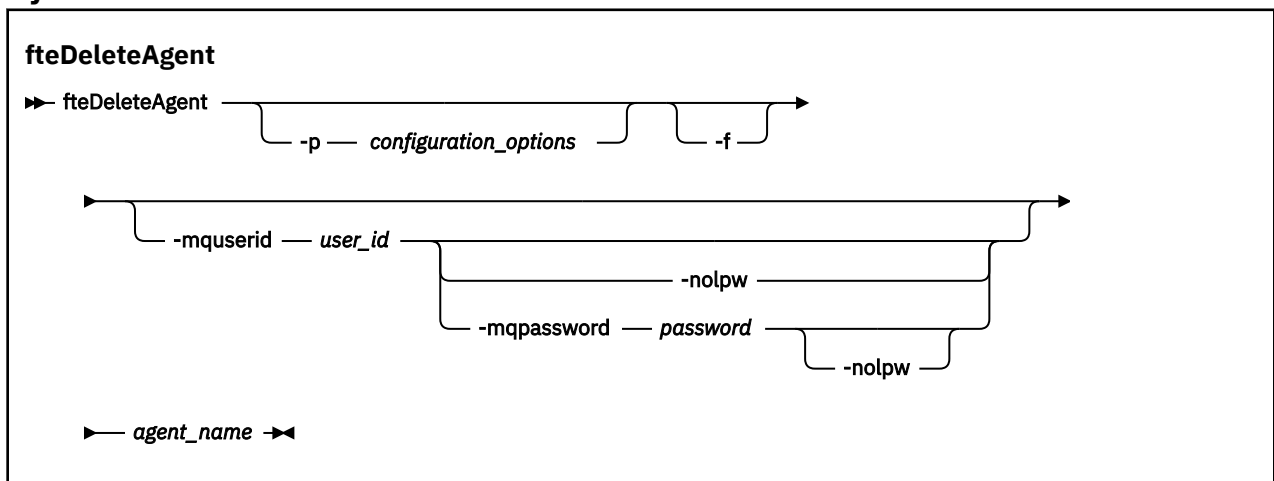
The **fteDeleteAgent** command provides you with the MQSC commands that you must run against your agent's queue manager to clear and delete the agent's system queues. These queues are as follows:

- SYSTEM.FTE.AUTHADM1.*agent\_name*
- SYSTEM.FTE.AUTHAGT1.*agent\_name*
- SYSTEM.FTE.AUTHMON1.*agent\_name*
- SYSTEM.FTE.AUTHOPS1.*agent\_name*
- SYSTEM.FTE.AUTHSCH1.*agent\_name*
- SYSTEM.FTE.AUTHTRN1.*agent\_name*
- SYSTEM.FTE.COMMAND.*agent\_name*
- SYSTEM.FTE.DATA.*agent\_name*
- SYSTEM.FTE.EVENT.*agent\_name*
- SYSTEM.FTE.REPLY.*agent\_name*
- SYSTEM.FTE.STATE.*agent\_name*

The **fteCreateAgent** command also provides these commands in a file in the following location:

```
MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name/agent_name_delete.mqsc
```

## Syntax



## Parameters

### **-p configuration\_options**

Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which agent configuration you want to delete. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in the `installation.properties` file are used. See [Configuration options](#) for more information.

### **-f**

Optional. Forces the command to deregister the agent from the coordination queue manager even if the agent's configuration files cannot be found. Because information about the agent's queue manager is not available in this situation, the command will connect directly to the coordination queue manager instead of using the agent queue manager as it normally would.



**-mquserid *user\_id***

Optional. Specifies the user ID to authenticate with the agent queue manager, unless the force **-f** parameter is present. If the **-f** parameter is present, it specifies the user ID to authenticate with the coordination queue manager.

**-mqpassword *password***

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

  From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

  **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

***agent\_name***

Required. The name of the agent that you want to delete.

**-? or -h**

Optional. Displays command syntax.

**Example**

In this example, AGENT3 and its configuration on coordination queue manager QM\_COORD1 are deleted:

```
fteDeleteAgent -p QM_COORD1 AGENT3
```

This example command outputs the following MQSC commands to delete the agent's three queues:

```
CLEAR QLOCAL(SYSTEM.FTE.COMMAND.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.COMMAND.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.DATA.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.DATA.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.REPLY.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.REPLY.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.STATE.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.STATE.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.EVENT.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.EVENT.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHADM1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHADM1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHAGT1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHAGT1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHTRN1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHTRN1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHOPS1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHOPS1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHSCH1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHSCH1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHMON1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHMON1.AGENT3)
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

### Related reference

[“fteStopAgent \(stop an MFT agent\)” on page 2146](#)

Use the **fteStopAgent** command to either stop a Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

[“fteCleanAgent \(clean up an MFT Agent\)” on page 1992](#)

Use the **fteCleanAgent** command to clean up the queues that a Managed File Transfer Agent uses, by deleting messages from the persistent and non-persistent queues used by the agent. Use the **fteCleanAgent** command if you are having problems starting an agent, which might be caused by information remaining on the queues used by the agent.

[“fteCreateAgent \(create an MFT agent\)” on page 1999](#)


The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.


[“fteStartAgent \(start an MFT agent\)” on page 2140](#)

The **fteStartAgent** command starts a Managed File Transfer agent from the command line.

## **fteDeleteLogger (delete an MFT logger and its configuration)**

Use the **fteDeleteLogger** command to delete a Managed File Transfer logger and its configuration. Existing log files associated with the logger can either be retained or deleted.

**Important:**  On IBM MQ for AIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the command:

- Be a member of the mqm group (if the mqm group is defined on the system).
- Be a member of the group named in the BFG\_GROUP\_NAME environment variable (if one is named).
- Have no value set in the BFG\_GROUP\_NAME environment variable when the command is run.

## Loggers on IBM i



Managed File Transfer loggers are not supported on the IBM i platform.

## Purpose

Stop the logger with the **fteStopLogger** command before running the **fteDeleteLogger** command.

If you have configured your logger to run as a Windows service, running the **fteDeleteLogger** command deletes the service definition.

The logger configuration directory contains a MQSC script to delete the queues and the subscription for the logger. These queues are as follows:

- SYSTEM.FTE.LOG.CMD.*logger\_name*
- SYSTEM.FTE.LOG.RJCT.*logger\_name*

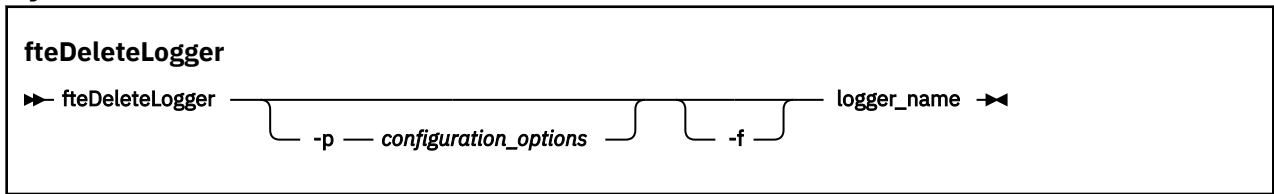
The subscription name is as follows:

- SYSTEM.FTE.AUTO.SUB.*logger\_name*

The MQSC script can be found at

*MQ\_DATA\_PATH*\mqft\config\coordination\_qmgr\loggers\*logger\_name*\*logger\_name\_delete*.mqsc

## Syntax



## Parameters

### **-p** *configuration\_options*

Optional. Determines the set of configuration options that is used to start the stand-alone database logger. Use the name of a set of configuration options as the value for the **-p** parameter. By convention this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

### **-f**

Optional. Forces the removal of any log files created by this logger. If this parameter is omitted, any log files created by the logger will be retained, and must be manually removed when they are no longer required.

### *logger\_name*

Required. The name of the logger that you want to delete.

### **-? or -h**

Optional. Displays command syntax.

## Example

In this example, a logger called `logger1` is deleted. The **-f** parameter has been specified, which causes the logger's log files to be removed as well as the logger's configuration files.

```
fteDeleteLogger -f logger1
```

## Return codes

### **0**

Command completed successfully.

### **1**

Command ended unsuccessfully.

## Related reference

[“fteCreateLogger \(create an MFT file or database logger\)” on page 2020](#)

Use the **fteCreateLogger** command to create a Managed File Transfer file or database logger.

[“fteStartLogger \(start an MFT logger\)” on page 2142](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

[“fteStopLogger \(stop an MFT logger\)” on page 2148](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 2100](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an

IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

## **fteDeleteMonitor (delete an MFT resource monitor)**

Use the **fteDeleteMonitor** command to stop and delete an existing Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

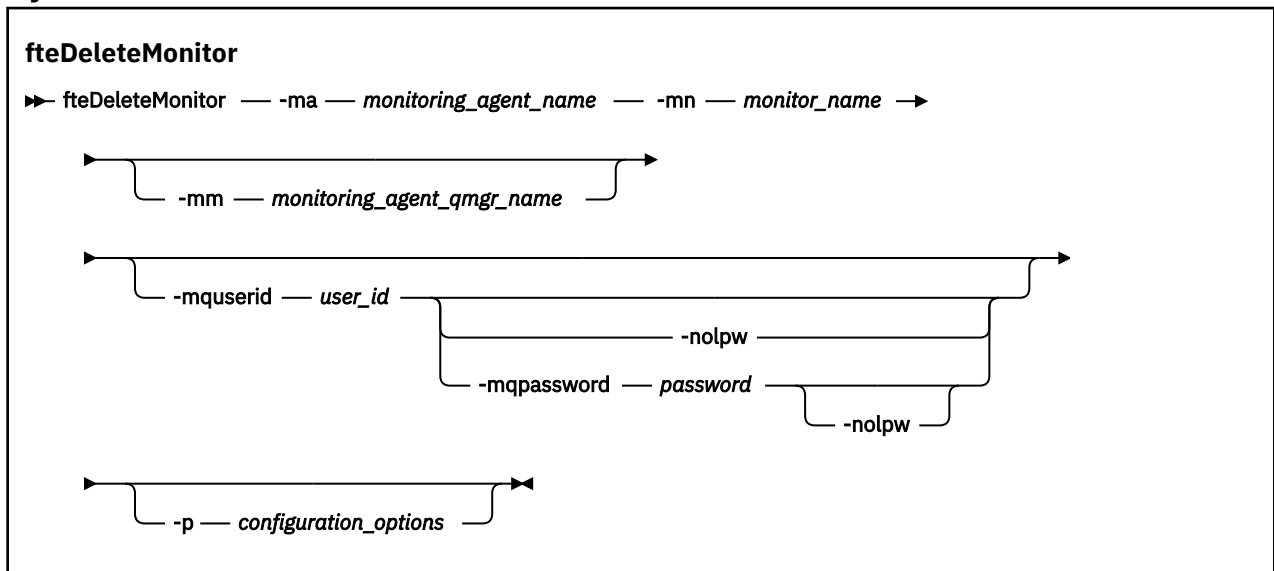
### **Purpose**

Use the **fteDeleteMonitor** command to stop monitoring a resource and remove the monitor's definition from the monitoring agent. When you run this command, no more polls of the resource occur and no further tasks are started.

You can run the **fteDeleteMonitor** command from any system that can connect to the IBM MQ network and subsequently route to the agent's queue manager. Specifically for the command to run, you must have installed a Managed File Transfer component (either Service or Agent) on this system and you must have configured this system's Managed File Transfer to communicate with the IBM MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

### **Syntax**



### **Parameters**

#### **-ma monitoring\_agent\_name**

Required. The name of the agent that performs the resource monitoring. This monitoring agent must also have been the source agent for the file transfer that you wanted to trigger.

#### **-mn monitor\_name**

Required. The name that you assigned to this resource monitor. You can delete a resource monitor and then create a new monitor with the same name.

#### **-mm monitoring\_agent\_qmgr\_name**

Optional. The name of the monitoring agent's queue manager. Because the monitoring agent and the source agent of the transfer the monitor triggered must be same, this queue manager is also your source agent's queue manager.

#### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

**-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using [MQCSP](#) authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

**V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**-p configuration\_options**

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

**-? or -h**

Optional. Displays command syntax.

**Example**

In this example, the resource monitor MONITOR1 with a monitoring (and file transfer source agent) AGENT1 is deleted:

```
fteDeleteMonitor -ma AGENT1 -mm QM_JUPITER -mn MONITOR1
```

**Return codes****0**

Command completed successfully.

**1**

Command ended unsuccessfully.

**Related tasks**

[Monitoring MFT resources](#)

**Related reference**

[“fteCreateMonitor \(create an MFT resource monitor\)” on page 2026](#)

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

[“fteListMonitors \(list MFT resource monitors\)” on page 2089](#)

Use the **fteListMonitors** command to list all of the existing resource monitors in a Managed File Transfer network using the command line.

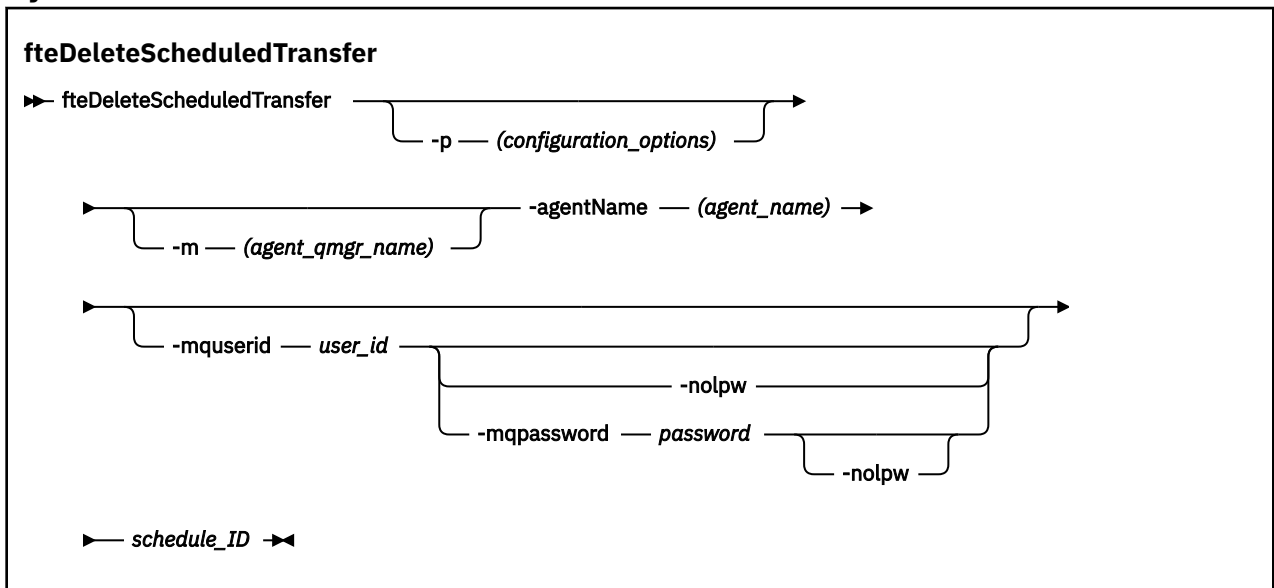
## fteDeleteScheduledTransfer (delete a scheduled MFT transfer)

### Purpose

Use the **fteDeleteScheduledTransfer** command to delete a Managed File Transfer scheduled transfer that you have previously created either using the command line or the IBM MQ Explorer.

Specify the optional **-p** parameter for this command only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in `installation.properties` are used. See [Configuration options](#) for more information.

### Syntax



### Parameters

#### **-p configuration\_options**

Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which scheduled transfer you want to delete. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

If you do not specify this parameter, the configuration options based on the default coordination queue manager are used.

#### **-m agent\_qmgr\_name**

Optional. The name of the queue manager that the source agent is connected to. If you do not specify this parameter, the agent's queue manager is determined from the configuration options in use.

#### **-agentName agent\_name**

Required. The name of the source agent that you want to delete the scheduled transfer from.

#### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

#### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.



**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using **MQCSP** authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

**V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password which, will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### **schedule\_ID**

Required. The ID of the scheduled transfer that you want to delete.

You can find the schedule ID by running the [fteListScheduledTransfers](#) command against the name of the source agent.

### **-? or -h**

Optional. Displays command syntax.

### **Example**

In this example, a scheduled transfer on source agent AGENT2 with the ID 27 is deleted:

```
fteDeleteScheduledTransfer -agentName AGENT2 27
```

### **Return codes**

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

### **Related tasks**

[Creating a scheduled file transfer](#)

### **Related reference**

[“fteListScheduledTransfers \(list all scheduled transfers\)”](#) on page 2093

Use the **fteListScheduledTransfers** command to list all of the Managed File Transfer transfers that you previously created using the command line or the IBM MQ Explorer.

## **fteDeleteTemplates (delete an MFT template)**

Use the **fteDeleteTemplates** command to delete an existing Managed File Transfer template from a coordination queue manager.

### **Purpose**

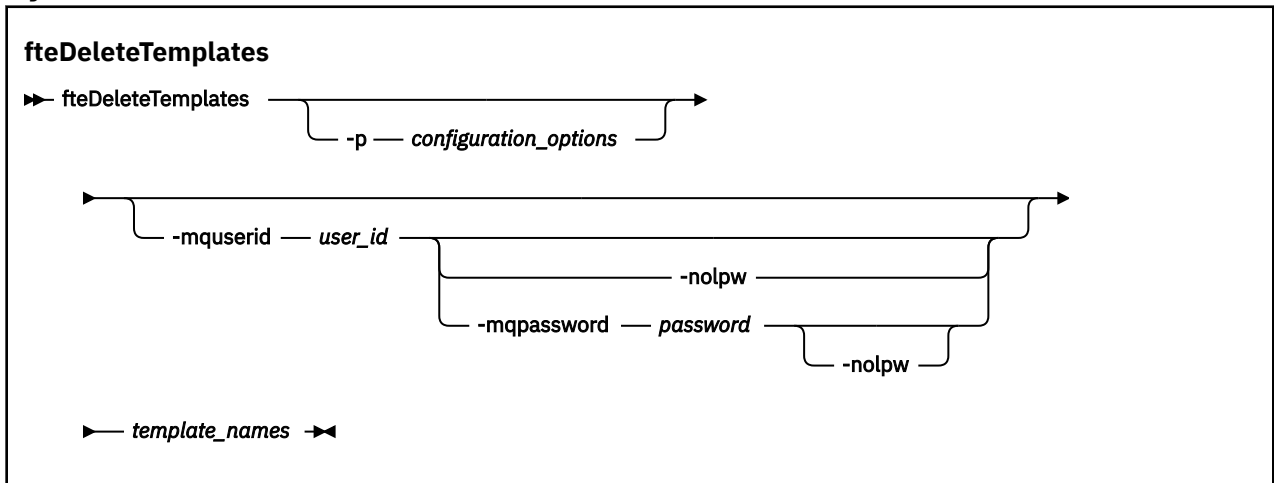
The **fteDeleteTemplates** command removes one or more file transfer templates from a coordination queue manager. When you run this command a request is passed to the IBM MQ system to remove the templates from the coordination queue manager so that the templates are no longer available to the IBM MQ Explorer or the command line. The templates you are deleting might continue to be accessed for a brief interval after the command completes until the IBM MQ system actions the request.

You can run the **fteDeleteTemplates** command from any system that can connect to the IBM MQ network and subsequently route to the coordination queue manager. Specifically for the command to run, you must have installed Managed File Transfer on this system and you must have configured this system's

Managed File Transfer to communicate with the IBM MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

## Syntax



## Parameters

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options to use to delete the template. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the coordination queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### **template\_names**

Required. Specify one or more template names that you want to delete. Specify the name as displayed by the **fteListTemplates** command.

**-? or -h**

Optional. Displays command syntax.

### Example

In this example, the template STANDBY is deleted:

```
fteDeleteTemplates STANDBY
```

### Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

### Related concepts

[Working with file transfer templates](#)

### Related tasks

[Creating a file transfer template using IBM MQ Explorer](#)

### Related reference

[“fteCreateTemplate \(create new file transfer template\)” on page 2033](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn *template\_name*** parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

[“fteListTemplates \(list available MFT transfer templates\)” on page 2095](#)

Use the **fteListTemplates** command to list the available Managed File Transfer transfer templates on a coordination queue manager.

## fteDisplayVersion (display installed version of MFT)

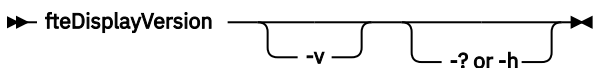
Use the **fteDisplayVersion** command to display the version of Managed File Transfer (MFT) that you have installed.

### Purpose

You might be asked to run the **fteDisplayVersion** command by an IBM Service Representative to help with problem determination.

### Syntax

**fteDisplayVersion**


►► fteDisplayVersion 

### Parameters

**-v**

Optional. Displays a verbose amount of information about the product version.

The precise details that are displayed when you specify the **-v** parameter might vary between product releases. You are not recommended to rely on specific information being available in the output from the `fteDisplayVersion -v` command.

 On z/OS, **-v** displays the value of the **productId** property, if the product Id has been specified.

### **-? or -h**

Optional. Displays command syntax.

### **Example with no parameters specified**

In this example, the **fteDisplayVersion** command is specified with no parameters.

```
fteDisplayVersion
```

The output from this command is the product version level. For example, this is the output for IBM MQ 9.3.0:

```
fteDisplayVersion -v
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Name:      IBM MQ Managed File Transfer
Version:   9.3.0.0
```

### **Example with -v parameter specified**

In this example, the **fteDisplayVersion** command is specified with the **-v** parameter.

```
fteDisplayVersion -v
```


The output from this command includes more detailed information about the product version. For example:

 IBM MQ 9.3.1

```
fteDisplayVersion -v
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Name:      IBM MQ Managed File Transfer
Version:   9.3.1.0
Level:     p931-dfct-USER-L220717.2
Platform:  Windows 10 (10.0)
Architecture: amd64
JVM:      JRE 1.8.0 Windows 10 amd64-64-Bit Compressed References 20220427_27745 (JIT enabled, AOT
enabled)
          OpenJ9    - b15041a
          OMR       - 3671a9f
          IBM       - 1b0232b
Product:   C:\Program Files\IBM\MQ
Configuration: C:\ProgramData\IBM\MQ\mqft
```

IBM MQ Components:

```
Name:      Common Services for Java Platform, Standard Edition
Version:   9.3.1.0
Level:     p931-dfct-USER-L220704.3
```

**Note:**  Before IBM MQ 9.3.1, the output from the **fteDisplayVersion** command included components that MFT does not use. From IBM MQ 9.3.1, these components are no longer included in the output.

 Long Term Support and Continuous Delivery before IBM MQ 9.3.1:

```
fteDisplayVersion -v
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Name:      IBM MQ Managed File Transfer
Version:   9.3.0.0
Level:     p930-L220526
Platform:  Linux (4.18.0-348.23.1.el8_5.s390x)
Architecture: s390x
JVM:      JRE 1.8.0 Linux s390x-64-Bit Compressed References 20220303_23850 (JIT enabled,
```

```
AOT enabled)
      OpenJ9   - 6890476
      OMR     - 1e70123
      IBM     - 1b0232b
Product: C:\Program Files\IBM\MQ
Configuration: C:\ProgramData\IBM\MQ\mqft
```

#### IBM MQ Components:

```
Name:      IBM MQ JMS Provider
Version:   9.3.0.0
Level:    p930-L220526

Name:      IBM MQ JMS Provider
Version:   9.3.0.0
Level:    p930-L220526

Name:      Common Services for Java Platform, Standard Edition
Version:   9.3.0.0
Level:    p930-L220526

Name:      IBM MQ classes for Java
Version:   9.3.0.0
Level:    p930-L220526

Name:      Java Message Service Client
Version:   9.3.0.0
Level:    p930-L220526

Name:      IBM MQ classes for Java Message Service
Version:   9.3.0.0
Level:    p930-L220526

Name:      Java Message Service Client
Version:   9.3.0.0
Level:    p930-L220526

Name:      IBM MQ classes for Java Message Service
Version:   9.3.0.0
Level:    unknown
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## **ftelistagents** (list the MFT agents for a coordination queue manager)

Use the **ftelistagents** command to list all of the Managed File Transfer agents that are registered with a particular coordination queue manager.

### Purpose

You can run the **ftelistagents** command from any system that can connect to the coordination queue manager. The following details for each agent are directed to the standard output device (STDOUT):

- Agent name
- Agent queue manager
- If the agent is a protocol bridge agent, the agent name is appended with ( bridge )
- If the agent is a Connect:Direct bridge agent, the agent name is appended with (Connect:Direct bridge)
- Agent status

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see [The MFT coordination.properties file](#).

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see [Configuration options](#).

If an agent is not listed by the **fteListAgents** command, use the diagnosis flowchart in the following topic to locate and fix the problem: [What to do if your MFT agent is not listed by the fteListAgents command](#).

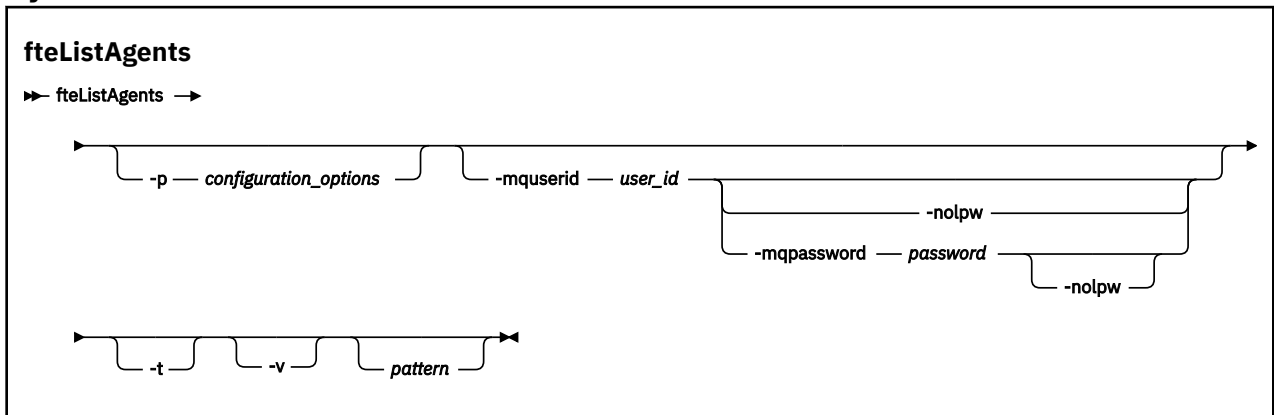
## Agent status information

The agent status information produced by this command is generated from the status messages that the agent publishes to the SYSTEM.FTE topic. These messages are described in the topic “[MFT agent status message format](#)” on page 2615. The status information produced by the **fteListAgents** command gives the agent status at the time when the last status message was published.

The frequency of these status messages depends on the value of the **agentStatusPublishRateLimit** property. For more details about this property, see [The MFT agent.properties file](#).

If the **Status Age** is surrounded by parenthesis, this indicates that the value is negative. This situation occurs if the system time of the machine, where the agent is running, is ahead of the system time of the coordination queue manager machine.

## Syntax



## Parameters

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to issue the request to list agents. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the coordination queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

**V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**-v**

Optional. Specifies verbose mode. Verbose mode generates additional output that shows the number of current managed transfers for each agent in the form Source/Destination, where:

- Source is the current number of source transfers and queued transfers for the agent.
- Destination is the current number of destination transfers.

The current transfer information is obtained from the agent status publication, which is described in the “MFT agent status message format” on page 2615 topic. As a result, this transfer information is only accurate to within the setting for the `agentStatusPublishRateLimit` agent property value (which defaults to 30 seconds).

**-t**

Optional. Specifies terse mode. From IBM MQ 9.1, the output includes the **Status Age** column by default. If you do not want to see the **Status Age** information, you can issue the command with the **-t** parameter to hide the column. For more information, see [What to do if an agent is shown as being in an UNKNOWN state](#).

**pattern**

Optional. The pattern to use to filter the list of Managed File Transfer agents. This pattern is matched against the agent name. Asterisk (\*) characters are interpreted as wildcards, that match any value, including zero characters.

**Linux** **AIX** On AIX and Linux systems, you must escape special characters like the asterisk (\*) and the number sign (#) with quotation marks ( ' ') or double quotation marks ( " ") if you want them to be handled as literals. If you do not escape these characters, they are interpreted according to their meaning on the specific AIX or Linux system.

If you do not specify this parameter, all agents registered with the coordination queue manager are listed.

**-? or -h**

Optional. Displays command syntax.

**Example**

In this example, all of the agents registered on the queue manager detailed in the configuration options with names beginning with B are listed:

```
fteListAgents "B*"
```

In this example, agents that are registered with the coordination queue manager QM\_EUROPE (the non-default coordination queue manager) are listed in verbose mode:

```
fteListAgents -p QM_EUROPE -v
```

The output from this command is as follows:

Agent Name:	Queue Manager Name:	Transfers: (Source/Destination)	Status:
BERLIN	QM_BERLIN	7/0	RUNNING

LONDON	QM_LONDON	0/0	RUNNING
MADRID	QM_MADRID	0/1	UNREACHABLE

For a list of the possible agent status values and their meanings, see the topic [“MFT agent status values”](#) on page 2485.

In this example, all agents that are registered with the coordination queue manager and that have names beginning with BRIDGE are listed in verbose mode:

```
fteListAgents -v "BRIDGE*"
```

The output from this command is as follows:

```
C:\Program Files\IBM\WMQFTE\bin>fteListAgents -v
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Name:                               Queue Manager Name:   Transfers:   Status:
                                           (Source/Destination)
BRIDGE_FTP ( bridge )                     QM_JUPITER            0/0          STOPPED
BRIDGE_CD1 (Connect:Direct bridge)        QM_JUPITER            0/0          STOPPED
```

From IBM MQ 9.1.4 the output from the command displays HA by an agent name if that agent is highly available. You must set **highlyAvailable=true** in the [agent.properties](#) file for an agent to be started in highly available mode. Note that HA is displayed, even if there are no standby instances running.

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Command executed at 2019-05-15 13:21:08 IDT
Coordination queue manager time 2019-05-15 07:51:08 UTC
Agent Name:                               Queue Manager Name:   Status:        Status Age:
IMQFT02 ( bridge )(HA) MFTQM              STOPPED        8:51:17
SRC (HA)                               MFTQM              READY          0:04:50
DEST                                    MFTQM              READY          0:05:50
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related concepts

[What to do if an agent is shown as being in an UNKNOWN state](#)

## Related tasks

[Listing MFT agents](#)

## Related reference

[“MFT agent status values”](#) on page 2485

The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

[“fteShowAgentDetails \(display MFT agent details\)”](#) on page 2129



Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

## fteListMonitors (list MFT resource monitors)

Use the **fteListMonitors** command to list all of the existing resource monitors in a Managed File Transfer network using the command line.

### Purpose

The **fteListMonitors** command lists existing resource monitors. You can filter the command output by specifying an agent name and a resource monitor name.

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see [The MFT coordination.properties file](#).

You can use the **-ox** parameter to export a resource monitor to an XML file. For more information on how to use this XML file, see [“fteCreateMonitor \(create an MFT resource monitor\)” on page 2026](#).

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see [Configuration options](#).

### Resource monitor names

Resource monitor names can contain characters that might not be valid for file names. From IBM MQ 9.1, if a resource monitor name contains any of the following characters, the **fteListMonitors -od** command converts that character to its ASCII equivalent:

- "\" (Backslash) = %5C
- "/" (Forward slash) = %2F
- ":" (Colon) = %3A
- "<" (Less than) = %3C
- ">" (Greater than) = %3E
- "\"" (Double quotes) = %22
- "|" (Pipe) = %7C

For example, a resource monitor with name:

```
SRC.TEST \ (TESTING-TEST\)
```

is saved to a file called:

```
SRC.TEST %5C (TESTING-TEST%5C)
```

In addition, from IBM MQ 9.1, you no longer have to use an escape character when specifying any special characters while using the **fteListMonitors -ma <agent name> -mn <monitor name>** command.

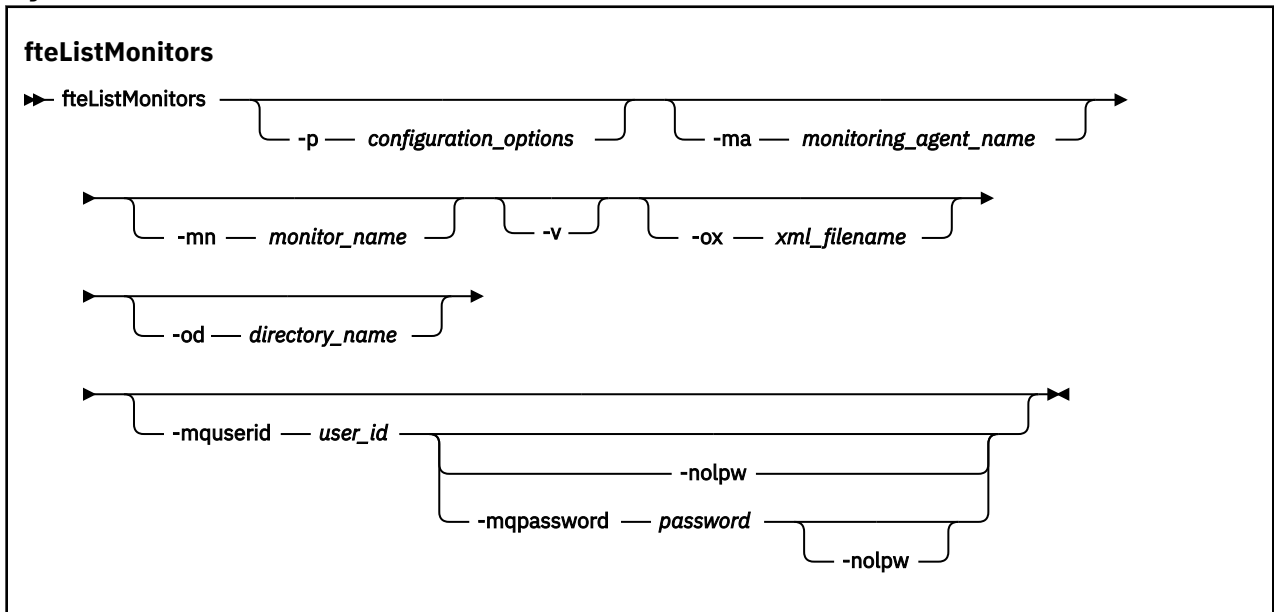
For example, where in earlier releases the command used to be:

```
fteListMonitors -ma SRC -mn "TEST \ (TESTING-TEST\)"
```

from IBM MQ 9.1 you enter:

```
fteListMonitors -ma SRC -mn "TEST (TESTING-TEST)"
```

## Syntax



## Parameters

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files that are associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### **-ma monitoring\_agent\_name**

Optional. Filters resource monitors by agent name by using the pattern that you provide as input. Asterisk (\*) characters are interpreted as wildcards that match zero or more characters. If you do not specify the **-ma** parameter, all resource monitors associated with all agents for the default coordination queue manager are listed by default.

### **-mn monitor\_name**

Optional. Filters resource monitors by monitor name by using the pattern that you provide as input. Asterisk (\*) characters are interpreted as wildcards that match zero or more characters. If you do not specify the **-mn** parameter, all resource monitors associated with all agents for the default coordination queue manager are listed by default.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the coordination queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**-v**

Optional. Generates verbose output that includes additional information about the status of the resource monitor, including whether the resource monitor is started or stopped, the directory resource path that is being monitored and the trigger conditions.

**-ox xml\_filename**

Optional. You must specify this parameter in combination with the **-ma** and **-mn** parameters. Exports the resource monitor to an XML file that can then be used by the **fteCreateMonitor** command and the **-ix** parameter.

The **-ox** parameter must not be combined with the **-od** parameter.

**-od directory\_name**

Optional. Exports multiple resource monitor definitions to the specified directory. Each resource monitor definition is saved to a separate XML file with a name in the format *agent\_name.monitor\_name.xml*. You must specify a valid target directory for the XML files, otherwise an error message is displayed. This parameter must not be combined with the **-ox** parameter.

**-? or -h**

Optional. Displays command syntax.

### Examples: list resource monitors

In this example, all resource monitors associated with the monitoring agent (and source agent for the file transfers associated with the resource monitor) AGENT1 are listed:

```
fteListMonitors -ma AGENT1
```

The output from this command is as follows:

```
C:/Users/Administrator>fteListMonitors -ma AGENT1
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Name:      Monitor Name:      Resource Type:
AGENT1           MONITOR1           Directory
```

In the following example, the command includes the **-v** parameter, which generates verbose output that includes additional information about the status of the resource monitor:

```
fteListMonitors -ma AGENT1 -v
```

In this case, the output from the command is as follows:

```
C:/Users/Administrator>fteListMonitors -ma AGENT1 -v
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Monitor Information:
  Name:           MONITOR1
  Agent:          AGENT1
  Status:         Stopped
  Resource Type:  Directory
  Resource:       C:\src
  Poll interval:  2 seconds
  Batch size:     1
  Condition:      Match
  Pattern:        * (wildcard)
```

### Example: export one resource monitor to an XML file

In this example, a single resource monitor, MONITOR1, on AGENT1 is exported to the XML file filename1.xml by specifying an XML file name with the **-ox** parameter:

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -ox filename1.xml
```

### Example: export one resource monitor to a specified directory

In this example, a single resource monitor, MONITOR1, on AGENT1 is exported to the directory that is specified by the **-od** parameter. Except for the difference in the XML file name format, this example is similar to using the **-ox** parameter.

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -od /usr/mft/resmonbackup
```

### Examples: export a batch of resource monitors to an XML file in a specified directory

In all of the following examples, the resource monitors are exported to the directory that is specified by the **-od** parameter. Each resource monitor definition is saved to separate XML file with a name in the format *agent name.monitor name.xml*.

In this example, all resource monitors are exported to the specified directory:

```
fteListMonitors -od /usr/mft/resmonbackup
```

In this example, all resource monitors on AGENT1 are exported to the specified directory:

```
fteListMonitors -ma AGENT1 -od /usr/mft/resmonbackup
```

You can use wildcard matching to define which resource monitors to export by using an asterisk character (\*) when you specify a pattern to match to agent names, or monitor names, or both.

In this example, all resource monitors on AGENT1 with names that match the pattern MON\* are exported to the specified directory:

```
fteListMonitors -ma AGENT1 -mn MON* -od /usr/mft/resmonbackup
```

In this example, all resource monitors on agents with names that match the pattern AGEN\* are exported to the specified directory:

```
fteListMonitors -ma AGEN* -od /usr/mft/resmonbackup
```

In this example, all resource monitors with names that match the pattern MON\* on agents with names that match the pattern AGEN\* are exported to the specified directory:

```
fteListMonitors -ma AGEN* -mn MON* -od /usr/mft/resmonbackup
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related tasks

[Monitoring MFT resources](#)

[Backing up and restoring MFT resource monitors](#)

## Related reference

[“fteCreateMonitor \(create an MFT resource monitor\)” on page 2026](#)

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) by using Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

“[fteDeleteMonitor \(delete an MFT resource monitor\)](#)” on page 2078

Use the **fteDeleteMonitor** command to stop and delete an existing Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

## fteListScheduledTransfers (list all scheduled transfers)

Use the **fteListScheduledTransfers** command to list all of the Managed File Transfer transfers that you previously created using the command line or the IBM MQ Explorer.

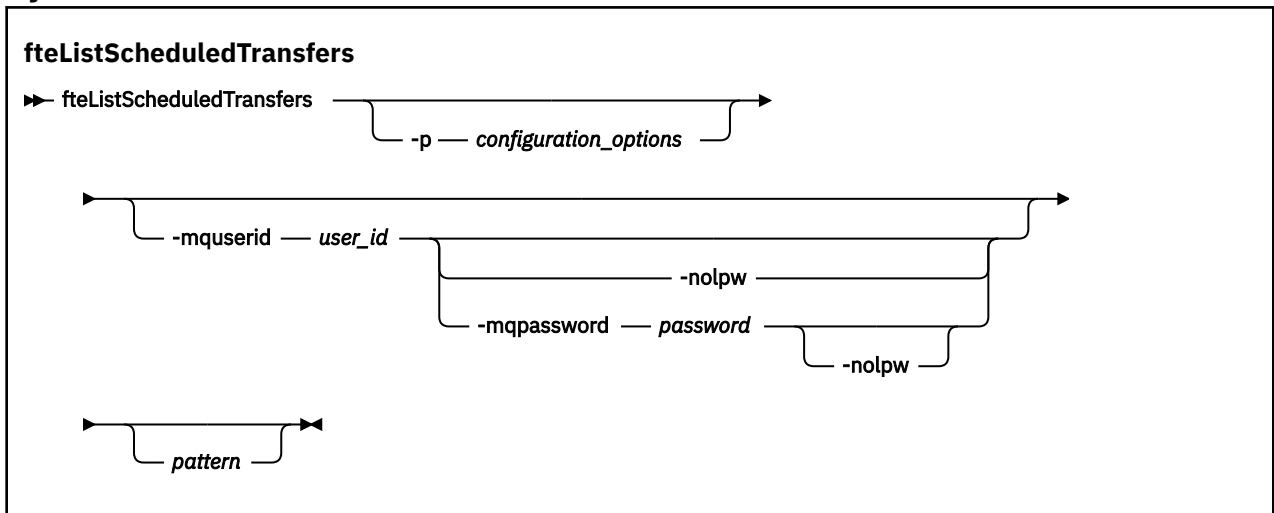
### Purpose

You can either list all scheduled transfers based on source agent names or based on the coordination queue manager.

Specify the optional **-p** parameter for this command only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in `installation.properties` are used. See [Configuration options](#) for more information.

When you run the **fteListScheduledTransfers** command, any scheduled transfer that has a transfer definition with a semantically incorrect date and time combination causes error messages to be displayed. From IBM MQ 9.3.0, these messages are BFGCL0810E messages that include the schedule ID of the invalid scheduled transfer. You can then run the **fteDeleteScheduledTransfer** command with the **schedule\_ID** parameter to delete the invalid scheduled transfer.

### Syntax



### Parameters

#### **-p configuration\_options**

Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which agents you want to list scheduled transfers for. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

If you do not specify this parameter, the configuration options based on the default coordination queue manager are used.

#### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the coordination queue manager.

#### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

**V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### **pattern**

Optional. The pattern to use to filter the list of Managed File Transfer scheduled transfers. This pattern is matched against the source agent name. Asterisk (\*) characters are interpreted as wildcards that match zero or more characters.

If you do not specify this parameter, all of the scheduled transfers registered with the coordination queue manager are listed by default.

### **-? or -h**

Optional. Displays command syntax.

### **Example**

In this example, all of the scheduled transfers with source agents that match the pattern \*2 are listed:

```
fteListScheduledTransfers "*2"
```

This example command produces the following output. The schedule start time and next transfer time are displayed in Coordinated Universal Time (UTC):

```
Schedule Identifier:      1
Source Agent Name:      AGENT2
Source File Name:       C:/export/Test/workspace/A.exe
Conversion Type:        binary
Destination File Name:  C:/import/Test/workspace/B001.zzx
Destination Agent Name: AGENT1
Schedule Start Time:    2008-10-23T16:08+0100
Next Transfer:          2008-10-23T16:08+0100
Schedule Time Base:     source
Repeat Interval:        minutes
Repeat Frequency:       1
Repeat Count:           30
```

### **Return codes**

- 0** Command completed successfully.
- 1** Command ended unsuccessfully.

### **Related tasks**

[Creating a scheduled file transfer](#)

### **Related reference**

[“fteDeleteScheduledTransfer \(delete a scheduled MFT transfer\)” on page 2080](#)

## fteListTemplates (list available MFT transfer templates)

Use the **fteListTemplates** command to list the available Managed File Transfer transfer templates on a coordination queue manager.

### Purpose

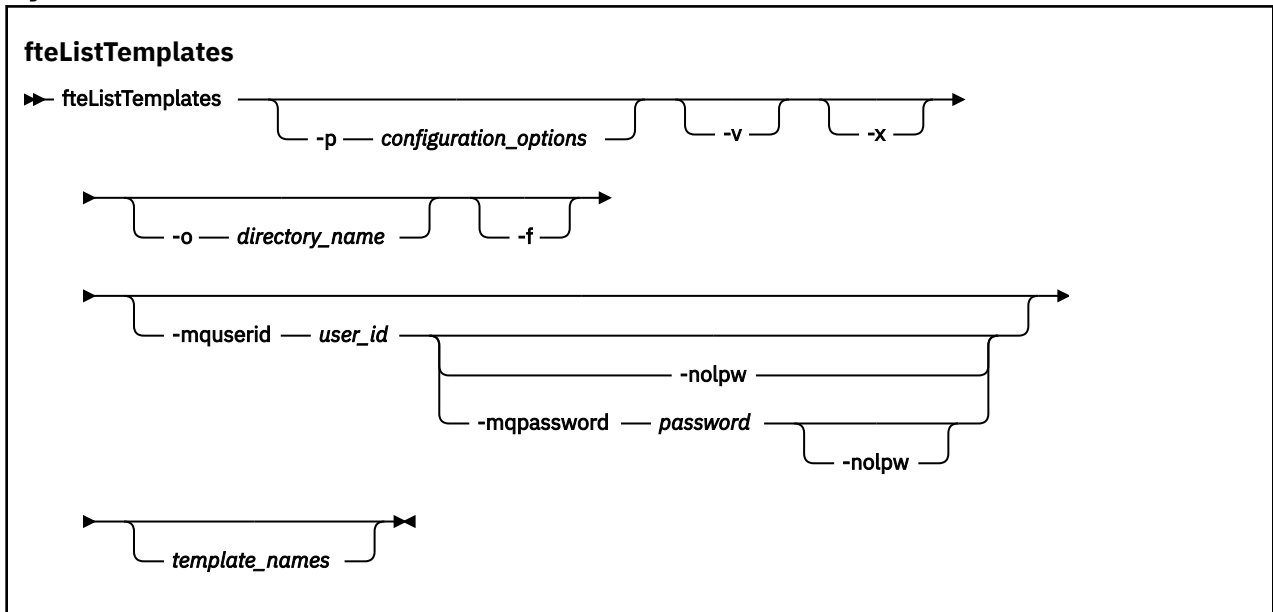
This command lists either all template names or a filtered selection of template names. The output format of the list can be any of the following:

- Template names only (default behavior)
- Template names with a summary of the templates (verbose mode)
- Complete XML message describing the templates (**-x** and **-o** parameters)

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see [The MFT coordination.properties file](#).

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

### Syntax



### Parameters

#### -p

Optional. This parameter determines the set of configuration options to use to delete the template. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

#### -v

Optional. Specifies verbose mode and provides a short summary of each matching template. This parameter is ignored if you have also specified the **-x** parameter.

The **-v** parameter includes a summary of each template. For example:

```
Template Name: STANDBY
```

```

Source Agent Name: AGENT1
Source QMgr: QM_JUPITER
Destination Agent Name: AGENT2
Destination QMgr: QM_NEPTUNE
Transfer Priority: 0
Transfer file specification
File Item Details
  Mode: binary
  Checksum: MD5
  Source File:
    C:\payroll_reports\*.xls
  Recursive: false
  Disposition: leave
  Destination File:
    C:\payroll_backup\*.xls
  Type: file
  Exist: error

```

If you do not specify the **-v** parameter, the default output mode is to list the matching templates names.

**-x**

Optional. Provides an XML-formatted message for each matching template. This parameter is ignored unless you also specify the **-o** parameter.



**Attention:** The XML-formatted messages are not compatible with the **fteCreateTemplate** command tools.

**-o directory\_name**

Optional. Sends the XML formatted-message to files in the named directory. One file for each template is created and each file has the same name as the template with an `.xml` suffix. This parameter is ignored unless you also specify the **-x** parameter.

**-f**

Optional. Forces any existing output file to be overwritten. This parameter is ignored unless you also specify the **-o** parameter. If you do not specify **-f** but you do specify the name of an existing output file, the default behavior is to report an error and continue.



**-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the coordination queue manager.

**-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

 **V 9.3.0**  **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

 **V 9.3.0**  **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**template\_names**

Optional. A list of one or more template names to be listed. A template name can include an asterisk as a wildcard that matches zero or more characters. Depending on your operating system, you might need to enclose any template names that include wildcard character in quotation marks (" ") or single quotation marks (' ') to avoid shell expansion. Shell expansion can cause unexpected behavior.



If you do not specify anything for *template\_names*, the default is to list all templates.

**-? or -h**

Optional. Displays command syntax.

**Example**

In this example, all the templates with names starting with ST are listed:

```
fteListTemplates "ST*"
```

This example creates the template STANDBY as an XML-formatted message to the file STANDBY.xml in the current directory:

```
fteListTemplates -x -o . STANDBY
```

This command creates the following output in STANDBY.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <transferTemplate id="1864c1dd-ba02-4b34-bda9-dc6862448418" version="3.00">
  <name>STANDBY</name>
  <sourceAgentName>AGENT1</sourceAgentName>
  <sourceAgentQMgr>QM_JUPITER</sourceAgentQMgr>
  <sourceAgentQMgrHost>null</sourceAgentQMgrHost>
  <sourceAgentQMgrPort>-1</sourceAgentQMgrPort>
  <sourceAgentQMgrChannel>null</sourceAgentQMgrChannel>
  <destinationAgentName>AGENT2</destinationAgentName>
  <destinationAgentQMgr>QM_NEPTUNE</destinationAgentQMgr>
- <fileSpecs>
  - <item checksumMethod="MD5" mode="binary">
    - <source disposition="leave" recursive="false">
      <file>C:\payroll_reports\*.xls</file>
    </source>
    - <destination exist="error" type="file">
      <file>C:\payroll_backup\*.xls</file>
    </destination>
  </item>
</fileSpecs>
<priority>0</priority>
</transferTemplate>
```

**Return codes**

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

**Related concepts**

[Working with file transfer templates](#)

**Related tasks**

[Creating a file transfer template using IBM MQ Explorer](#)

[Backing up a file transfer template definition](#)

**Related reference**

[“fteCreateTemplate \(create new file transfer template\)” on page 2033](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use.

The only required parameter is the **-tn template\_name** parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

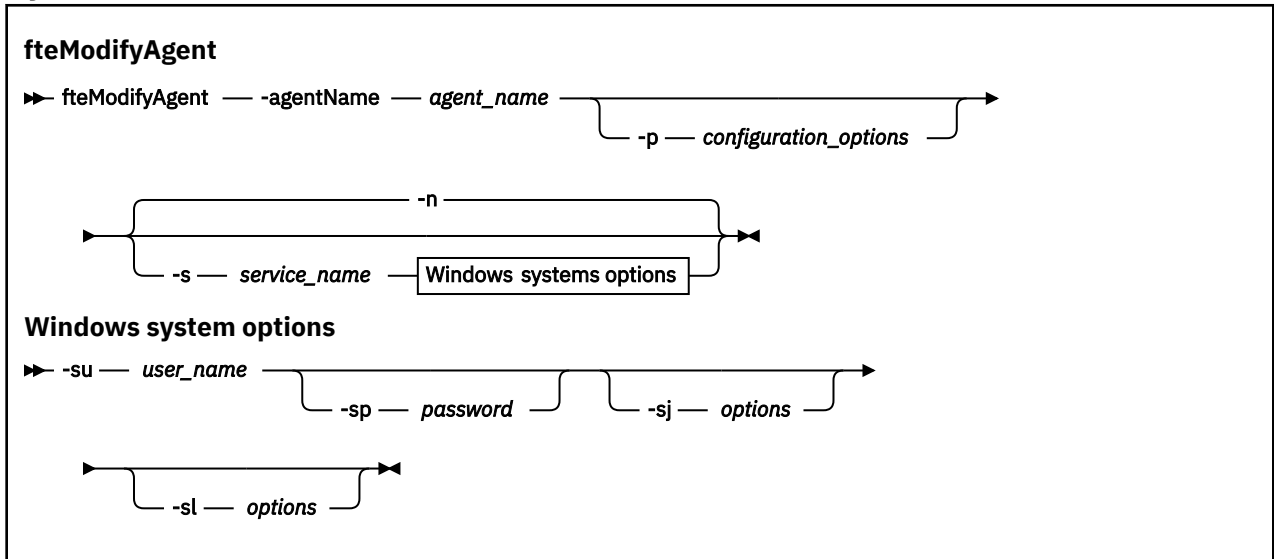
[“fteDeleteTemplates \(delete an MFT template\)” on page 2081](#)

Use the **fteDeleteTemplates** command to delete an existing Managed File Transfer template from a coordination queue manager.

## Windows **fteModifyAgent (run an MFT agent as a Windows service)**

The **fteModifyAgent** command modifies an existing agent so that it can be run as a Windows service. This command is only available on Windows, and must be run by a user who is an IBM MQ administrator and a member of the mqm group.

### Syntax



### Parameters

#### **-agentName agent\_name**

Required. The name of the agent you want to modify.

#### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to modify the agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteModifyAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

#### **-s service\_name**

Optional. Indicates that the agent is to run as a Windows service. If you do not specify *service\_name*, the service is named mqmftAgentAGENTQMGR, where *AGENT* is the agent name and *QMGR* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Agent AGENT@QMGR**.

**Note:** If the redistributable agent is going to run as a Windows service, then the **BFG\_DATA** environment variable needs to be set in the system environment for the service to work.

#### **-su user\_name**

Optional. When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service should run. To run the agent using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see [Troubleshooting an MFT agent or logger running as a Windows service](#).

This parameter is required when **-s** is specified.

#### **-sp password**

Optional. This parameter is only valid when **-s** is specified.

#### **-sj options**

Optional. When the agent is started as a Windows service, this parameter defines a list of options in the form of **-D** or **-X** that will be passed to the Java Virtual Machine (JVM). The options are separated using the number sign (**#**) or semicolon (**;**) character. If you need to embed any **#** or **;** characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified.

For more information about the way in which the **fteModifyAgent** command handles the validation of updates to the JVM options see [How agent and logger JVM options are processed](#).

#### **-sl options**

Optional. Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified.

#### **-n**

Optional. Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither the **-s** nor the **-n** option is specified, then the agent is configured as a normal Windows process.

#### **-? or -h**

Optional. Displays command syntax.

### **Example**

In this example, AGENT1 is modified to run as a Windows service:

```
fteModifyAgent -agentName AGENT1 -s -su fteuser -sp ftepassword
```

In this example, AGENT1 is modified to remove the Windows service:

```
fteModifyAgent -agentName AGENT1
```

You must stop the agent you want to modify, using the [fteStopAgent](#) command, before you can run the [fteModifyAgent](#) command.

### **Return codes**

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

### **Related concepts**

[Troubleshooting an MFT agent or logger running as a Windows service](#)

### **Related tasks**

[Starting an MFT agent as a Windows service](#)

### **Related reference**

[“fteCreateAgent \(create an MFT agent\)” on page 1999](#)

The **fteCreateAgent** command creates a Managed File Transfer Agent and its associated configuration.

“[fteModifyLogger \(run an MFT logger as a Windows service\)](#)” on page 2100

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

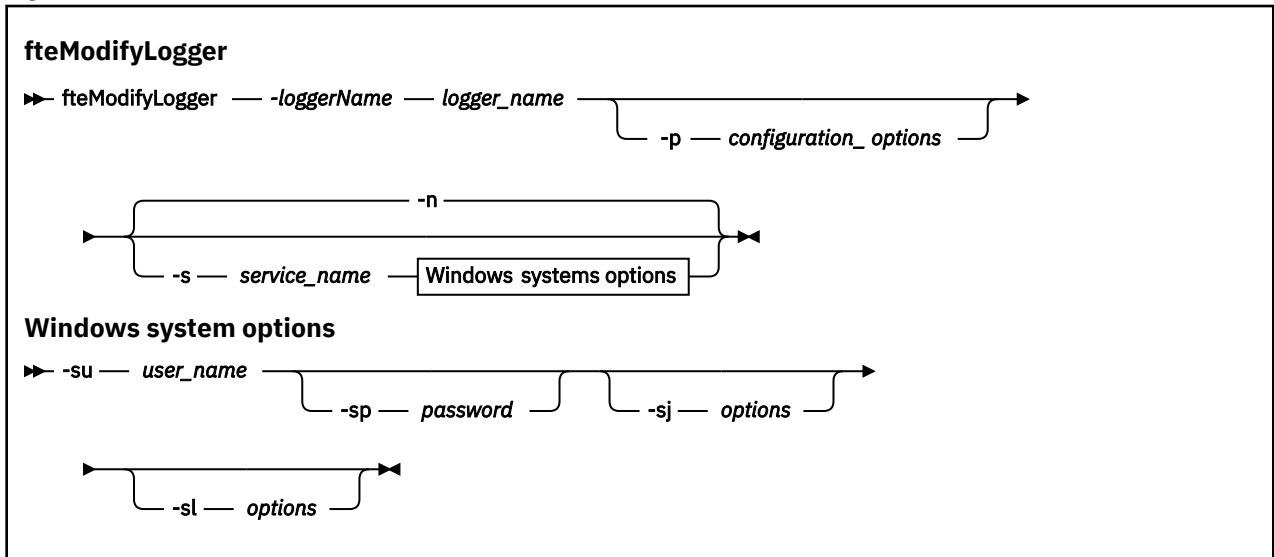
## Windows **fteModifyLogger (run an MFT logger as a Windows service)**

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

### Purpose

A stand-alone logger, whether for a file or for a database, is shown as "Managed File Transfer logger for property set *logger\_name@logger\_qm*" in the **Name** column of the **Services** application. The value of *logger\_qm* is the name of the command queue manager of the logger.

### Syntax



### Parameters

#### **-loggerName** *logger\_name*

Required. The name of the Managed File Transfer logger you want to modify.

#### **-p** *configuration\_options*

Optional. This parameter determines the set of configuration options that is used to modify the logger. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteModifyLogger** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

#### **-s** *service\_name*

Optional. Indicates that the logger is to run as a Windows service. If you do not specify *service\_name*, the service is named `mqmftLogger1LOGGERQMGR`, where *LOGGER* is the logger name and *QMGR* is your logger queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **Managed File Transfer Logger *LOGGER@QMGR***.

**-su user\_name**

Required when **-s** is specified. Specifies the name of the account under which the Windows service should run. To run the agent using a Windows domain user account, specify the value in the form `DomainName\UserName`. To run the service using an account from the local built-in domain, specify the value in the form `UserName`.

The Windows user account that you specify using the **-su** parameter must have the permission to log on as a service. For information about how to grant this permission, see [Troubleshooting an MFT agent or logger running as a Windows service](#).

**-sp password**

Optional. Only valid when **-s** is specified. Password for the user account set by the **-su** parameter.

If you do not specify this parameter when you specify the **-s** parameter, you are warned that you must set the password by using the Windows Services tool before the service can start successfully.

**-sj options**

Optional. Only valid when **-s** is specified. When the logger is started as a Windows service, this parameter defines a list of options in the form of **-D** or **-X** that will be passed to the JVM. The options are separated using the number sign (**#**) or semicolon (**;**) character. If you need to embed any **#** or **;** characters, put them inside single quotation marks (**'**).

For more information about the way in which the **fteModifyLogger** command handles the validation of updates to the JVM options see [How agent and logger JVM options are processed](#).

**-sl options**

Optional. Only valid when **-s** is specified. Sets the Windows service log level. Valid options are: `error`, `info`, `warn`, `debug`. The default is `info`. This option can be useful if you are having problems with the Windows service. Setting it to `debug` gives more detailed information in the service log file.

**-n**

Optional. Indicates that the logger is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither the **-s** nor the **-n** option is specified, then the logger is configured as a normal Windows process.

**-? or -h**

Optional. Displays command syntax.

**Example**

You must stop the logger by using the [fteStopLogger](#) command, before running the **fteModifyLogger** command.

In this example, a logger named `logger1` has previously been created. This command shows how the logger can be changed to run as a Windows service:

```
fteModifyLogger -loggerName logger1 -s -su fteuser -sp ftepassword
```

**Return codes**

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

**Related concepts**

[Troubleshooting an MFT agent or logger running as a Windows service](#)

**Related tasks**

[Starting an MFT agent as a Windows service](#)

## Related reference

[“fteStartLogger \(start an MFT logger\)” on page 2142](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

[“fteStopLogger \(stop an MFT logger\)” on page 2148](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

## fteObfuscate (encrypt sensitive data)

The **fteObfuscate** command encrypts sensitive data in credentials files. This stops the contents of credentials files being read by someone who gains access to the file. From IBM MQ 9.2.0, the command provides a more secure credentials protection method to improve the encryption.

## Purpose

User name and password properties in credentials files can be encrypted. These properties are transformed to a new related property, with a Cipher suffix. For example:

```
<!--
MQMFTCredentials properties before
-->
<tns:logger name="logger1" user="user1" password="passw0rd" />
<tns:file path="$HOME/trust.jks" password="passw0rd" />

<!--
MQMFTCredentials properties after
-->
<tns:logger name="logger1" userCipher="e71vKCg2pf" passwordCipher="se71vKCg" />
<tns:file path="$HOME/trust.jks" passwordCipher="e71vKCg2pf" />

<!--
ProtocolBridgeCredentials Properties before
-->
<tns:user name="Fred" serverUserId="fred" serverPassword="passw0rd" />

<!--
ProtocolBridgeCredentials properties after
-->
<tns:user name="Fred" serverUserIdCipher="e51vVCg2pf" serverPasswordCipher="se51vBCg" />

<!--
ConnectDirectCredentials properties before
-->
<tns:user name="fteuser" ignorecase="true" pattern="wildcard"
  cdUserId="cdUser" cdPassword="cdPassword" pnodeUserId="pnodeUser"
  pnodePassword="pnodePassword">
  <tns:snode name="snode1" pattern="wildcard" userId="snodeUser" password="snodePassword"/>
</tns:user>

<!--
ConnectDirectCredentials properties after
-->
<tns:user name="fteuser" ignorecase="true" pattern="wildcard"
  cdUserIdCipher="e71vKCg2pf" cdPasswordCipher="se71vKCg"
  pnodeUserIdCipher="2f1vgCg6df" pnodePasswordCipher="e71vKCg2pf">
  <tns:snode name="snode1" pattern="wildcard" userIdCipher="e51vVCg2pf" passwordCipher="se51vBCg"/>
</tns:user>
```

From IBM MQ 9.2.0, the preferred formats are:

## MFT

```
<tns:qmgr mqPasswordCipher="mqmftcred!1!kvAzYv/1aCMfSQ5igkFVmQ==!f4rX5KL7aFKHJ17Ln0X+0Q=="
mqUserIdCipher="mqmftcred!1!w2PQGhQcyq1NwYzGItz0VA==!Q40i2rRSEMGwrx6gnRFe8g=="
name="MFTQM" user="JOHNDOE"/>
```

## ProtocolBridgeCredentials

```
<tns:agent name="agent3">
  <tns:serverHost name="ftpsServer"
    keyStorePasswordCipher="mqmftcred!1!w2PQGhQcyq1NwYzGItz3VA==!
```

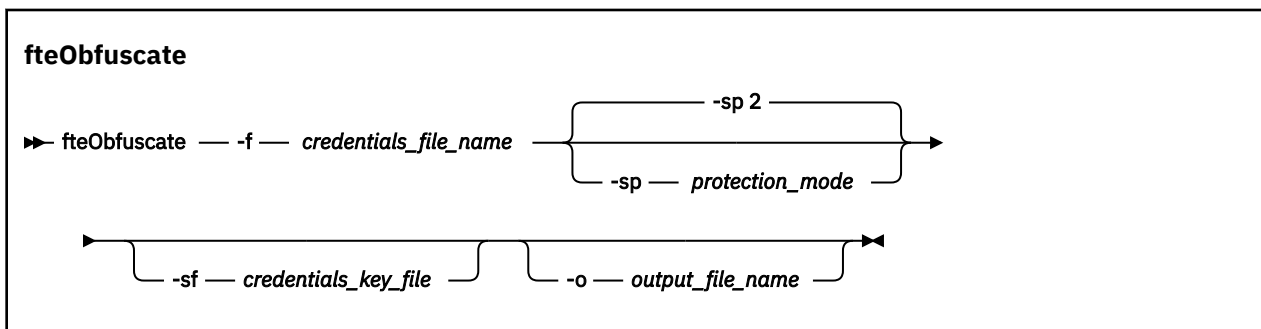
```
Q40i2rRSEMGwrx6gnRFe8g=="
trustStorePasswordCipher="mqmftcred!1!w2PQGhQcyq1NwYzGktz0VA==!Q40i2rRSEMGwrx6gnRFe8g==">
  </tns:serverHost>
</tns:agent>
```

## ConnectDirectCredentials

```
<tns:agent name="CDAGENT01">
  <tns:pnode name="cdnode*" pattern="wildcard">
    <tns:user name="MUSR_.*"
      ignorecase="true"
      pattern="regex"
      cdUserIdCipher="mqmftcred!1!w2PQGhQcyq1NwYzGIItz0VA==!Q40i2rRSEMGwrx6gnRFe8g==" />
    cdPasswordCipher=="!mqmftcred!1!w2PQGhQcyq1NwYzGIItz0VA==!Q40i2rRSEMGwrx6gnRFe8g==" />
    pnodeUserIdCipher="mqmftcred!1!w2PQGhQcyq1NwYzGIItz0VA==!Q40i2rRSEMGwrx6gnRFe8g==" />
    pnodePasswordCipher="mqmftcred!1!w2PQGhQcyq1NwYzGIItz0VA==!Q40i2rRSEMGwrx6gnRFe8g==" />
  <tns:snode name="cdnode2" pattern="wildcard" userId="sue" passwordCipher="!mqmftcred!1!
w2PQGhQcyq1NwYzGIItz0VA==!Q40i2rRSEMGwrx6gnRFe8g==" />
  </tns:user>
</tns:pnode>
</tns:agent>
```

## Syntax

From IBM MQ 9.2.0, the syntax is as follows:



## Parameters

### -f *credentials\_file\_name*

Required. Name of the credentials file whose contents will be encrypted.

**Note:** Deprecated This parameter replaces the `-credentialsFile` parameter that is deprecated from IBM MQ 9.2.0.

### -sp *protection\_mode*

Optional. The protection mode to be used for encrypting credentials. The value can be:

**0**

Use the credentials protection method deprecated at IBM MQ 9.2.0.

**1**

Use the more secure credentials protection method introduced at IBM MQ 9.2.0. For more information, see [Encrypting stored credentials in MFT](#).

This is the default value before IBM MQ 9.3.0.

V 9.3.0 From IBM MQ 9.3.0, you can use protection mode 1 for compatibility with versions earlier than IBM MQ 9.3.0. However, consider migrating any stored passwords to the enhanced protection provided by setting *protection mode* to 2.

V 9.3.0 **2**

Use the enhanced credentials protection mode introduced at IBM MQ 9.3.0. For more information, see [Encrypting stored credentials in MFT](#).

This is the default from IBM MQ 9.3.0.

**-sf *credentials\_key\_file***

Optional. The name of the file containing the credentials key. If this parameter is omitted, the command uses the default credentials key. For more information, see [Protecting passwords in IBM MQ component configuration files](#).

**-o *output\_file\_name***

Optional. Name of the file to output the protected credentials.

**-? or -h**

Optional. Displays command syntax.

**Examples: protection mode 2 (default protection mode)**

To encrypt credentials in the `MQMFTCredentials.xml` file using the default protection mode 2 with the latest algorithm and a fixed key, and store them in the enhanced format, issue the following command:

```
fte0bfuscate -f /usr/home/MQMFTCredentials.xml
```

To encrypt credentials in the `MQMFTCredentials.xml` file using the default protection mode 2 with the latest algorithm and a user specified key, and store them in the enhanced format, issue the following command:

```
fte0bfuscate -sf /var/mqmft/credKeyfile.key -f /usr/home/MQMFTCredentials.xml
```

To encrypt credentials in the `MQMFTCredentials.xml` file using the default protection mode 2 with the latest algorithm and a user specified key, and output the encrypted credentials to another file, issue the **fte0bfuscate** command:

```
fte0bfuscate -sf /var/mqmft/credKeyfile.key -sp 2  
-f /usr/home/MQMFTCredentials.xml -o /usr/home/enccred.xml
```

To encrypt credentials in the `ProtocolBridgeCredentials.xml` file using the default protection mode with the latest algorithm and a user specified key, and store them in the enhanced format, issue the following command:

```
fte0bfuscate -sf /var/mqmft/credKeyfile.key  
-f /usr/home/ProtocolBridgeCredentials.xml
```

**Examples: protection mode 1**

To encrypt credentials in the `MQMFTCredentials.xml` file using protection mode 1 with the latest algorithm and a user specified key, and store them in the more secure format introduced at IBM MQ 9.2.0, issue the following command:

```
fte0bfuscate -sf /var/mqmft/credKeyfile.key -sp 1 -f /usr/home/MQMFTCredentials.xml
```

To encrypt credentials in the `MQMFTCredentials.xml` file using protection mode 1 with the latest algorithm and a user specified key, and output the encrypted credentials to another file, issue the following command:

```
fte0bfuscate -sf /var/mqmft/credKeyfile.key -sp 1  
-f /usr/home/MQMFTCredentials.xml -o /usr/home/enccred.xml
```

**Examples: protection mode 0**

To encrypt credentials in the `MQMFTCredentials.xml` file using the deprecated algorithm, and store them in the deprecated format, issue the following command:

```
fte0bfuscate -f /usr/home/MQMFTCredentials.xml -sp 0
```



## Using data sets on z/OS



Encrypt a data set and output it as an XML file format:

```
fte0bfuscate -f "//test.creds(creds)" -o enc.xml
```

Encrypt a data set with the key specified in the data set:

```
/fte0bfuscate -f "//test.creds(creds)" -sf "//test.creds(key)"
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related reference

[“MFT credentials file format” on page 2697](#)

The `MQMFTCredentials.xml` file contains sensitive user ID and password information. The elements in the `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The security of credentials files is the responsibility of the user.

[MFT and IBM MQ connection authentication](#)

## ftePingAgent (check whether an MFT agent is active)

The **ftePingAgent** command pings a Managed File Transfer agent to determine whether the agent is reachable and, if so, whether it is able to respond to a simple query.

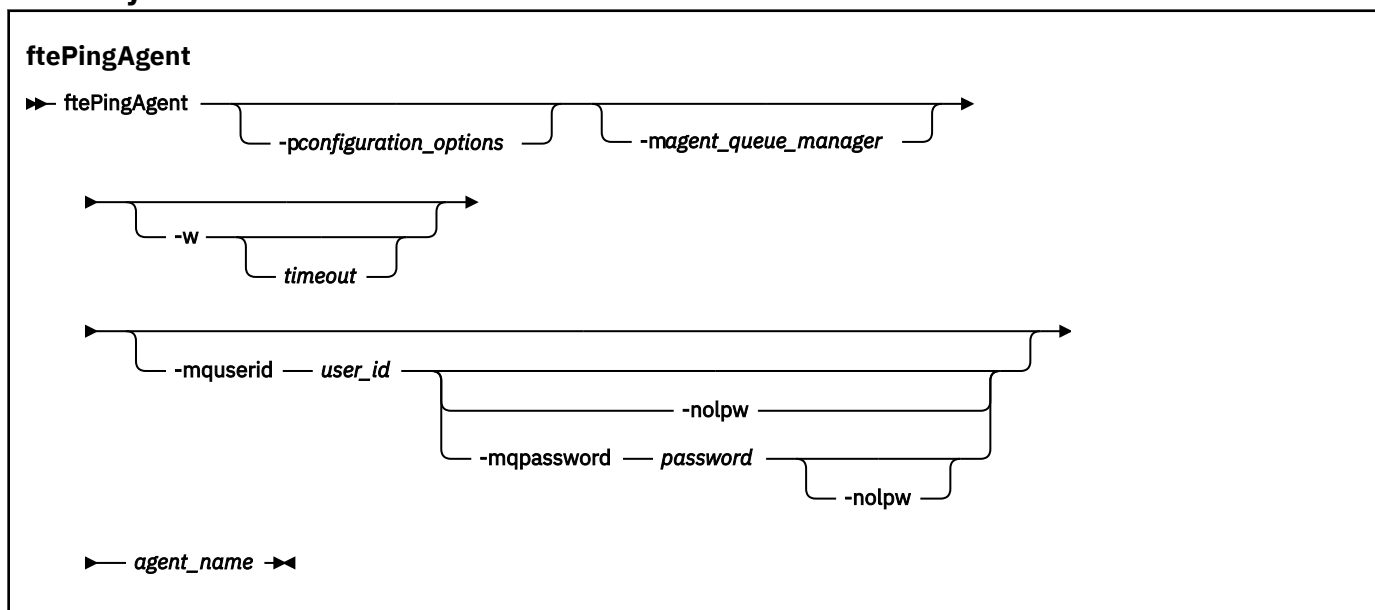
## Purpose

Use the **ftePingAgent** command to check whether a Managed File Transfer agent is reachable and, if so, whether it is able to respond to a simple query along the lines of `are you there?`. An example output of this command is as follows:

```
C:\> ftePingAgent AGENT86
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGPR0127W: No credentials file has been specified to connect to IBM MQ. Therefore, the
assumption is that IBM MQ authentication has been disabled.
BFGCL0212I: Issuing ping request to agent AGENT86
BFGCL0213I: agent AGENT86 responded to ping in 0.094 seconds.
```

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

## Syntax



## Parameters

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to issue the request to ping an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used. See [Configuration options](#) for more information.

### **-m agent\_queue\_manager**

Optional. The name of the queue manager that the agent you want to ping is connected to. If you do not specify the `-m` parameter the queue manager used is determined from the set of configuration options in use.

### **-w timeout**

Optional. Specifies that the command should wait for up to *timeout* seconds for the agent to respond. If you do not specify a timeout, or specify a timeout value of `-1`, then the command waits indefinitely until the agent responds. If you do not specify this option then the default is to wait up to five seconds for the agent to respond.

If *timeout* has been specified, **ftePingAgent** command messages will time out after double the value of *timeout* rather than going to the designated dead letter queue. The command messages will not time out if the command has been set to wait indefinitely.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

► V 9.3.0 ► V 9.3.0 **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**agent\_name**

Required. The name of the Managed File Transfer agent that you want to ping.

**-? or -h**

Optional. Displays command syntax.

**Example**

In this example, the command pings the agent AGENT1, which is connected to QM\_MERCURY. The command waits for up to 40 seconds for AGENT1 to respond before returning.

```
ftePingAgent -m QM_MERCURY -w 40 AGENT1
```

**Return codes**

**0**

Command completed successfully. The agent is active and able to process transfers.

**1**

Command ended unsuccessfully. The command was not able to send a message to the agent.

**2**

Command ended with a timeout. The command sent a message to the agent, but the agent did not respond within the time.

**Related concepts**

[What to do if you think that your file transfer is stuck](#)

**Related reference**

[“fteListAgents \(list the MFT agents for a coordination queue manager\)” on page 2085](#)

Use the **fteListAgents** command to list all of the Managed File Transfer agents that are registered with a particular coordination queue manager.

[“fteShowAgentDetails \(display MFT agent details\)” on page 2129](#)

Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

**fteRAS (collect MFT troubleshooting information)**

The **fteRAS** command collects troubleshooting information (MustGather data) for Managed File Transfer. The data that **fteRAS** collects is specific to the Managed File Transfer installation on the system where the program is being run.

**Purpose**

Use the **fteRAS** command to run the Reliability, Availability, and Serviceability information (RAS) gathering tool if you need to collect troubleshooting information to use to help find a solution when a Managed File Transfer agent, database logger or other command is reporting a problem or failing to work properly.



**Warning:** If there are a large number of Managed File Transfer agents configured on the server, the **fteRAS** command can take a long time to complete. If that happens, you can choose to compress the contents of the Managed File Transfer agent's logs and configuration directories into a zip file.

When you run the **fteRAS** command, the output directory in which the resulting archive (.zip) file is placed can be either the default location, or a directory of your choosing.

## Progress information displayed about the **fteRAS** command while it is running

V 9.3.0

From IBM MQ 9.3.0, the output of the **fteRAS** command to the console has been enhanced, so that you can see the progress of the command while it is running, by displaying the following information:

- The start time and end time of each step in the format yyyy-MM-dd HH:mm:ss, where the time zone is the local time zone
- The total number of files getting copied to a temporary folder from the configuration, log, installations, and bin directories in the step **FTEConfigLogsCollector**
- The size of the data being compressed
- A counter of the number of bytes compressed
- A counter of the total percentage compressed

V 9.3.0

From IBM MQ 9.3.0, the output of the **fteRAS** command also includes subscription information. This information shows the number of messages received for agents, resource monitors, and scheduled transfers. It also shows the number of resource monitor definitions saved to the file system.

The following example shows a sample run from the command:

V 9.3.0

```
C:\Users\Administrator>fteRAS
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGCL0819I: Collector FTEDirectoryListingCollector started at 2022-01-03 15:37:44 India Standard Time.
BFGCL0820I: Collector FTEDirectoryListingCollector completed at 2022-01-03 15:38:38 India Standard Time.

BFGCL0819I: Collector FTEListAgentsCollector started at 2022-01-03 15:38:38 India Standard Time.
BFGMQ1046I: Fetching messages from subscription to topic string 'Agents/#'. Number of messages received: '9'
BFGCL0820I: Collector FTEListAgentsCollector completed at 2022-01-03 15:38:49 India Standard Time.

BFGCL0819I: Collector FTEDisplayVersionCollector started at 2022-01-03 15:38:49 India Standard Time.
BFGCL0820I: Collector FTEDisplayVersionCollector completed at 2022-01-03 15:38:50 India Standard Time.

BFGCL0819I: Collector FTEListMonitorsCollector started at 2022-01-03 15:38:50 India Standard Time.
BFGMQ1046I: Fetching messages from subscription to topic string 'monitors/#'. Number of messages received: '6'
BFGCL0827I: 6 of 6 resource monitor definitions saved to file system.
BFGCL0820I: Collector FTEListMonitorsCollector completed at 2022-01-03 15:39:19 India Standard Time.

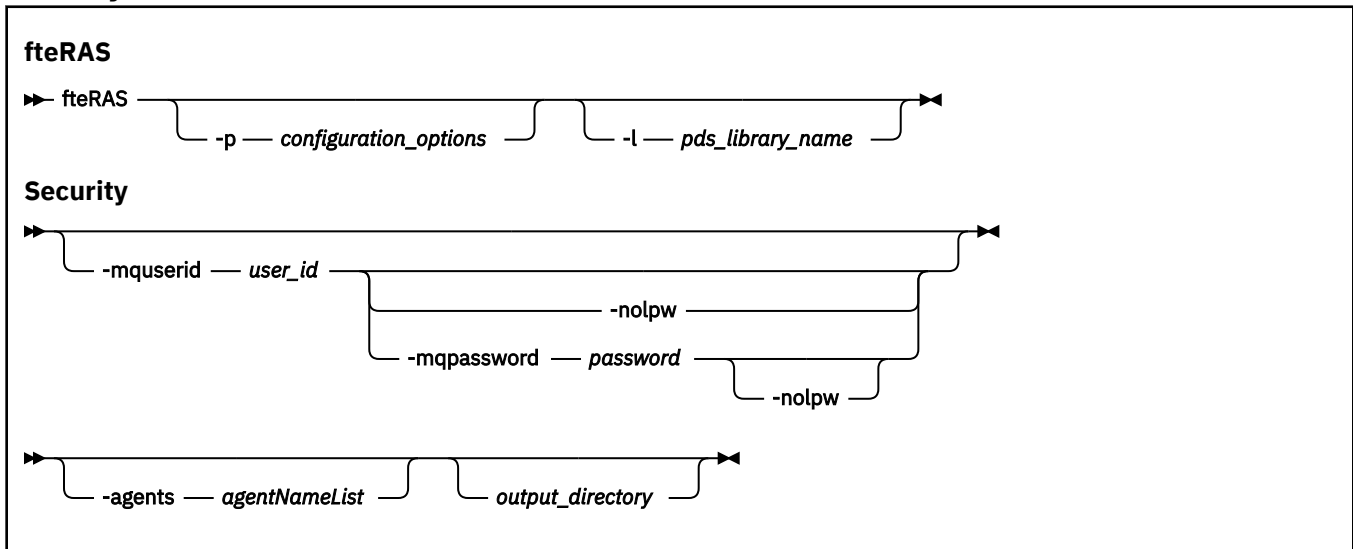
BFGCL0819I: Collector FTEListScheduledTransfersCollector started at 2022-01-03 15:39:19 India Standard Time.
BFGMQ1046I: Fetching messages from subscription to topic string 'Scheduler/#'. Number of messages received: '9'
BFGCL0820I: Collector FTEListScheduledTransfersCollector completed at 2022-01-03 15:39:20 India Standard Time.

BFGCL0819I: Collector FTETopicSubscriptionCollector started at 2022-01-03 15:39:20 India Standard Time.
BFGCL0820I: Collector FTETopicSubscriptionCollector completed at 2022-01-03 15:41:11 India Standard Time.

BFGCL0819I: Collector FTEConfigLogsCollector started at 2022-01-03 15:41:11 India Standard Time.
BFGCL0822I: Copying 8041 file(s) from 'config' directory to a temporary directory.
BFGCL0822I: Copying 8785 file(s) from 'logs' directory to a temporary directory.
BFGCL0821I: Copying file(s) from 'bin' directory to a temporary directory.
BFGCL0822I: Copying 1 file(s) from 'installations' directory to a temporary directory.
BFGCL0820I: Collector FTEConfigLogsCollector completed at 2022-01-03 15:42:21 India Standard Time.

BFGCL0606I: fteRAS command is compressing the output.
BFGCL0823I: Total size of data to compress '1394633' bytes.
BFGCL0824I: Compressed '1394633' bytes. Compression '100%' completed.
BFGCL0604I: fteRAS command completed successfully. Output is stored in
C:\ProgramData\IBM\MQ\mqft\logs\fteRAS.zip.\fteRAS.zip.
```

## Syntax



## Parameters

### **-p configuration\_options**

Optional. Determines the set of configuration options that is used to gather the troubleshooting information. Use the name of a set of configuration options as the value for the **-p** parameter. By convention, this name is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

### **z/OS -l pds\_library\_name**

Optional. z/OS only. Specifies the name of a PDS library that contains JCL scripts that invoke MQMFT commands for a particular agent or logger. This option is always set when the command is run from a command PDS library's BFGZRAS JCL script, such that all members of the PDS library are captured in the output directory.

**Note:** BFGZRAS creates the BFGRAS member when the BFGCUSTM job is run.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

## V 9.3.4 -agents

Optional. Specifies the names of the agents for which data will be collected.

Agent names must be enclosed in quotes and multiple agent names must be provided as a comma separated list of agent names. For example:

```
fteRAS -agents "AGENT_LIN,AGENT_LIN3"
```

Agent names can include asterisk wildcard characters (\*) where \* matches zero or more characters. Asterisks are the only type of wildcard character that **fteRAS** supports. For example:

```
fteRAS -agents "AGENT_*"
```

```
fteRAS -agents "A*,G*"
```

If, during validation, **fteRAS** cannot find an agent on the system with a name that matches the supplied agent name or agent name pattern with a wildcard character, it fails with message [BFGCL0835E](#).

### output\_directory

Optional. A directory to use when you are gathering the RAS data, and where the output file, for example `fteRAS.zip`, is stored after the data is gathered successfully. If the directory does not exist, it is created. The default location is the `mqft` logs directory.

### -? or -h

Optional. Displays command syntax.

### Examples

#### Linux

#### AIX

On AIX and Linux, to store the output file `fteRAS.zip` in the `/var/mqm/errors` directory, run **fteRAS** as shown in the following example:

```
fteRAS /var/mqm/errors
```

The following message confirms that the command has completed successfully:

```
BFGCL0604I: fteRAS command completed successfully. Output is stored in /var/mqm/errors/fteRAS.zip
```

#### Windows

On Windows, to store the output file `fteRAS.zip` in the default errors directory for a new installation of IBM MQ, run **fteRAS** as shown in the following example:

```
fteRAS "C:\ProgramData\IBM\MQ\errors"
```

The following message confirms that the command has completed successfully:

```
BFGCL0604I: fteRAS command completed successfully. Output is stored in  
C:\ProgramData\IBM\MQ\errors\fteRAS.zip
```

**Note:** For IBM MQ 8.0 or later, if this is not a new installation of that version of the product, the location of the errors directory might be different on your system. For more information, see [Program and data directory locations on Windows](#).

#### IBM i

On IBM i, to copy the output file to `/QIBM/UserData/mqm/errors`, run the **fteRAS** command from the Qshell as shown in the following example:

```
/QIBM/ProdData/mqm/bin/fteRAS /QIBM/UserData/mqm/errors
```

The following message confirms that the command has completed successfully:

```
BFGCL0604I: fteRAS command completed successfully. Output is stored in /QIBM/UserData/mqm/errors/  
fteRAS.zip
```

### Related tasks

[Troubleshooting MFT](#)

[Collecting information for Managed File Transfer problems on Multiplatforms](#)

## fteSetAgentLogLevel (Turn on or turn off logging to file of certain MFT agent operations)

**V 9.3.0** Use the **fteSetAgentLogLevel** command to turn on or turn off logging for the interactions between a protocol bridge agent and file servers, resource monitor activity, and transfer logs.

### Purpose

IBM MQ Managed File Transfer provides a logging mechanism that can be used to capture:

- Information about the flows between a protocol bridge agent and file servers
- Details about the polls performed by resource monitors
- **V 9.3.0** Progress of transfers

When you use the **fteSetAgentLogLevel** command to enable logging for a protocol bridge agent, the agent records details of the FTP, SFTP and FTPS commands that are sent to the file server, and the responses that are received. This information is written to a log file called `agenteventN.log`, where *N* stands for a number.

- **Multi** On Multiplatforms, the `agenteventN.log` file is in the `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name` directory.
- **z/OS** On z/OS, the `agenteventN.log` file is in the `BFG_DATA/mqft/logs/coordination_qmgr_name/agents/agent_name` directory.

The information in the log file can be useful in diagnosing issues that might occur during a file transfer involving the protocol bridge agent.

When you use the **fteSetAgentLogLevel** command to enable logging for resource monitors, the agent records information about the polls performed by the monitors into a log file called `resmoneventN.log`, where *N* stands for a number.

- **Multi** On Multiplatforms, the `resmoneventN.log` file is in the `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name` directory.
- **z/OS** On z/OS, the `resmoneventN.log` file is in the `BFG_DATA/mqft/logs/coordination_qmgr_name/agents/agent_name` directory.

The information in the log file includes:

- The time when the monitor started and finished a poll.
- Details of any managed transfers submitted as a result of a poll.

For more information on resource monitor logging, see [Logging MFT resource monitors](#).

You can turn on, turn off, and set the level of logging that you require, in two ways:

- Use the **fteSetAgentLogLevel** command to enable or disable logging while the agent is running. You do not need to restart the agent for the change to the logging level to take effect.
- Set properties in the `agent.properties` file to enable or disable logging from startup. The properties that need to be set depend on whether logging is being enabled for a protocol bridge agent, or for resource monitors:
  - For protocol bridge agents, logging is controlled using the **agentLog** property.
  - To enable or disable resource monitor logging, use the **resourceMonitorLog** property.

**V 9.3.0** If you use the **fteSetAgentLogLevel** command to enable transfer logging, the agent records information about transfer progress into a log file called `transferlog0.json`.

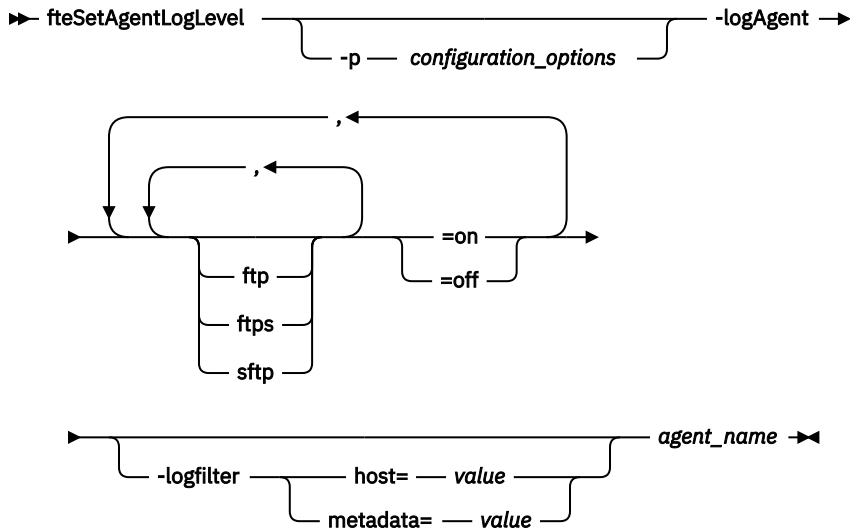
- **Multi** On Multiplatforms, the transferlog0.json file is in the MQ\_DATA\_PATH/mqft/logs/coordination\_qmgr\_name/agents/agent\_name/logs directory.
- **z/OS** On z/OS, the transferlog0.json file is in the BFG\_DATA/mqft/logs/coordination\_qmgr\_name/agents/agent\_name/logs directory.

For more information, see [The agent.properties file](#).

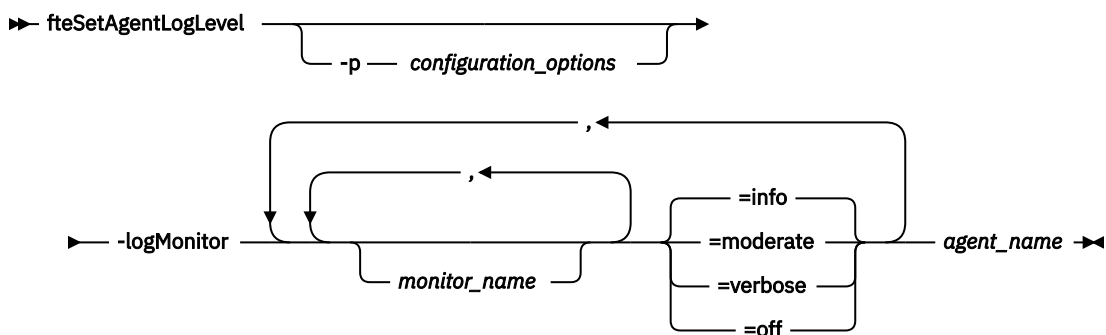


**Attention:** When the command is run on the system where a standby instance of a highly available (HA) agent is running, the new log level is applied only to that instance. The log level on the active instance of the agent, and any other standby instances, will not be changed.

### Syntax - LogAgent option

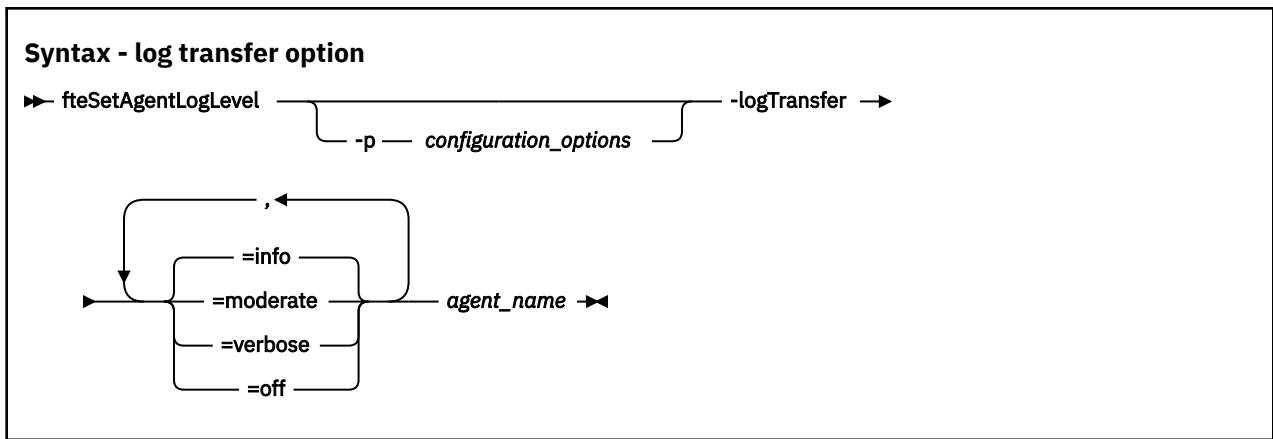


### Syntax - log monitor option



V 9.3.0





**Note:**

**V 9.3.0** You must select only one of the **logAgent**, **logMonitor** or **logTransfer** options. If you specify more than one option, the command fails with the following error message:

BFGCL0756E:Invalid command options. Specify only one of logAgent, logMonitor, or logTransfer options.

**Parameters**

**-logMonitor monitor\_name=log\_level**

Required.

A comma separated list of resource monitors and logging levels, where:

**monitor\_name**

Optional. The name of the resource monitor, or a comma-separated list of resource monitors, that the logging level is to be applied to. If you do not specify a monitor name, or a comma-separated list of resource monitors, the logging level is applied to all resource monitors running within the agent.



**Attention:** If you have specified non-existent resource monitor names in the command, no error is displayed on the console.

**log\_level**

Required.

The logging level to use. This can be one of the following values:

**info**

Turn on information level logging. This is the default value and enables high level logging for the resource monitors of the agent..

To enable **info** level logging for monitor MON1 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1=info AGENT1
```

**moderate**

Turn on moderate level logging.

To enable **moderate** level logging for monitors MON1 and MON2 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1,MON2=moderate AGENT1
```

**verbose**

Turn on verbose level logging.

Enable **verbose** level logging for all monitors of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor =verbose AGENT1
```

**off**

Turn off logging.

To turn off logging for monitors MON1 and MON2 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1,MON2=off AGENT1
```

To turn off logging for monitor MON1 and enable info level logging for monitor MON2 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1=off,MON2=info AGENT1
```

To turn off logging for all monitors of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor=off AGENT1
```

If the same resource monitor name is repeated in a command, then the last occurrence of a component is considered as valid. For example:

```
fteSetAgentLogLevel -logMonitor MON1=info,MON2=off,MON1=off AGENT1 turns off logging for resource monitor MON1.
```

For more information about the different logging levels, and the resource monitor events that are logged at each level, see [Logging MFT resource monitors](#).

### 9.3.0

You cannot use the **logMonitor** option with the **logAgent** and **logTransfer** options.

#### **-logAgent component=operation**

Required.

Protocol bridge agent logging can be enabled or disabled for the FTP, FTPS, and SFTP protocols. Specify one of the three possible server protocols and add an operation value to turn the logging off or on for the protocol bridge agent.

#### **component**

Optional.

The valid components are:

#### **ftp**

The logging operation is applied to all communication between a protocol bridge agent and file servers that use the FTP protocol.

#### **ftps**

The logging operation is applied to the communication between a protocol bridge agent and file servers that use the FTPS protocol.

#### **sftp**

The logging operation is applied to the communication between a protocol bridge agent and file servers that use the SFTP protocol.

If a component starts with a plus sign (+), the list of components following the plus sign are added to any existing log component currently being logged.

#### **operation**

The valid log level operation options are as follows:

#### **off**

Disable all logging for a protocol bridge agent. This option is the default.

```
fteSetAgentLogLevel -logAgent =off PBA1
```

To disable logging for a specified component that the protocol bridge agent is connecting to, use these commands:

```
fteSetAgentLogLevel -logAgent ftp=off PBA1
```

```
fteSetAgentLogLevel -logAgent ftps=off PBA1
```

```
fteSetAgentLogLevel -logAgent sftp=off PBA1
```

#### on

To enable logging for all three possible file server components that the protocol bridge agent is connecting to, use this command:

```
fteSetAgentLogLevel -logAgent =on PBA1
```

To enable logging for a specified component that a protocol bridge agent is connecting to, use these commands:

```
fteSetAgentLogLevel -logAgent ftp=on PBA1
```

```
fteSetAgentLogLevel -logAgent ftps=on PBA1
```

```
fteSetAgentLogLevel -logAgent sftp=on PBA1
```

If any component starts with a plus sign (+), the list of components following the plus sign is added to any existing log component currently being logged.

For further configuration options, see “[Example 1](#)” on page 2116 and “[Example 2](#)” on page 2117.

**V 9.3.0** You cannot use the **logAgent** option with the **logTransfer** and **logMonitor** options.

#### **-logFilter filter=value**

Optional.

Use the **logFilter** parameter to limit protocol bridge agent logging based on the specified filter criteria. You must specify a value for either one or more file server hosts, or a property within the user metadata for a managed transfer.

#### **host**

Use **host** to filter by:

- The host name of the system where the file server is located.
- A list of comma separated host names or IP addresses.

To log the FTP commands sent to, and the responses received from, the file server `ftpprod.ibm.com`, use this command:

```
fteSetAgentLogLevel -logAgent ftp=on -logFilter host=ftpprod.ibm.com PBA1
```

To log the SFTP commands sent to, and the responses received from, all file servers that have IP addresses starting with `9.182.*`, use this command:

```
fteSetAgentLogLevel -logAgent sftp=on -logFilter host=9.182.* PBA1
```

#### **metadata**

Specify any text, as defined by the user during the transfer creation, in a *key=value* format. For example **metadata="BANK=WORLD BANK"**.

To enable logging for all file servers that connect to the protocol bridge agent PBA1 using the FTP protocol, and filter the output to only include entries for managed transfers that contain the metadata `"BANK=WORLD BANK"`, use this command:

```
fteSetAgentLogLevel -logAgent ftp=on metadata="BANK=WORLD BANK" PBA1
```

**Note:** In order to filter by metadata, the value you are filtering by, must be specified under the **-md** parameter as a part of a file transfer. For more information, see [“fteCreateTransfer \(start a new file transfer\)”](#) on page 2048.

### V 9.3.0 **-logTransfer log\_level**

Required.



**Attention:** If the **fteSetAgentLogLevel** command is run by a user other the one that started the agent, error message BFGNV0066E is output on the console:

Turn on or turn off transfer logs. Possible log levels are:

#### **info**

Turn on high level logging information of a transfer.

To enable **info** level transfer logging of agent AGENTQM:

```
fteSetAgentLogLevel -p AGENTQM -logTransfer info SRC
```

This is the default value which means high level transfer logs are written for every transfer, and will use file system space, up to a maximum of 100MB.

#### **moderate**

Turn on intermediate level log information of a transfer

#### **verbose**

Turn on detailed log information of a transfer.

Enable detailed level logging of agent AGENTQM:

```
fteSetAgentLogLevel -p AGENTQM -logTransfer verbose SRC
```

#### **off**

Turn off transfer logging.

To turn off transfer logging of agent AGENTQM:

```
fteSetAgentLogLevel -p AGENTQM -logTransfer off SRC
```

You cannot use the **logTransfer** option with the **logAgent** and **logMonitor** options.

See [Output produced by the LogTransfer function](#) for examples of the logging information produced.

#### **-p configuration\_options**

Optional.

Determines the set of configuration options that is used to set the agent log level. Use the name of a set of configuration options as the value for the **-p** parameter.

By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

#### **-? or -h**

Optional. Displays the command syntax.

#### **agent\_name**

Required. Name of the protocol bridge agent for which the logging is enabled or disabled.



**Attention:** If you have specified non-existent resource monitor names in the command, no error is displayed on the console.

#### **Example 1**

In this example, multiple components are specified in one command, by using a command delimited group. Logging is enabled for the FTP and SFTP protocol, and disabled for the FTPS protocol, on the protocol bridge agent PBA1.

```
fteSetAgentLogLevel -logAgent ftp=on,ftps=off,sftp=on PBA1
```

You can also separate the components with a comma to achieve the same result, for example;

```
fteSetAgentLogLevel -logAgent ftp,sftp=on,ftps=off PBA1
```

### Example 2

In this example, the same component is repeated in a command. The last instance of a *component=operation* pair is considered as valid. This example disables logging for the FTP protocol on the protocol bridge agent PBA1.

```
fteSetAgentLogLevel -logAgent ftp=on,ftp=off PBA1
```

The previous example has the same effect as this example:

```
fteSetAgentLogLevel -logAgent ftp=off PBA1
```

### Example 3

This example enables the default value of *info* level logging for monitor MON1 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1=info AGENT1
```

### Example 4

This example enables , *moderate* level logging for monitors MON1 and MON2 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1,MON2=moderate AGENT1
```

### Example 5

This example turns off logging for monitor MON1 and enable *info* level logging for monitor MON2 of agent AGENT1:

```
fteSetAgentLogLevel -logMonitor MON1=off,MON2=info AGENT1
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

### Related reference

[The protocol bridge](#)

[The MFT agent.properties file](#)

[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

## fteSetAgentTraceLevel (modify current trace level for an agent)

Use the **fteSetAgentTraceLevel** command to modify the current trace level for an agent dynamically.

### Purpose

Use this command to switch agent trace on and off or to change the level of agent trace that is set. When you use the **fteSetAgentTraceLevel** command, you do not have to shut down and restart an agent to modify the trace level. The trace files produced are located in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/trace%PID%/trace%PID%.txt`, where `%PID%` is the process ID for the agent instance.



#### Attention:

**Multi** When using IBM MQ on Multiplatforms, only the user that the agent process is running under can run the **fteSetAgentTraceLevel** command.

**z/OS** The **fteSetAgentTraceLevel** command can be run by either:

- The same userid that the agent process is running as.
- Members of the group specified by the agent property **adminGroup**.

For more information, see the **adminGroup** property in [The MFT agent.properties file](#).

The **fteSetAgentTraceLevel** command also writes a trace for the agent process controller. The trace files produced are located in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name/logs/pctrace%PID%/pctrace%PID%.txt`, where `%PID%` is the process ID for the agent instance.

You can also use the command to cause the agent process to generate a Javacore. The agent generates a Javacore file in the following directory: `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/agents/agent_name`.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.



#### Attention:

1. You must run this command on the system where the agent is running.
2. The traces and logging do not persist across an agent restart.

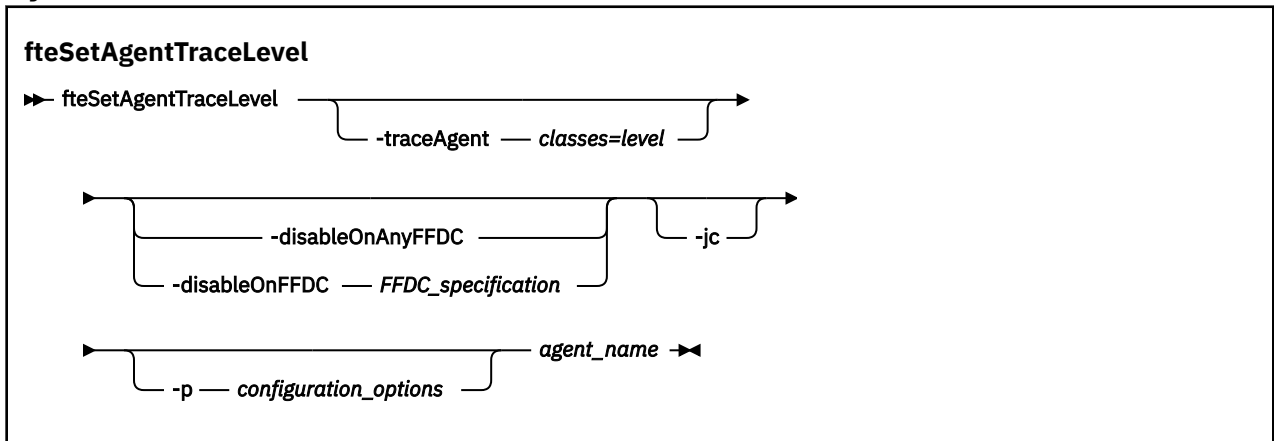
If the agent terminates and is restarted by the Process Controller process, the dynamic traces and logs are not in effect until the `agent.properties` file has been updated to include the required trace and log properties.

3. When the command is run on the system where a standby instance of a highly available (HA) agent is running, the trace level is only applied to that instance. The trace level on the active instance of the agent, and any other standby instances, will not be changed.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the `agent.properties` file. These properties are described in [Advanced agent properties](#).

Specify the optional `-p` parameter for this command only if you want to use a set of configuration options different from your default set. See [The MFT agent.properties file](#) for more information.

## Syntax



## Parameters

### **-traceAgent classes=level**

Required. Level to set the agent trace and which classes to apply the trace to.

You can specify a colon-separated list of class specifications. This option enables you to set tracing of different classes, and at different levels. For example:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.agent=all:com.ibm.wmqfte.cmdhandler=moderate AGENT1
```

You can still specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes. Use the following format:

```
classes=level
```

For example:

```
com.ibm.wmqfte=all
```

You can substitute *classes* with a Managed File Transfer package name to trace a specific package only. However, because this option captures just a subset of the agent's behavior, you are generally not recommended to use package filtering.

If *classes* start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

### **off**

Switches the agent trace off but continues to write information to the log files. This is the default option.

### **flow**

Captures data for trace points associated with processing flow in the agent.

### **moderate**

Captures a moderate amount of diagnostic information in the trace.

### **verbose**

Captures a verbose amount of diagnostic information in the trace.

### **all**

Sets agent trace to run on all agent classes.

To start full tracing for the agent, run the following command:

```
fteSetAgentTraceLevel -traceAgent =all agent_name
```

To stop full tracing for the agent, run the following command:

```
fteSetAgentTraceLevel -traceAgent =off agent_name
```

#### **-disableOnAnyFFDC**

Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

#### **-disableOnFFDC *ffdc\_specification***

Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file that matches the *ffdc\_specification*. *ffdc\_specification* is a comma-separated list of values. The format of the values can be either:

##### ***class\_name***

The name of the class where the FFDC originated. For example, `com.ibm.wmqfte.classA`.

##### ***class\_name:probe\_id***

The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, `com.ibm.wmqfte.classB:1`.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

#### **-jc**

Optional. Requests that the agent generates a Javacore file. The IBM service team may request that you run the command with this parameter to assist with problem diagnosis. This parameter cannot be used with any other parameter except **-p**.

#### **-p *configuration\_options***

Optional. This parameter determines the set of configuration options that is used to set the agent trace level. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

#### ***agent\_name***

Required. The name of the Managed File Transfer Agent that you want to set the trace level for.

#### **-? or -h**

Optional. Displays command syntax.

### **Example**

In this example, the trace level is set to `all` for all classes for AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte=all AGENT1
```

In this example, the trace level is set to `all` for the classes `com.ibm.wmqfte.agent.Agent` and `com.ibm.wmqfte.cmdhandler` for AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.cmdhandler=moderate AGENT1
```



In this example, subclasses are excluded from the trace because the **-traceLevel** parameter is set to off. All classes starting with com.ibm.outer are traced at verbose level except classes starting with com.ibm.outer.inner:

```
fteSetAgentTraceLevel -traceAgent com.ibm.outer=verbose AGENT1
fteSetAgentTraceLevel -traceAgent +com.ibm.outer.inner=off AGENT1
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## fteSetLoggerTraceLevel (modify current trace level for a logger)

Use the **fteSetLoggerTraceLevel** command to modify the current trace level for a Managed File Transfer logger dynamically.

### Purpose

Use this command to switch logger trace on and off or change the level of logger trace that is set. When you use the **fteSetLoggerTraceLevel** command, you do not have to shut down and restart a logger to modify the trace level. The trace files that are produced are located in *MQ\_DATA\_PATH/mqft/logs/coordination\_qmgr\_name/loggers/logger\_name/logs/trace%PID%/trace%PID%.txt*, where *%PID%* is the process ID for the logger instance.

The **fteSetLoggerTraceLevel** command also writes a trace for the logger process controller. The trace files that are produced are located in *MQ\_DATA\_PATH/mqft/logs/coordination\_qmgr\_name/loggers/logger\_name/logs/pctrace%PID%/pctrace%PID%.txt*, where *%PID%* is the process ID for the logger instance.

The command can also be used to cause the logger process to generate a Javacore. The logger generates a Javacore file in the following directory: *MQ\_DATA\_PATH/mqft/logs/coordination\_qmgr\_name/loggers/logger\_name*.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the `logger.properties` file. These properties are described in [Logger properties](#).

Specify the optional `-p` parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see [Logger properties](#).

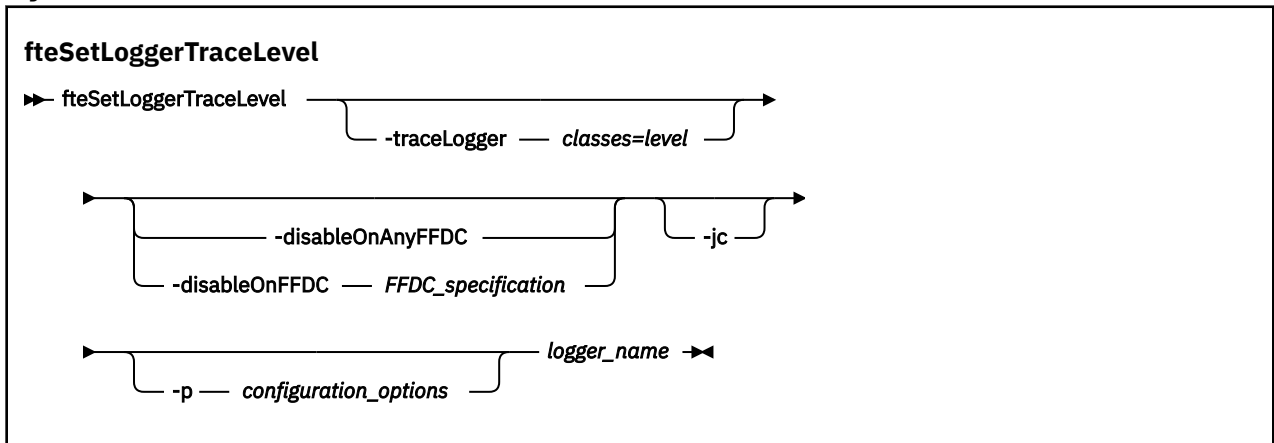


#### Attention:

1. You must run this command on the system where the logger is running.
2. The traces and logging do not persist across a logger restart.

If the logger terminates and is restarted by the Process Controller process, the dynamic traces and logs are not in effect until the `logger.properties` file has been updated to include the required trace and log properties.

## Syntax



## Parameters

### **-traceLogger *classes=level***

Required. Level to set the logger trace and which classes to apply the trace to.

You can specify a colon-separated list of class specifications. This option enables you to set tracing of different classes, and at different levels. For example:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.wmqfte.logger=all:com.ibm.wmqfte.cmdhandler=moderate  
LOGGER1
```

You can still specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes. Use the following format:

```
classes=level
```

For example:

```
com.ibm.wmqfte=all
```

Specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all logger classes.

If (*classes*) start with a plus sign (+), the list of trace classes that follow the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

#### **off**

Switches off the logger trace but continues to write information to the log files. This is the default option.

#### **flow**

Captures data for trace points associated with processing flow in the logger.

#### **moderate**

Captures a moderate amount of diagnostic information in the trace.

#### **verbose**

Captures a verbose amount of diagnostic information in the trace.

#### **all**

Sets logger trace to run on all logger classes.

### **-disableOnAnyFFDC**

Optional. If this parameter is specified, trace is disabled on the logger when it generates a First Failure Data Capture (FFDC) file.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

### **-disableOnFFDC *ffdc\_specification***

Optional. If this parameter is specified, trace is disabled on the logger when it generates a First Failure Data Capture (FFDC) file that matches the *ffdc\_specification*. *ffdc\_specification* is a comma-separated list of values. The value can be one of the following formats:

#### ***class\_name***

The name of the class where the FFDC originated. For example, `com.ibm.wmqfte.classA`.

#### ***class\_name:probe\_id***

The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, `com.ibm.wmqfte.classB:1`.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

### **-jc**

Optional. Requests that the logger generates a Javacore file. The IBM service team might request that you run the command with this parameter to assist with problem diagnosis. You cannot use the **-jc** parameter with any other parameter.

### **-p *configuration\_options***

Optional. This parameter determines the set of configuration options that is used to set the logger trace level. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### ***logger\_name***

Required. The name of the Managed File Transfer Logger that you want to set the trace level for.

### **-? or -h**

Optional. Displays command syntax.

## **Example**

In this example, the trace level is set to `all` for all classes for `LOGGER1`:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.wmqfte=all LOGGER1
```

In this example, the trace level is set to `all` for the classes `com.ibm.wmqfte.logger.logger` and `com.ibm.wmqfte.cmdhandler` for `LOGGER1`:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.wmqfte.logger.logger,com.ibm.wmqfte.cmdhandler=moderate  
LOGGER1
```

In this example, subclasses are excluded from the trace because the **-traceLevel** parameter is set to `off`. All classes that start with `com.ibm.outer` are traced at verbose level except classes that start with `com.ibm.outer.inner`:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.outer=verbose LOGGER1  
fteSetLoggerTraceLevel -traceLogger +com.ibm.outer.inner=off LOGGER1
```

## **Return codes**

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## **fteSetProductId (set z/OS SCRT recording product id)**

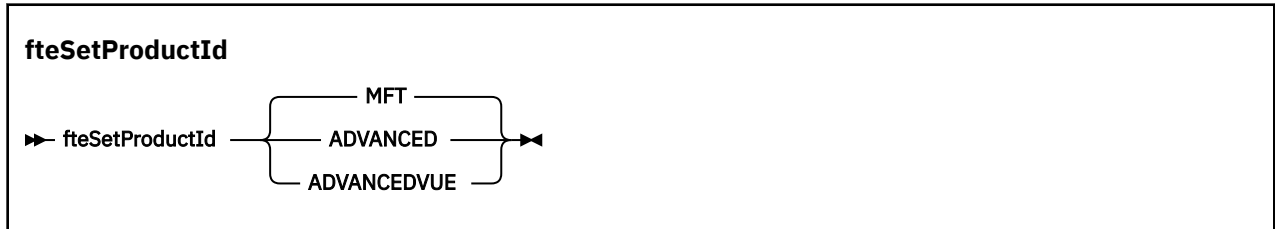
The **fteSetProductId** is used to set the product type against which Managed File Transfer usage is to be recorded for the installation. This command is valid only on z/OS.

### Purpose

This command can be run at any time, after at least one coordination queue manager has been defined, or the [MFT installation.properties](#) file for the installation has been created.

See [Reporting product information](#) for more information on product usage recording.

### Syntax



### Parameters

The product type for usage recording:

Specify one of:

#### **MFT**

Usage is recorded as a stand-alone Managed File Transfer product, with product ID 5655-MF9.

#### **ADVANCED**

Usage is recorded as part of an IBM MQ Advanced for z/OS product, with product ID 5655-AV9.

#### **ADVANCEDVUE**

Usage is recorded as part of an IBM MQ Advanced for z/OS Value Unit Edition product, with product ID 5655-AV1.

### Return codes

**0**

Command completed successfully.

**1**


Command ended unsuccessfully, or if the product type has not been set to the requested value.


### Related tasks

[Configuring the coordination queue manager for MFT](#)

## **fteSetupCommands (create the MFT command.properties file)**

The **fteSetupCommands** command creates the Managed File Transfer `command.properties` file. This properties file specifies the details of the queue manager that connects to the IBM MQ network when you issue commands.

**Important:**  On IBM MQ for AIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the `mqm` group) can run this command. If you try to run this command as a user who is not an IBM MQ administrator, you will receive the error message `BFGCL0502E: You are not authorized to perform the requested operation.` and the command will not run.

 On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the command:

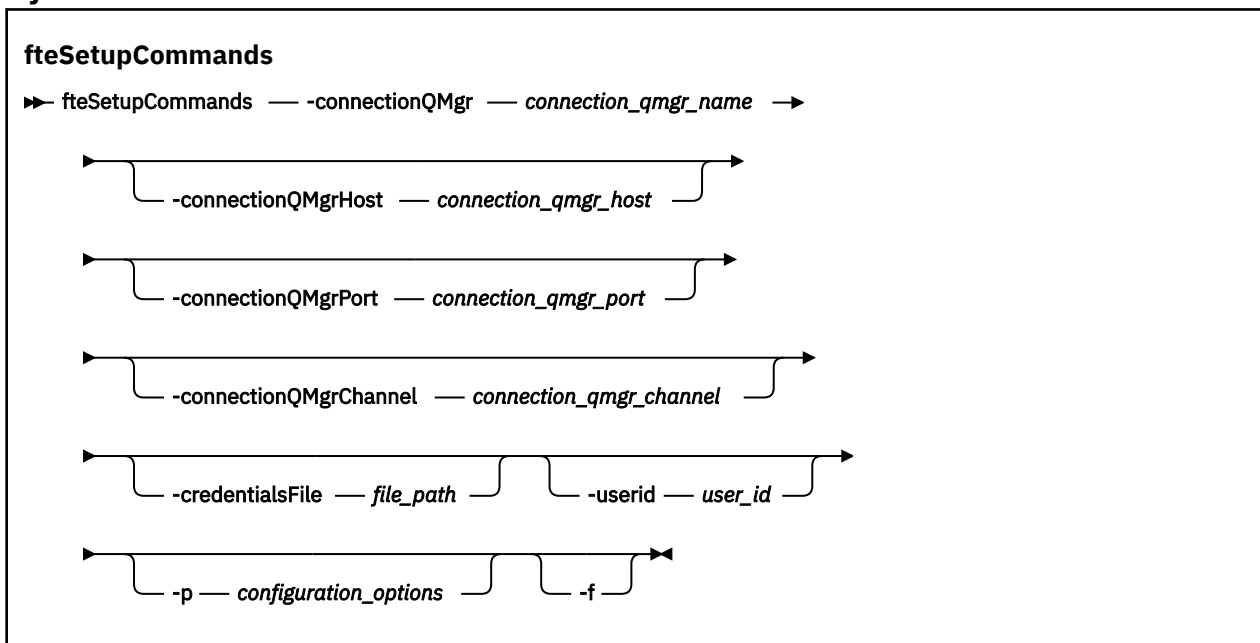
- Be a member of the mqm group (if the mqm group is defined on the system).
- Be a member of the group named in the BFG\_GROUP\_NAME environment variable (if one is named).
- Have no value set in the BFG\_GROUP\_NAME environment variable when the command is run.

## Purpose

Use the **fteSetupCommands** command to create a `command.properties` file in the coordination queue manager configuration directory. The command uses the `MQ_DATA_PATH` environment variable and the `installation.properties` file to determine where to locate the `command.properties` file. Ensure that you have already created and configured a coordination queue manager before you issue the **fteSetupCommands** command.

For more information about properties files, see [The MFT command.properties file](#).

## Syntax



## Parameters

### **-connectionQMgr connection\_qmgr\_name**

Required. The name of the queue manager used to connect to the IBM MQ network to issue commands.

### **-connectionQMgrHost connection\_qmgr\_host**

Optional. The host name or IP address of the connection queue manager.

If you do not specify the **-connectionQMgrHost** parameter, a bindings mode connection is assumed. Therefore, this parameter is required if you are using a client mode connection.

If you specify a value for the **-connectionQMgrHost** parameter but do not specify values for the **-connectionQMgrPort** and **-connectionQMgrChannel** properties, a port number of 1414 and a channel of `SYSTEM.DEF.SVRCONN` are used by default.

### **-connectionQMgrPort connection\_qmgr\_port**

Optional. The port number used to connect to the connection queue manager in client mode. If you specify the **-connectionQMgrPort** parameter, you must also specify the **-connectionQMgrHost** parameter.

**-connectionQMgrChannel *connection\_qmgr\_channel***

Optional. The channel name used to connect to the connection queue manager. If you specify the **-connectionQMgrChannel** parameter, you must also specify the **-connectionQMgrHost** parameter.

**-p *configuration\_options***

Optional. This parameter determines the set of configuration options that is used to set up a command queue manager. Use the name of a non-default coordination queue manager as the input for this parameter. The **fteSetupCommands** command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

**-credentialsFile *file\_path***

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

**-userid *user\_id***

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

**-f**

Optional. Forces an overwrite of the existing `command.properties` file with the details specified in this command.

**-? or -h**

Optional. Displays command syntax.

## Example

```
fteSetupCommands -connectionQMgr QM_NEPTUNE -connectionQMgrHost 9.146.157.241  
-connectionQMgrPort 1414 -connectionQMgrChannel SYSTEM.DEF.SVRCONN
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related reference


The MFT `command.properties` file

[“fteSetupCoordination \(set up properties files and directories for coordination queue manager\)” on page 2126](#)

The **fteSetupCoordination** command creates properties files and the coordination queue manager directory for Managed File Transfer.

## fteSetupCoordination (set up properties files and directories for coordination queue manager)

The **fteSetupCoordination** command creates properties files and the coordination queue manager directory for Managed File Transfer.

**Important:**  On IBM MQ for AIX, Linux, and Windows, only users who are IBM MQ administrators (and members of the `mqm` group) can run this command. If you try to run this command as

a user who is not an IBM MQ administrator, you will receive the error message BFGCL0502E: You are not authorized to perform the requested operation. and the command will not run.

**z/OS** On z/OS systems, the user must satisfy (at least) one of these conditions in order to run the command:

- Be a member of the mqm group (if the mqm group is defined on the system).
- Be a member of the group named in the BFG\_GROUP\_NAME environment variable (if one is named).
- Have no value set in the BFG\_GROUP\_NAME environment variable when the command is run.

## Purpose

Use the **fteSetupCoordination** command to create the following Managed File Transfer objects:

- Coordination queue manager directory
- Data directory mqft (if this does not exist)
- installation.properties file
- coordination.properties file

This command also provides you with the following MQSC commands that you must run against your coordination queue manager to configure Managed File Transfer. The MQSC commands create a topic, a topic string, the SYSTEM.FTE queue, and the default database logger queues. These commands also update a namelist and set the PSMODE attribute of the coordination queue manager to ENABLED.

**z/OS** If the coordination queue manager is on z/OS, before you run these MQSC commands, you must ensure that the following required objects already exist:

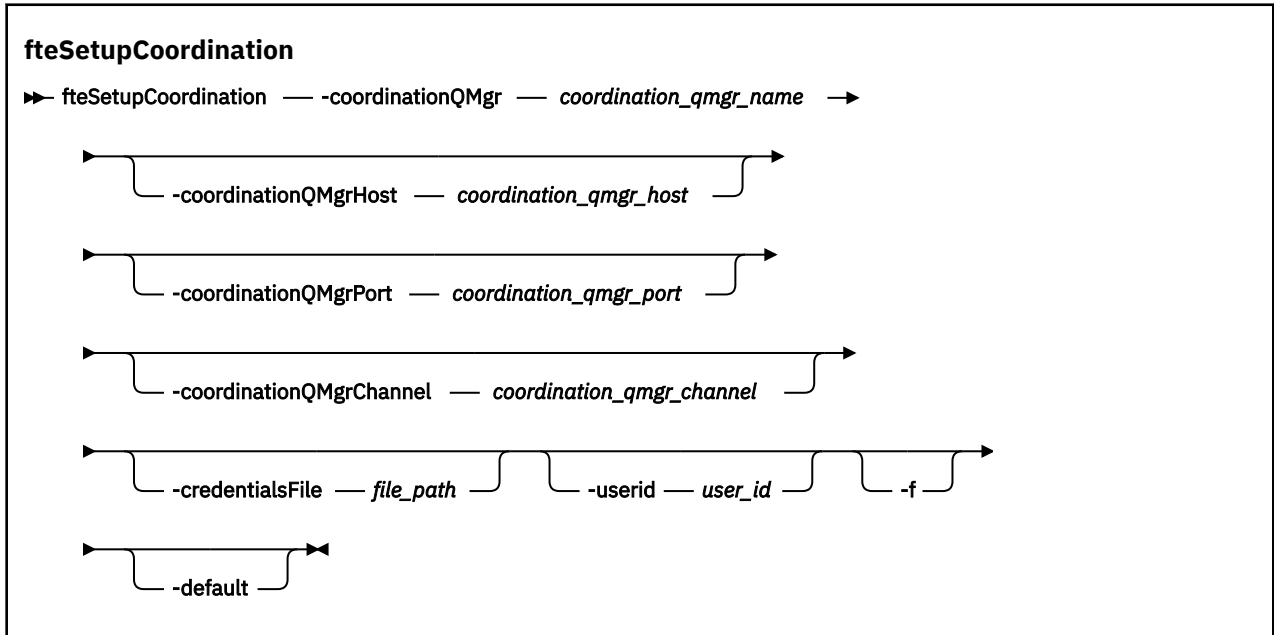
- SYSTEM.BROKER.DEFAULT.STREAM queue
- SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist
- SYSTEM.BROKER.DEFAULT.STREAM and SYSTEM.BROKER.ADMIN.STREAM streams

```
DEFINE TOPIC('SYSTEM.FTE') TOPICSTR('SYSTEM.FTE') REPLACE
ALTER TOPIC('SYSTEM.FTE') NPMGDLV(ALLAVAIL) PMGDLV(ALLAVAIL)
DEFINE QLOCAL(SYSTEM.FTE) LIKE(SYSTEM.BROKER.DEFAULT.STREAM) REPLACE
ALTER QLOCAL(SYSTEM.FTE) DESCR('Stream for WMQFTE Pub/Sub interface')
* Altering namelist: SYSTEM.QPUBSUB.QUEUE.NAMELIST
* Value prior to alteration:
DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
ALTER NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST) +
  NAMES(SYSTEM.BROKER.DEFAULT.STREAM+
  ,SYSTEM.BROKER.ADMIN.STREAM,SYSTEM.FTE)
* Altering PSMODE. Value prior to alteration:
DISPLAY QMGR PSMODE
ALTER QMGR PSMODE(ENABLED)
```

For more information about properties files, see [Configuration options](#).

**z/OS** If you are using z/OS, you can issue the **fteSetupCoordination** command and other commands from JCL with scripts generated from the Managed File Transfer command template PDSE library data set. For more information, see [Creating an MFT Agent or Logger command data set](#).

## Syntax



## Parameters

### **-coordinationQMGr** *coordination\_qmgr\_name*

Required. The name of the coordination queue manager. This queue manager must be an IBM WebSphere MQ 7.0 or later queue manager.

### **-coordinationQMGrHost** *coordination\_qmgr\_host*

Optional. The host name or IP address of the coordination queue manager.

If you do not specify the **-coordinationQMGrHost** parameter, a bindings mode connection is assumed.

If you specify a value for the **-coordinationQMGrHost** parameter but do not specify values for the **-coordinationQMGrPort** and **-coordinationQMGrChannel** parameters, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

### **-coordinationQMGrPort** *coordination\_qmgr\_port*

Optional. The port number used for client connections to the coordination queue manager.

If you specify the **-coordinationQMGrPort** parameter, you must also specify the **-coordinationQMGrHost** parameter.

### **-coordinationQMGrChannel** *coordination\_qmgr\_channel*

Optional. The channel name used to connect to the coordination queue manager. If you specify the **-coordinationQMGrChannel** parameter, you must also specify the **-coordinationQMGrHost** parameter.

### **-credentialsFile** *file\_path*

Optional. The full file path of an existing or new credentials file, to which the IBM MQ authentication details are added.

This command supports the addition of a set of IBM MQ authentication details, to a named Managed File Transfer credentials file. Use this command when IBM MQ connection authentication has been enabled. If you update the existing details, you must use the **-f** force parameter.

### **-userid** *user\_id*

Optional. The user ID used to associate the credential details. If you do not specify a user ID, the credential details will apply to all users. You must also specify the **-credentialsFile** parameter.

### **-f**

Optional. Forces an overwrite of the existing coordination queue manager configuration with the details specified in this command.



**-default**

Optional. Updates the default configuration options to the options associated with the coordination queue manager specified in this command.

**-? or -h**

Optional. Displays command syntax.

**Example**

In this example, the required objects are set up for a coordination queue manager called QM\_SATURN, which is connected to in client mode:

```
fteSetupCoordination -coordinationQMgr QM_SATURN
-coordinationQMgrHost myhost.ibm.com -coordinationQMgrPort 1415
-coordinationQMgrChannel SYSTEM.DEF.SVRCONN
```

**Return codes****0**

Command completed successfully.

**1**

Command ended unsuccessfully.

**Related concepts**

[Configuration options](#)

[SSL/TLS properties for the coordination.properties file](#)

**Related tasks**

[Configuring the coordination queue manager for MFT](#)

 [Configuring MQMFTCredentials.xml on z/OS](#)

**Related reference**

[The MFT agent.properties file](#)

**fteShowAgentDetails (display MFT agent details)**

Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.


**Purpose**

You can run the **fteShowAgentDetails** command from any system that can connect to the coordination queue manager. This command uses the `coordination.properties` file to connect to the coordination queue manager.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see [Configuration options](#).

The agent status information that is produced by this command is generated from the status messages that the agent publishes to the `SYSTEM.FTE` topic. These messages are described in [“MFT agent status message format” on page 2615](#). The status information that is produced by the **fteShowAgentDetails** command gives the agent status at the time when the last status message was published. The frequency of these status messages depends on the value of the **agentStatusPublishRateLimit** property. For more information, see [The MFT agent.properties file](#).

Specify the optional **-d** parameter for this command if you want to see diagnostic information about a local agent. This information includes current transfers, scheduled transfers, monitors, and agent queue depths. You can use this information to determine the health and status of a local agent.

 For z/OS, from IBM MQ 9.0.2 and IBM MQ 9.0.0 Fix Pack 1, the **-d** parameter can only be specified if the **fteShowAgentDetails** command is run by:

- The same userid that the agent process is running as.
- Members of the group that is specified by the agent property **adminGroup**.

For more information, see the **adminGroup** property in [The MFT agent.properties file](#).

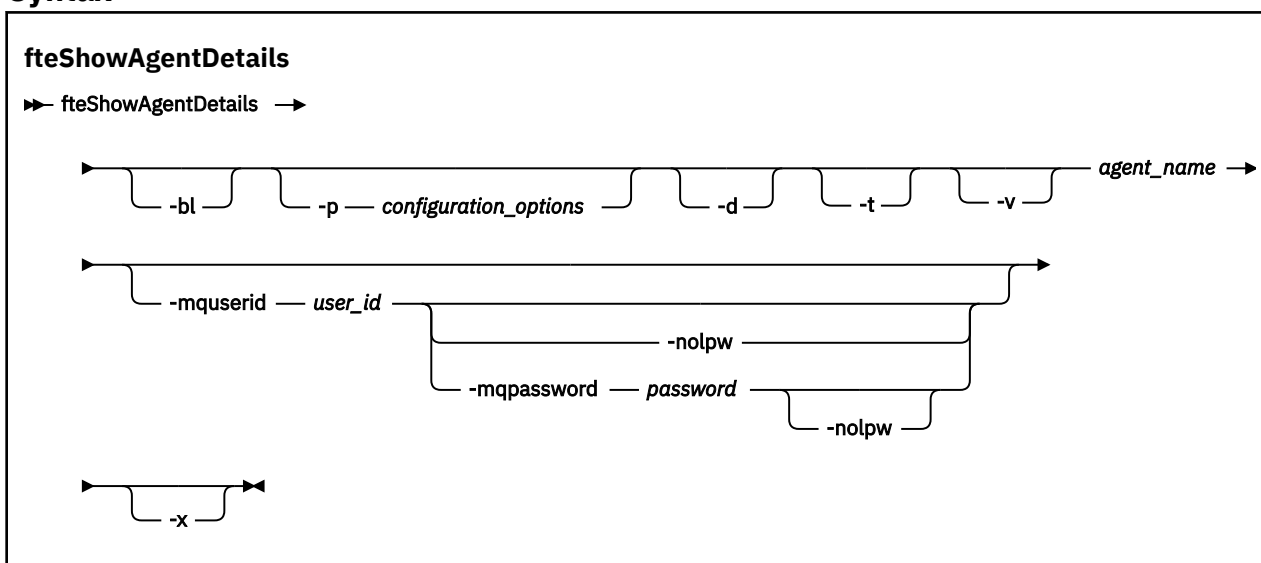
The status of the agent process controller and queue manager is available if you run the command on the same system as the agent. You can use this information to help with problem determination. Also, when you run the command on the same system as the agent, more detailed agent status information is available for the case where the agent ended unexpectedly.

For a list of the possible agent status values and their meanings, see [“MFT agent status values” on page 2485](#).

For a list of the possible status values for the agent process controller and their meanings, see [“MFT agent process controller status values” on page 2488](#).

For a list of agent trace values and FFDC specifications and their meanings, see [“fteSetAgentTraceLevel \(modify current trace level for an agent\)” on page 2118](#)

## Syntax



## Parameter

### -bl

Optional. Additionally outputs the product build level for the agent.

### -p *configuration\_options*

Optional. This parameter determines the set of configuration options that is used to issue the request to display the details of an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### -d

Optional. Specifies that diagnostic information is collected for *agent\_name*.

The diagnostic information is output to the console, and written to a file called `diagnostics.<yyyyMMdd>.<HHmmss>.<ssss>.<number>.properties` in the directory `MQ_DATA_PATH\mqft\logs\coordination_qmgr_name\agents\agent_name\logs`. A maximum of five historical files containing diagnostic information about an agent will be created. If five historical files have been created for an agent when the **fteShowAgentDetails** command is run

with the **-d** parameter specified, the oldest historical file will be deleted and replaced with a new file containing the latest diagnostic information about the agent.

You can use this parameter only when the agent is running, and on the local system.

**-t**

Optional. Specifies terse mode. From IBM MQ 9.1, the output includes the **Status Age** information by default. If you do not want to see this information, you can issue the command with the **-t** parameter. For more information, see [What to do if an agent is shown as being in an UNKNOWN state.](#)

**-v**

Optional. Specifies verbose mode, which generates additional output for the agent. These include host name, product version, product build level, trace level, and First Failure Data Capture (FFDC) specification, and a list of transfer states for each of the current source and destination transfers.

The current transfer information is obtained from the agent status publication, which is described in [“MFT agent status message format” on page 2615](#). Therefore this transfer information is only accurate to within the value of the `agentStatusPublishRateLimit` property. For more details about this property, see [The MFT agent.properties file](#).

**agent\_name**

Required. The name of the Managed File Transfer Agent that you want to display.

**-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the coordination queue manager.

**-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using [MQCSP](#) authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

**V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**-? or -h**

Optional. Displays command syntax.

**-x**

Optional. Provides information about all the active and, if they exist, standby instances.

## Example

In the following example, running `bindings agent`, issuing the **fteShowAgentDetails** command locally to the agent:

```
fteShowAgentDetails -v AGENT1
```

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name:                AGENT1
  Type:                Standard
  Description:
```

```

Operating System: Windows Server 2003
Time Zone: Greenwich Mean Time
Product Version: 7.5
Build Level: f000-20120312-0957
Trace Level: com.ibm.wmqfte.Agent=all
              com.ibm.wmqfte.common=all
              com.ibm.wmqfte.common:Any
              com.ibm.wmqfte.Agent:1

Trace FFDC:

Agent Controller Information:
  Controller type: MQMFT Process Controller
  Status: STARTED
  Status Details: The agent process controller has
                  started the agent process.

  Agent Restarts within Interval: 0
  Total Agent Restart Count: 0

Agent Availability Information:
  Status: READY
  Status Details: The agent is running and is publishing
                  its status at regular intervals. The
                  last update was received within the
                  expected time period. The agent is
                  ready to process transfers, but none
                  are currently in progress.

Queue Manager Information:
  Name: QM1
  Transport: Bindings
  Last Status Reported: AVAILABLE (Last Error MQRC: 0)
  Status Details: The queue manager is available.

Maximum Number of Running Source Transfers: 25
Maximum Number of Queued Source Transfers: 1000
Source Transfer States:
  No current transfers

Maximum Number of Running Destination Transfers: 25
Destination Transfer States:
  TransferId State
  414d51204d49414f57202020202020202020822c5b4a648c0b20 progress
  414d51204d49414f57202020202020202020822c5b4a346c0b20 progress

```

In the following example, QMGR1 is the non-default coordination queue manager used as input for the configuration options, and diagnostic information is requested with the **-d** parameter. The **fteShowAgentDetails** command is issued on an IBM MQ system with a local agent:

```

fteShowAgentDetails -p QMGR1 -d AGENT1
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name: AGENT1
  Type: Standard
  Description:
  Operating System: Linux
  Time Zone: Greenwich Mean Time

Agent Controller Information:
  Controller type: MQMFT Process Controller
  Status: STARTED
  Status Details: The agent process controller has started
                  the agent process.

  Agent Restarts within Interval: 0
  Total Agent Restart Count: 0

Agent Availability Information:
  Status: ACTIVE
  Status Details: The agent is running and is publishing
                  its status at regular intervals. The last
                  update was received within the expected
                  time period. The agent is currently
                  processing one or more transfers.

Queue Manager Information:
  Name: QMGR1
  Transport: Client
  Host: host1.hursley.ibm.com
  Port: 2021
  Channel: SYSTEM.DEF.SVRCONN
  Last Status Reported: UNKNOWN
  Status Details: Information about the queue manager is

```

not available because the agent has a client connection to the queue manager.

Agent Diagnostic Information:

Command Handler Diagnostics:

Last Command Queue Read Time: 2012-07-30T15:23:10.705Z  
Pending Command Queue Size: 0  
Last Internal Command Type: Resync Request (from sender) -  
414d5120514d43414e4445202020202079e20f5064230010  
Last Internal Command Time: 2012-07-30T14:17:10.506Z  
Last External Command Type: New Monitor Request  
Last External Command Time: 2012-07-30T14:10:57.751Z  
Diagnostic Properties File name: C:\Program Files (x86)\IBM\WebSphere  
MQ\mqft\logs\MUNGEE\agents\MUNGEE\logs\diagnostics.20121031.083420.0477.1.properties

Command Handler Worker Thread 0 Diagnostics:  
Status: Waiting

Command Handler Worker Thread 1 Diagnostics:  
Status: Waiting

Command Handler Worker Thread 2 Diagnostics:  
Status: Waiting

Command Handler Worker Thread 3 Diagnostics:  
Status: Waiting

Command Handler Worker Thread 4 Diagnostics:  
Status: Waiting

File Transfer Diagnostics:

Source Transfers: 1  
Destination Transfers: 2

File Transfer 0 Diagnostics:

Transfer Id: 414d5120514d43414e4445202020202079e20f5064230010  
Role: SOURCE  
State: ReSynchronisingTransfer  
Status: INACTIVE  
Start Time: Not started  
Retry Count: 0  
CheckPoint Index: 0  
CheckPoint Position: 0

File Transfer 1 Diagnostics:

Transfer Id: 414d5120514d43414e44452020202020c8fbd54f144f0d20  
Role: DESTINATION  
State: RunningTransfer  
CheckPoint Index: 0  
CheckPoint Position: 0  
Write Index: 0  
Write Position: 0

File Transfer 2 Diagnostics:

Transfer Id: 414d5120514d43414e4445202020202079e20f5086020010  
Role: DESTINATION  
State: RunningTransfer  
CheckPoint Index: 9  
CheckPoint Position: 0  
Write Index: 3  
Write Position: 140923

Monitor 0 Diagnostics:

Name: MONITOR1  
Status: STARTED  
Resource Type: directory  
Resource: /tmp/monitor  
Poll Interval: 1 minutes  
Batch Size: 2  
Condition: Match  
Pattern: \* (wildcard)  
Executing: false  
Last Execute Start Time: 2012-04-04T16:19:01.852Z  
Last Execute End Time: 2012-04-04T16:19:01.852Z  
Last Execute Match Count: 0

Schedule 1 Diagnostics:

Id: 1  
Next Trigger Time: 2012-07-17T16:00+0100

```

Occurrences So Far:      14
Repeat Interval:        hours
Repeat Frequency:       5
Source Agent:           AGCANDE
Destination Agent:      AGCANDE
Source File:             /tmp/source/a.txt, ...
Destination File:       /tmp/dest/a.txt, ...

```

In the following example, stopped bindings agent, issuing the **fteShowAgentDetails** command remotely from the agent:

```

fteShowAgentDetails AGENT2
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name:                AGENT2
  Type:                Standard
  Description:
  Operating System:    Linux
  Time Zone:           Greenwich Mean Time

Agent Controller Information:
  Controller type:     MQMFT Process Controller
  Status:              UNKNOWN
  Status Details:     Information about the agent controller
                     is not available, either because the
                     agent is not running or the agent is
                     running on a different system.

  Agent Restarts within Interval: 0
  Total Agent Restart Count:      0

Agent Availability Information:
  Status:              STOPPED
  Status Details:     The agent has been stopped. It was shut
                     down in a controlled manner.

Queue Manager Information:
  Name:                QM2
  Transport:           Bindings
  Last Status Reported: UNKNOWN
  Status Details:     Information about the queue manager is
                     not available, either because the agent
                     is not running or the agent is running
                     on a different system.

```

In the following example, bindings agent is waiting to restart with the agent queue manager stopped. The agent has already been restarted once before Total Agent Restart Count: 1, possibly due to a previous agent queue manager restart:

**Note:** The Last Error MQRC against the Last Status Reported for the queue manager information; this information will remain even when the queue manager becomes available.

```

fteShowAgentDetails AGENT1
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name:                AGENT1
  Type:                Standard
  Description:
  Operating System:    Windows Server 2003
  Time Zone:           Greenwich Mean Time

Agent Controller Information:
  Controller type:     MQMFT Process Controller
  Status:              WAITING
  Status Details:     The agent process controller is waiting
                     for the queue manager to become
                     available before starting the agent.

  Agent Restarts within Interval: 0
  Total Agent Restart Count:      1

Agent Availability Information:
  Status:              STOPPED
  Status Details:     The agent has been stopped. It was shut
                     down in a controlled manner.

Queue Manager Information:
  Name:                QM1
  Transport:           Bindings

```

```

Last Status Reported:      UNAVAILABLE (Last Error MQR: 2059)
Status Details:           The queue manager is unavailable. It
                           might be that the queue manager has not
                           been started or an incorrect queue
                           manager name has been configured. Look
                           up the MQ reason code reported against
                           the status to understand the problem.

```

In the following example, the client mode agent has just ended unexpectedly and the agent process controller tries to recover the situation by restarting it after a delay, specified by the `maxRestartDelay` agent property value. The default `maxRestartDelay` agent property value is `-1`, and this causes the agent process controller to terminate; hence in this example the `maxRestartDelay` property value must have been set to a value greater than 0. The `Current Agent Restart Count: 4` implies that there have been 4 restarts within the `maxRestartInterval` agent property time period. If the `maxRestartCount` agent property is 4 then after 4 restarts within the `maxRestartInterval`, the agent process controller will wait for `maxRestartDelay` seconds before restarting the agent, which is the case here. The `Total Agent restart Count: 8` suggests that this has occurred before. This example is not typical and you would only expect to see the agent ending unexpectedly if the agent runs out of memory, or a custom user exit has caused some sort of runtime error. Full details as to why the agent ended unexpectedly are in the agent's `output0.log` file:

```

fteShowAgentDetails AGENT3
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name:                AGENT3
  Type:                Standard
  Description:
  Operating System:    Windows Server 2003
  Time Zone:           Greenwich Mean Time

Agent Controller Information:
  Controller type:     MQMFT Process Controller
  Status:              RECOVERING
  Status Details:     The agent process unexpectedly stopped
                     and the process controller will attempt
                     to restart it.

  Current Agent Restart Count: 4
  Total Agent Restart Count: 8

Agent Availability Information:
  Status:              ENDED UNEXPECTEDLY
  Status Details:     The agent has ended unexpectedly due to
                     an unrecoverable problem. The agent
                     will be automatically restarted.

Queue Manager Information:
  Name:                QM3
  Transport:           Client
  Host:                host3.hursley.ibm.com
  Port:                3031
  Channel:             SYSTEM.DEF.SVRCONN

```

In the following example, the results for a `Connect:Direct` bridge agent are displayed:

```

fteShowAgentDetails AG_CD1
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
  Name:                AG_CD1
  Type:                Connect:Direct bridge
  Description:
  Connect:Direct Node Name:  CDNODE
  Connect:Direct Node Host: localhost:1363
  Operating System:    Windows Server 2003
  Time Zone:           Greenwich Mean Time

Agent Controller Information:
  Controller type:     MQMFT Process Controller
  Status:              UNKNOWN
  Status Details:     Information about the agent controller
                     is not available, either because the
                     agent is not running or the agent is
                     running on a different system.

  Agent Restarts within Interval: 0
  Total Agent Restart Count: 0


```

```

Agent Availability Information:
Status: STOPPED
Status Details: The agent has been stopped. It was shut
down in a controlled manner.

Queue Manager Information:
Name: QM_JUPITER
Transport: Bindings
Last Status Reported: UNKNOWN
Status Details: Information about the queue manager is
not available, either because the agent
is not running or the agent is running
on a different system.

```

 In the following example, an agent running on z/OS is registered with the Automatic Restart Manager (ARM):

```

fteShowAgentDetails AGENTZ
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Agent Information:
Name: AGENTZ
Type: Standard
Description:
Operating System: z/OS
Time Zone: Greenwich Mean Time

Agent Controller Information:
Controller Type: z/OS Automatic Restart Manager (ARM)
Agent registered with ARM: Yes (ELEMTYPE: SYSBFGAG, ELEMENT: AGENTZ)
Agent Restarted: No

Agent Availability Information:
Status: READY
Status Details: The agent is running and is publishing
its status at regular intervals. The last
update was received within the expected
time period. The agent is ready to
process transfers, but none are currently
in progress.

Queue Manager Information:
Name: ZQM
Transport: Bindings
Last Status Reported: AVAILABLE
Status Details: The queue manager is available.

```

From IBM MQ 9.1.4 the output from the command displays information of all the available instances when you specify the **-x** parameter. Note that if you do not specify the **-x** parameter the output is unchanged from the current format

```

24-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Host Name: Type: Version:
9.122.123.124 Active 9.1.4.0
myhost.ibm.com Standby 9.1.4.0
10.20.40.123 Standby 9.1.4.0

```

If the agent started in high availability mode has no standby instances running, the output contains information about the active instance only. For example:

```

24-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Host: Type: Version:
9.122.123.124 Active 9.1.4.0

```

If you specify the **-x** parameter and the agent has been started as normal, that is, not in high availability mode, you receive the following message:

```

BFGCL0790I: No standby instance information available for agent '<agent name>'.
The agent is either not running or is not publishing status.

```



## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

### Related reference

[“fteListAgents \(list the MFT agents for a coordination queue manager\)” on page 2085](#)

Use the **fteListAgents** command to list all of the Managed File Transfer agents that are registered with a particular coordination queue manager.

[“MFT agent status values” on page 2485](#)

The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

[“MFT agent process controller status values” on page 2488](#)

The **fteShowAgentDetails** command produces agent process controller status information. There are several possible values for this status.

## fteShowLoggerDetails (display MFT logger details)

Use the **fteShowLoggerDetails** command to display the details of a particular Managed File Transfer logger.

### Purpose

You must run the **fteShowLoggerDetails** command on the same system as the logger. It displays the status of the logger process controller and the logger queue manager, which you can use to help with problem determination. The **fteShowLoggerDetails** command lists the following details for a particular Managed File Transfer logger:

- Logger controller status.
- Logger restarts within interval
- Total logger restart count
- Logger availability status
- Logger queue manager name
- Logger queue manager transport type
- Logger queue manager last status reported (applies to binding transport mode only)

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [Configuration options](#) for more information.

For a list of the possible logger status values and their meanings, see [“MFT logger status values” on page 2489](#).

For a list of the possible status values for the logger process controller and their meanings, see [“MFT logger process controller status values” on page 2489](#).

From IBM MQ 9.1.0, the output of the command displays connection information that the logger is using to connect to the queue manager. If the logger is connected in clients mode, the output for:

### Last Status Reported

Is shown as UNKNOWN

### Status details

Is shown as Information about the queue manager is not available because the logger has a client connection to queue manager.

## Syntax

### **fteShowLoggerDetails**

```
► fteShowLoggerDetails ———— configuration_options ———— logger_name ◄◄
```

## Parameter

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to issue the request to display the details of an logger. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### **logger\_name**

Required. The name of the Managed File Transfer logger that you want to display.

### **-? or -h**

Optional. Displays command syntax.

## Example

In this example, a started logger, issuing the **fteShowLoggerDetails** command locally to the logger:

```
fteShowLoggerDetails LOGGER1
```

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Logger Controller Information:
  Status:                               STARTED
  Status Details:                        The logger process controller has
                                          started the logger process.
  Logger Restarts within Interval:      0
  Total Logger Restart Count:           0

Queue Manager Information:
  Name:                                  QM_gbthink
  Transport:                             Bindings
  Last Status Reported:                  AVAILABLE
  Status Details:                        The queue manager is available.
```

In this example, a logger waiting due to an unavailable queue manager, issuing the **fteShowLoggerDetails** command locally to the logger:

```
fteShowLoggerDetails LOGGER2
```

```
5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Logger Controller Information:
  Status:                               WAITING
  Status Details:                        The logger process controller is
                                          waiting for the queue manager to
                                          become available before starting the
                                          logger.
  Logger Restarts within Interval:      0
  Total Logger Restart Count:           0

Logger Availability Information:
  Status:                               STOPPED
  Status Details:                        The logger has been stopped. It was
                                          shut down in a controlled manner.
```

```

Queue Manager Information:
  Name:                QM_gbthink
  Transport:           Bindings
  Last Status Reported: UNAVAILABLE (Last Error MQRC: 2059)
  Status Details:     The queue manager is unavailable. It
                      might be that the queue manager has
                      not been started or an incorrect
                      queue manager name has been
                      configured. Look up the MQ reason code
                      reported against the status to
                      understand the problem.

```

## z/OS

In this example on z/OS, a running logger (not registered with ARM):

```
fteShowLoggerDetails loggerv8
```

```

5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Logger Controller Information:
  Controller Type:      z/OS Automatic Restart Manager (ARM)
  Registered with ARM: No
  Restarted:           n/a

Queue Manager Information:
  Name:                FT8E
  Transport:           Bindings
  Last Status Reported: AVAILABLE
  Status Details:     The queue manager is available.

```

## z/OS

In this example on z/OS, a logger that is not running, or running on a different system:

```
fteShowLoggerDetails loggerv8
```

```

5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
Logger Controller Information:
  Controller Type:      UNKNOWN

Queue Manager Information:
  Name:                FT8E
  Transport:           Bindings
  Last Status Reported: UNKNOWN
  Status Details:     Information about the queue manager is
                      not available, either because the
                      logger is not running, or the logger
                      is running on a different system.

```

## Return codes

- 0** Command completed successfully.
- 1** Command ended unsuccessfully.

## Related reference

[“MFT logger status values” on page 2489](#)

The **fteShowLoggerDetails** commands produce logger status information. There are several possible values for this status.

[“MFT logger process controller status values” on page 2489](#)

The **fteShowLoggerDetails** command produces logger process controller status information. There are several possible values for this status.

## fteStartAgent (start an MFT agent)

The **fteStartAgent** command starts a Managed File Transfer agent from the command line.

### Purpose

Use the **fteStartAgent** command to start a Managed File Transfer agent. You must start an agent before you can use it to perform file transfers. The **fteStartAgent** command starts an agent on the system where you issue the command: you cannot start an agent on a remote system.

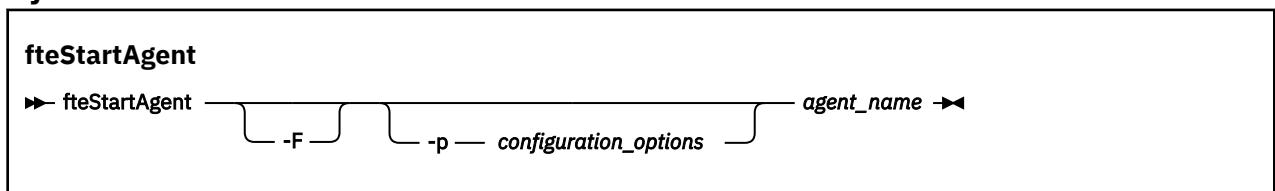
For IBM WebSphere MQ 7.5 or later, the agent process controller manages starting the agent. However, the agent process controller may wait for a period of time, for example where there have been a high rate of agent failures, before attempting to start the agent again. As an IBM MQ administrator you can use **fteStartAgent** command to override this wait and initiate a start of the agent. If the agent process controller was waiting for the queue manager to become available this command will also initiate the agent process controller attempting to reconnect to the queue manager.

**Windows** If you have configured the agent to run as a Windows service by using the [fteCreateAgent](#) or [fteModifyAgent](#) command, running the **fteStartAgent** command starts the Windows service.

This command returns an error if the agent does not start or is already started. The agent communicates with its queue manager based on the values defined in the `agent.properties` file.

Specify the optional **-p** parameter for this command only if you want to use a different set of configuration options than your default set. See [The MFT agent.properties](#) file for more information.

### Syntax



### Parameters

#### **-F**

Optional. This parameter runs the agent daemon as a foreground process. The default is for the agent daemon to run in the background.

If you are running on Windows, and you have configured the agent to run as a Windows service by using the **fteCreateAgent** or **fteModifyAgent** commands, the **-F** parameter overrides this configuration.

#### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to issue the request to start an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

#### **agent\_name**

Required. The name of the Managed File Transfer agent that you want to start.

#### **-? or -h**

Optional. Displays command syntax.

## Example

In this example, AGENT2 is started and runs in the foreground.

```
fteStartAgent -F AGENT2
```

In the following example (for AIX and Linux systems), AGENT2 is started with a non-default coordination queue manager, QM\_SATURN:

```
./fteStartAgent -p QM_SATURN AGENT2
```

You can also run the command by specifying the path to **fteStartAgent** as follows:

```
path/fteStartAgent agentname
```

## Return codes

### 0 (RC\_SUCCESS)

Command completed successfully.

### 1 (RC\_FAILURE)

Command ended unsuccessfully.

### 78 (RC\_CONFIG)

A configuration error was encountered.

### 79 (RC\_API\_ERROR)

A MFT exception occurred.

### 80 (RC\_IO\_ERROR)

A Java IOException occurred.

### 81 (RC\_IPC\_ERROR)

An MFT interprocess communication error occurred.

## Responses

In some circumstances, you might see error messages after running the **fteStartAgent** command:


- If you run the **fteStartAgent** command and see the following error message, your environment probably has additional library paths that conflict with Managed File Transfer:

```
BFGCL0001E: An internal error has occurred. The exception was: 'CC=2;RC=2495;AMQ8568:  
The native JNI library 'mqjbnd' was not found. [3=mqjbnd]
```

If the LD\_LIBRARY\_PATH or LIBPATH environment variable is set to reference a 64-bit version of the library before the 32-bit version when the agent is running with a 32-bit version of Java (as is currently the case for most platforms), this error occurs.

To resolve this issue, set the Managed File Transfer agent property javaLibraryPath to reference the correct location for the library. For example, for mqjbnd on AIX, set to: /usr/mqm/java/lib. For mqjbnd on Linux, set to: /opt/mqm/java/lib

## Related tasks

 [Starting an MFT agent on z/OS](#)

[Starting an MFT agent as a Windows service](#)

[Listing MFT agents](#)

[Stopping an MFT agent](#)

## fteStartLogger (start an MFT logger)

The **fteStartLogger** command starts a Managed File Transfer logging application.

### Purpose

Use the **fteStartLogger** command to start a logger.

The logger can be either a file or database application that runs on the same system as the coordination queue manager. For more information, see [Configuring an MFT logger](#).

For IBM WebSphere MQ 7.5, or later, the logger process controller manages starting the logger. However, the logger process controller might wait for a period of time, for example where there has been a high rate of logger failures, before attempting to start the logger again. As an IBM MQ administrator you can use the **fteStartLogger** command to override this wait and initiate a start of the logger. If the logger process controller was waiting for the queue manager to become available, this command will also initiate the logger process controller attempting to reconnect to the queue manager.

**Windows** If you have configured a logger to run as a Windows service by using the [fteModifyLogger](#) command, running the **fteStartLogger** command starts the Windows service.

This command returns an error if the logger does not start or is already started.

The logger communicates with its queue manager based on the values defined in the `logger.properties` file.

Specify the **-p** parameter for this command only if you want to use a set of configuration options different from the default. For more information on logger properties, see [MFT logger configuration properties](#)

### Syntax

#### fteStartLogger

```
► fteStartLogger -p configuration_options -F logger_name ◄
```

### Parameters

#### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to issue the request to start a logger. Use the name of a non-default coordination queue manager as the input for this parameter. **fteStartLogger** then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

#### **-F**

Optional. Runs the logger as a foreground process (rather than as the default background process). If you have configured the logger to run as a Windows service by using the **fteModifyLogger** command, the **-F** parameter overrides this configuration.

#### **-? or -h**

Optional. Displays command syntax.

#### **logger\_name**

Required. The name of the Managed File Transfer logger you want to start.

## Example

In this example, a logger has previously been created named `logger1`. This command shows how the logger can be started as a foreground process:

```
fteStartLogger -F logger1
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related concepts

[MFT logger error handling and rejection](#)

## Related tasks

[Configuring an MFT logger](#)

## Related reference

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 2100](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the `mqm` group, and you must first stop the logger by using the **fteStopLogger** command.

[“fteStopLogger \(stop an MFT logger\)” on page 2148](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

**V 9.3.0**

## **fteStartMonitor (start an MFT resource monitor)**

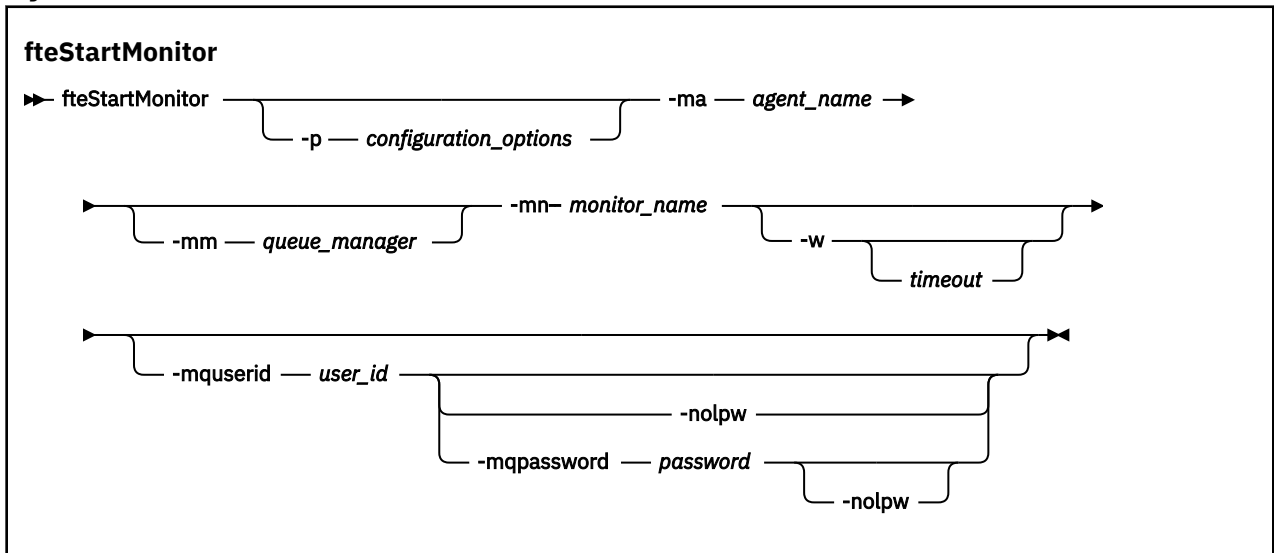
The **fteStartMonitor** command starts a Managed File Transfer resource monitor from the command line.

## Purpose

From IBM MQ 9.3.0, you can use the **fteStartMonitor** command to start a resource monitor without needing to stop or restart an agent.

You can run this command from any system where the Managed File Transfer commands component is installed, which means that you can start a resource monitor from anywhere, and are not restricted to the system where the agent that owns the resource monitor is running.

## Syntax



## Parameters

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to start a resource monitor. Use the name of a set of configuration options as the value for the **-p** parameter. By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

### **-ma agent\_name**

Required. The name of the agent running the resource monitor operation.

### **-mm queue\_manager**

Optional. The name of the queue manager that the agent is connected to.

The **fteStartMonitor** command connects to the command queue manager. If the command queue manager is also the agent queue manager for the monitoring agent, then the **-mm** parameter is optional, otherwise you must specify the agent queue manager with the **-mm** parameter.

### **-mn monitor\_name**

Required. The name of the resource monitor that you want to start.

### **-w timeout**

Optional. Specifies to wait for up to timeout seconds for the agent to respond. If you do not specify a timeout, or specify a timeout value of minus one, then the command waits forever for the agent to respond. If you do not specify this option, then the default is to wait up to five seconds for the agent to respond.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

The command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.



When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password which, will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### Example 1

The following command starts a resource monitor in an agent running on the same machine:

```
fteStartMonitor -mn MNTR -ma SOURCE
```

This command outputs the following messages:

```
BFGCL0816I: A request to start resource monitor 'MNTR' of agent 'SOURCE' has been issued.  
BFGCL0251I: The request has successfully completed.
```

The following event is logged in the agent's output0.log:

```
BFGDM0032I: Monitor MNTR has been started.
```

External return code = 0

### Example 2

The following command starts a resource monitor in an agent running on a different machine:

```
fteStartMonitor -mn MNTR -ma SOURCE -mm SRCQM
```

This command outputs the following messages:

```
BFGCL0816I: A request to start resource monitor 'MNTR' of agent 'SOURCE' has been issued.  
BFGCL0251I: The request has successfully completed.
```

The following event is logged in the agent's output0.log:

```
BFGDM0032I: Monitor MNTR has been started.
```

### Examples of other messages

If you receive any of the following messages, you can check the state of the agent by using the **fteListMonitors** command with the **-v** parameter. The **-v** parameter generates verbose output that includes additional information about the status of the resource monitor, including whether the resource monitor is started or stopped, the directory resource path that is being monitored and the trigger conditions.

#### The resource monitor is already in the started state

The command outputs the following messages:

```
BFGCL0816I: A request to start resource monitor 'MNTR' of agent 'SOURCE' has been issued.  
BFGCL0814I: Resource monitor 'MNTR' of agent 'SOURCE' is already running.
```

#### The resource monitor is in an agent that is stopped

The command outputs the following messages:

```
BFGCL0816I: A request to start resource monitor 'MNTR' of agent 'SOURCE' has been issued.  
BFGCL0253W: No acknowledgement to command from agent within timeout.
```

#### The specified agent is unknown

The command outputs the following message:

```
BFGUB0009E: The following required property file is missing:  
"/root/mftdata/mqft/config/MFTHAQM/agents/UNKNOWNAGENT/agent.properties"
```

#### The specified resource monitor cannot be found on the specified agent

The command outputs the following messages:

```
BFGCL081608I: A request to start resource monitor 'UNKNOWNMONITOR' of agent 'IJQ' has been issued.  
BFGCL0250E: The monitor could not be found on the given agent.
```

## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related tasks

[Starting an MFT resource monitor](#)

## Related reference

[“fteListMonitors \(list MFT resource monitors\)” on page 2089](#)

Use the **fteListMonitors** command to list all of the existing resource monitors in a Managed File Transfer network using the command line.

[“fteStopMonitor \(stop an MFT resource monitor\)” on page 2150](#)

The **fteStopMonitor** command stops a Managed File Transfer resource monitor from the command line.

## fteStopAgent (stop an MFT agent)

Use the **fteStopAgent** command to either stop a Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

## Purpose

When you stop an agent by using the **fteStopAgent** command, you can either allow the agent to complete its current file transfer before stopping, or stop the agent immediately even if the agent is currently transferring a file. When the agent has stopped, you cannot use that agent to transfer files until you restart the agent.

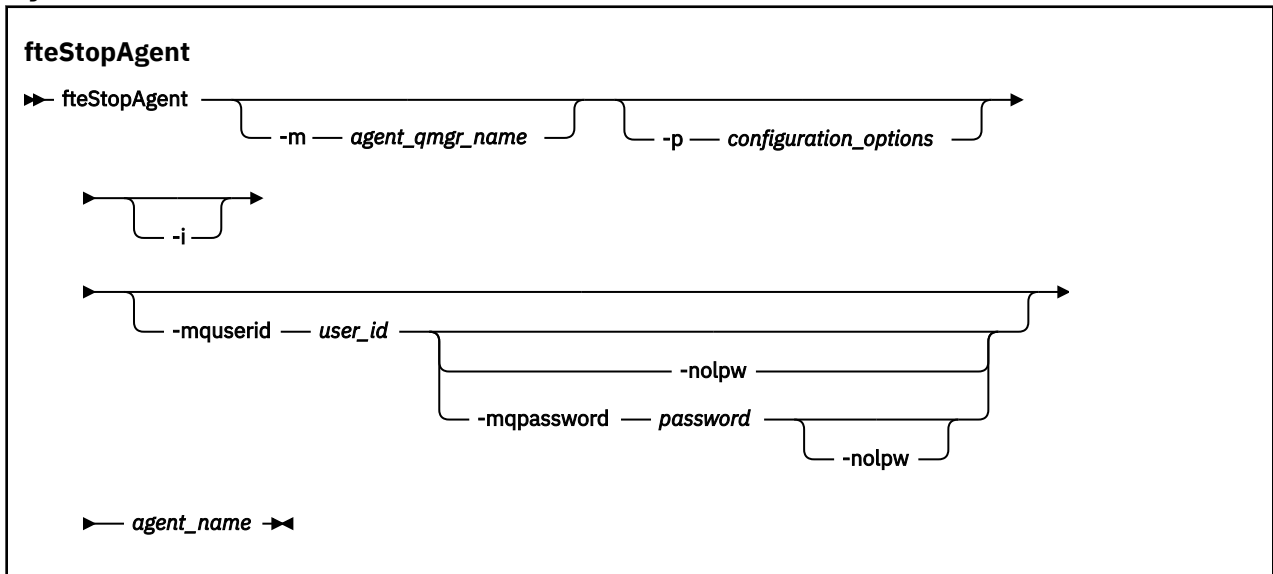
If the agent you want to stop is connected to the IBM MQ network, you can run the **fteStopAgent** command from any system that can connect to the IBM MQ network and route to the agent queue manager. Specifically for the command to run, you must have installed and configured a Managed File Transfer component (either Service or Agent) on this system to communicate with the IBM MQ network. If no connectivity details are available, a bindings mode connection is made to the default queue manager on the local system. If `command.properties` does not exist then an error is generated.

If the agent you want to stop is not connected to the IBM MQ network, for example if the IBM MQ network is not currently available, you can only run the **fteStopAgent** command from the system that the agent is running on. In order to stop an agent that is not connected to the IBM MQ network you must run the **fteStopAgent** command from the same user the agent is running as. Alternatively, if the agent is running on a Windows system you can run the command as an administrator.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See [The MFT agent.properties file](#) for more information.

**Windows** If your agent is running as a Windows service, running the **fteStopAgent** command stops the Windows service. For more information, see [Starting an MFT agent as a Windows service](#).

## Syntax



## Parameters

### **-m agent\_qmgr\_name**

Optional. The name of the queue manager that the agent that you want to stop is connected to.

If the agent is on a remote system, or if the agent is on the local system but you are not the user that started it, you must use the **-m** parameter, and have the appropriate authorities. For more information about authorities, see [Restricting group authorities for MFT-specific resources](#).

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to issue the request to stop an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

### **-i**

Optional. Immediately stops the agent. The agent does not complete any transfers that are currently in progress.

If you do not specify the **-i** parameter, the agent completes any transfers currently in progress but the agent does not start any new transfers.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-no1pw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-no1pw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-no1pw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**agent\_name**

Required. The name of the Managed File Transfer agent that you want to stop.

**-? or -h**

Optional. Displays command syntax.

**Example**

In this example the agent AGENT2 on queue manager QM\_JUPITER is stopped. The **-m** parameter is used because this queue manager that AGENT2 is connected to differs from the queue manager specified by the set of configuration options.

```
fteStopAgent -m QM_JUPITER AGENT2
```

**Return codes**

**0**

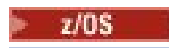
Command completed successfully.

**1**

Command ended unsuccessfully.

**Related tasks**

[Stopping an MFT agent](#)

 [Stopping an MFT agent on z/OS](#)

**Related reference**

[“fteStartAgent \(start an MFT agent\)” on page 2140](#)

The **fteStartAgent** command starts a Managed File Transfer agent from the command line.

**fteStopLogger (stop an MFT logger)**

The **fteStopLogger** command stops a Managed File Transfer logger.

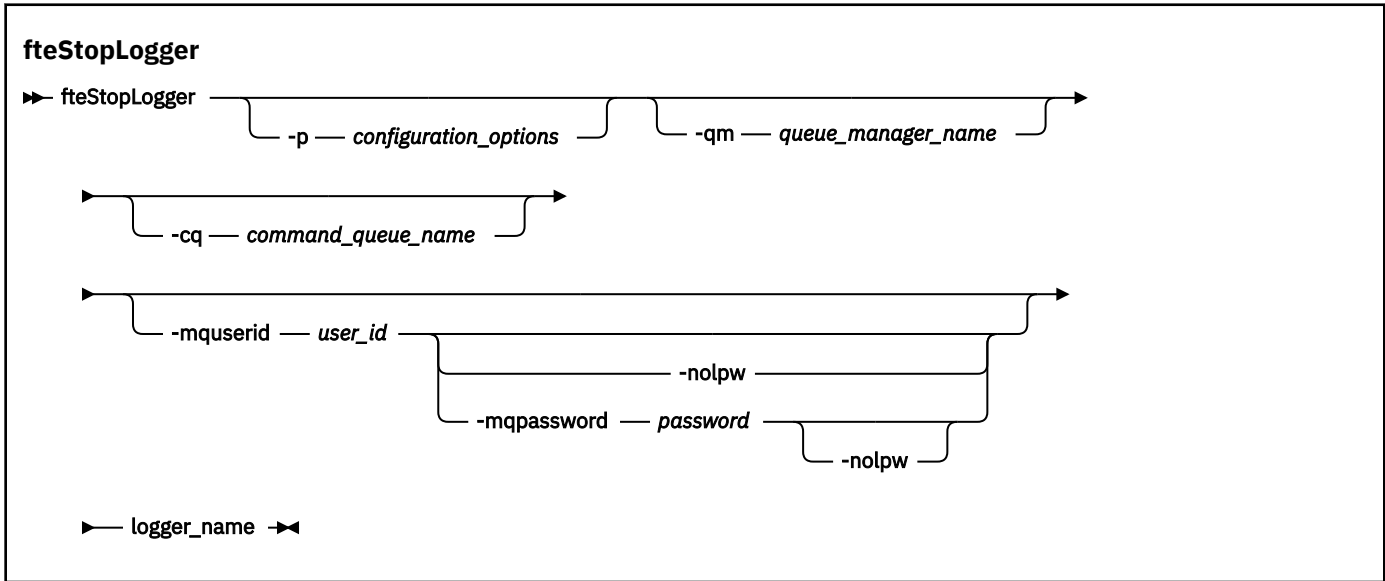
**Purpose**

Use the **fteStopLogger** command to stop a logger. The logger can be either a file logger, which records a history of managed file transfer activity to a file, or a database logger which records the history to a database.

**Additional notes about stopping a stand-alone logger process**

If your logger is running as a Windows service, running the **fteStopLogger** command stops the Windows service.

## Syntax



## Parameters

### **-p (configuration\_options)**

Optional. Determines the set of configuration options that is used to stop the logger. Use the name of a set of configuration options as the value for the **-p** parameter. By convention this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

### **-qm (queue\_manager\_name)**

Optional. By default, the logger's command queue is assumed to be on the coordination queue manager specified by the **-p** parameter (or its default). If you want to send logger commands to a command queue located elsewhere, use the **-qm** parameter to specify an alternative destination. In all cases, this command connects to the command queue manager indicated by the **-p** parameter, regardless of the message's ultimate destination.

### **-cq (command\_queue\_name)**

Optional. Specifies the command queue to send the stop message to. In most cases, loggers use the default queue name meaning this parameter is not necessary.

### **-mquserid (user\_id)**

Optional. Specifies the user ID to authenticate with the command queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**V 9.3.0** **V 9.3.0** From IBM MQ 9.3 onwards, the command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **V 9.3.0** **V 9.3.0** **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-no1pw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

**logger\_name**

Required. The name of the Managed File Transfer logger you want to stop.

**-? or -h**

Optional. Displays command syntax.

**Example**

In this example, a logger has previously been created named logger1 and is currently running. This command shows how the logger can be stopped:

```
fteStopLogger logger1
```

**Return codes**

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

**Related tasks**

[Configuring an MFT logger](#)

**Related reference**

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 2100](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

[“fteStartLogger \(start an MFT logger\)” on page 2142](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

**V 9.3.0 fteStopMonitor (stop an MFT resource monitor)**

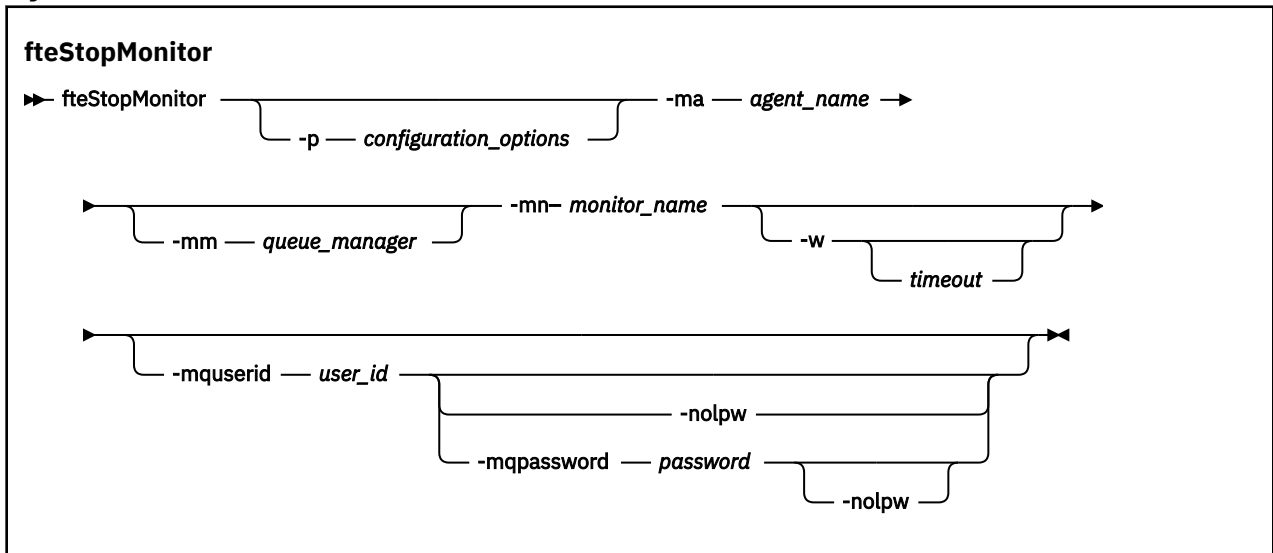
The **fteStopMonitor** command stops a Managed File Transfer resource monitor from the command line.

**Purpose**

From IBM MQ 9.3.0, you can use the **fteStopMonitor** command to stop a resource monitor without needing to stop or restart an agent.

You can run this command from any system where the Managed File Transfer commands component is installed, which means that you can stop a resource monitor from anywhere, and are not restricted to the system where the agent that owns the resource monitor is running.

## Syntax



## Parameters

### **-p configuration\_options**

Optional. This parameter determines the set of configuration options that is used to stop a resource monitor. Use the name of a set of configuration options as the value for the **-p** parameter. By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

### **-ma agent\_name**

Required. The name of the agent running the resource monitor operation.

### **-mm queue\_manager**

Optional. The name of the queue manager that the agent is connected to.

The **fteStopMonitor** command connects to the command queue manager. If the command queue manager is also the agent queue manager for the monitoring agent, then the **-mm** parameter is optional, otherwise you must specify the agent queue manager with the **-mm** parameter.

### **-mn monitor\_name**

Required. The name of the resource monitor that you want to stop.

### **-w timeout**

Optional. If you do not specify a *timeout* value, the default is to wait up to five seconds for the agent to respond.

The *timeout* value specifies the value in seconds for the agent to respond. If you specify a timeout value of minus one, the command waits forever for the agent to respond.

### **-mquserid user\_id**

Optional. Specifies the user ID to authenticate with the command queue manager.

### **-mqpassword password**

Optional. Specifies the password to authenticate with the coordination queue manager.

When the **-mqpassword** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

The command defaults to using MQCSP authentication, and sends the user ID and password to be authenticated to the queue manager in an MQCSP structure.

### **-nolpw**

Optional. Specify this parameter if the command connects to a queue manager that does not support passwords longer than 12 characters.

When the **-nolpw** parameter is specified, you must also specify the **-mquserid** parameter. If you specify **-mquserid** and **-nolpw**, but do not specify **-mqpassword**, you are prompted to supply the associated password, which will not be displayed.

**Note:** If you specify the **-nolpw** option, and the password is longer than 12 characters the command fails with error message BFGCL0829E.

### Example 1: resource monitor and agent on the same machine

The following command stops a resource monitor in an agent running on the same machine:

```
fteStopMonitor -mn MNTR -ma SOURCE
```

This command outputs the following messages:

```
BFGCL0813I: A request to stop resource monitor 'MNTR' of agent 'SOURCE' has been issued.  
BFGCL0251I: The request has successfully completed.
```

The following event is logged in the agent's output0.log.

```
BFGDM0069I: Monitor MNTR has been stopped.
```

### Example 2 - resource monitor and agent running on different machines

The following command stops a resource monitor in an agent running on a different machine:

```
fteStopMonitor -mn MNTR -ma SOURCE -mm SRCQM
```

This command outputs the following messages:

```
BFGCL0813I: A request to stop resource monitor 'MNTR' of agent 'SOURCE' has been issued.  
BFGCL0251I: The request has successfully completed.
```

The following event is logged in the agent's output0.log.

```
BFGDM0069I: Monitor MNTR has been stopped.
```

### Examples of other messages

If you receive any of the following messages, you can check the state of the agent by using the **fteListMonitors** command with the **-v** parameter. The **-v** parameter generates verbose output that includes additional information about the status of the resource monitor, including whether the resource monitor is started or stopped, the directory resource path that is being monitored and the trigger conditions.

#### The resource monitor is already in the stopped state

The command outputs the following messages:

```
BFGCL0813I: A request to stop resource monitor 'MNTR' of agent 'SOURCE' has been issued.  
BFGCL0815I: Resource monitor 'MNTR' of agent 'SOURCE' is already stopped.
```

#### The resource monitor is in an agent that is stopped

The command outputs the following messages:

```
BFGCL0813I: A request to stop resource monitor 'MNTR' of agent 'SOURCE' has been issued.  
BFGCL0253W: No acknowledgement to command from agent within timeout.
```

#### The specified agent is unknown

The command outputs the following message:

```
BFGUB0009E: The following required property file is missing:  
"/root/mftdata/mqft/config/MFTHAQM/agents/UNKNOWNAGENT/agent.properties"
```

#### The specified resource monitor cannot be found on the specified agent

The command outputs the following messages:

```
BFGCL0813I: A request to stop resource monitor 'UNKNOWNMONITOR' of agent 'IJQ' has been issued.  
BFGCL0250E: The monitor could not be found on the given agent.
```



## Return codes

**0**

Command completed successfully.

**1**

Command ended unsuccessfully.

## Related tasks

[Stopping an MFT resource monitor](#)

## Related reference

[“fteListMonitors \(list MFT resource monitors\)” on page 2089](#)

Use the **fteListMonitors** command to list all of the existing resource monitors in a Managed File Transfer network using the command line.

[“fteStartMonitor \(start an MFT resource monitor\)” on page 2143](#)

The **fteStartMonitor** command starts a Managed File Transfer resource monitor from the command line.

## MQIPT commands reference

Reference information about the syntax and usage of the various IBM MQ Internet Pass-Thru (MQIPT) commands.

### mqipt (start MQIPT)

Start IBM MQ Internet Pass-Thru (MQIPT).

#### Purpose

Use the **mqipt** command to start MQIPT. You can optionally specify a name to be given to the MQIPT instance that is being started.

#### Syntax

```
➤ mqipt — home_directory — -n mqipt_name — -sf encryption_key_file —
```

#### Required parameters

##### *home\_directory*

The MQIPT home directory, where the `mqipt.conf` configuration file is located. The MQIPT home directory can be specified as either an absolute path or relative to the current working directory of the command shell.

#### Optional parameters

##### **-n** *mqipt\_name*

The name to be given to the MQIPT instance that is being started. The name of the MQIPT instance is used to administer local instances of MQIPT with the **mqiptAdmin** command without needing to use a command port. If this parameter is not specified, the name of the MQIPT home directory is used as the name of the MQIPT instance.

##### **-sf** *encryption\_key\_file*

The name of a file that contains the password encryption key. The encryption key is used by MQIPT to access encrypted passwords in the `mqipt.conf` configuration file.

For information about the password encryption key file and other methods to specify the file name when starting MQIPT, see [Specifying the password encryption key](#).

For more information on encrypting stored passwords to be used by MQIPT, see [Encrypting stored passwords](#).

## mqiptAdmin (administer MQIPT)

Issue an administrative command to an active instance of IBM MQ Internet Pass-Thru (MQIPT).

### Purpose

Use the **mqiptAdmin** command to issue an administrative command to an active instance of MQIPT.

### Usage notes

The **mqiptAdmin** command connects to the active instance of MQIPT to administer using one of the following methods:

- by making a network connection to a command port
- by connecting to a local instance of MQIPT without using the command port.

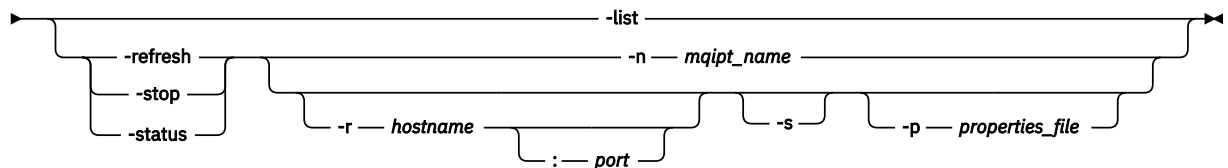
An MQIPT command port might be configured to accept only TLS connections. When connecting to a TLS command port, the **-s** parameter to the **mqiptAdmin** command must be specified.

In order for **mqiptAdmin** to be authorized to connect to a local instance of MQIPT without using the command port, the MQIPT instance must be running on the same system and under the same user ID as **mqiptAdmin**. Alternatively, on AIX and Linux, **mqiptAdmin** can be run as **root**.

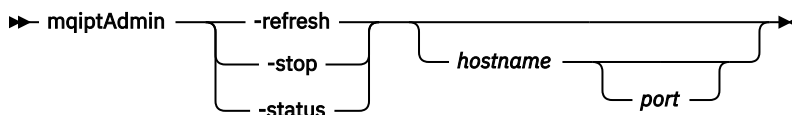
**Note:** The **mqiptAdmin** command is compatible with previous versions of MQIPT, but the command cannot be used to administer versions of MQIPT that are a higher version than the version of the **mqiptAdmin** command. In an environment that includes different versions of MQIPT, use the latest version of the **mqiptAdmin** command.

### Syntax

►► mqiptAdmin →



### Deprecated syntax



### Keywords and parameters

#### -list

Display the names of all local instances of MQIPT that support local administration without the command port.

On AIX and Linux, if **mqiptAdmin** is running as **root**, all local active instances of MQIPT are displayed. Otherwise, only instances of MQIPT that are running under the same user ID as **mqiptAdmin** are displayed.

#### -refresh

Refresh an active instance of MQIPT to bring any configuration changes into effect.

**-stop**

Stops the instance of MQIPT.

MQIPT closes all connections, stops listening for incoming connections, and then exits. The stop command is ignored if the `mqipt.conf` file specifies `RemoteShutDown=false`.

**-status**

Display information about the connection thread pool usage. The information is displayed in the MQIPT console output. This parameter is intended for use by IBM Support.

**-n mqipt\_name**

The name of a local instance of MQIPT to administer. This parameter must be specified to administer a local instance of MQIPT without using a command port.

**-r hostname:port**

The host name and command port of an instance of MQIPT to administer using the command port. This parameter is optional. If neither the `-n` nor `-r` parameters are specified, `mqiptAdmin` connects to `localhost`, port 1881.

**-s**

Secure the connection to MQIPT using TLS. This parameter must be specified to connect to a TLS command port.

**-p properties\_file**

The name of a file containing configuration properties that are necessary to connect to MQIPT using TLS. The name of the properties file can also be specified by setting the `MQS_MQIPTADM_PROP_FILE` environment variable.

For the list of properties that can be specified in the properties file, see [mqiptAdmin properties](#).

**Return codes**

Table 346. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

**Windows****mqiptIcons (create MQIPT Start menu icons)**

Create and remove IBM MQ Internet Pass-Thru (MQIPT) Start menu icons on Windows platforms.

**Purpose**

Use the `mqiptIcons` command to create and remove Start menu icons for MQIPT functions on Windows platforms.

You must run the `mqiptIcons` command as a user with administrator privileges.

**Syntax**

```
➤ mqiptIcons -install -remove installation_name ➤
```

**Parameters****-install**

Create MQIPT icons on the Start menu.

**-remove**

Remove MQIPT icons from the Start menu.

### **installation\_name**

A name that you choose to distinguish this installation of MQIPT from any other. The name is appended to the name of the Start menu folder that is created to contain the MQIPT icons.

## **Return codes**

Table 347. Return code identifiers and descriptions

<b>Return code</b>	<b>Description</b>
0	Command successful.
>0	Command not successful.

## **mqiptPW (encrypt stored password)**

Encrypt a password for use by IBM MQ Internet Pass-Thru (MQIPT).

### **Purpose**

Use the **mqiptPW** command to encrypt a password that is stored for use by MQIPT.

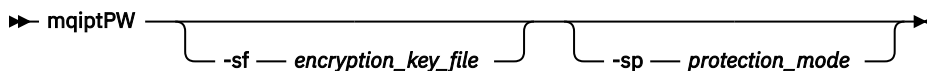
The MQIPT configuration might include passwords to access various resources, as well as the MQIPT access password for administration using the command port.

In versions earlier than IBM MQ 9.1.5, only passwords that are used by MQIPT to access key rings, or cryptographic hardware key stores, can be encrypted. From IBM MQ 9.1.5, all stored passwords for use by MQIPT should be protected by encrypting the password with the **mqiptPW** command.

### **Syntax**

Use this syntax to call the **mqiptPW** command to encrypt any password for use by MQIPT in IBM MQ 9.1.5 or higher. Store the encrypted password in the appropriate property in the `mqipt.conf` configuration file.

The command prompts for the password to be encrypted to be entered.



### **Optional parameters**

#### **-sf encryption\_key\_file**

The name of a file that contains the password encryption key. If specified, the file must contain at least one character, and only one line.

If this parameter is not specified, the default password encryption key is used.

This parameter can be specified only with password protection mode 1 or higher.

#### **-sp protection\_mode**

The password protection mode to be used by the command. One of the following values can be specified:

**V9.3.0 2**

Use the latest password protection mode. This is the default value from IBM MQ 9.3.0.

**1**

Use the IBM MQ 9.1.5 password protection mode for compatibility with versions earlier than IBM MQ 9.3.0. This is the default value in versions earlier than IBM MQ 9.3.0.

**0**

**Deprecated** Use the deprecated password protection mode.

## Deprecated syntax to encrypt key ring passwords

Use this syntax to call the **mqiptPW** command to encrypt a key ring password. The encrypted password is stored in file which can be read by any version of MQIPT. **Deprecated** This syntax is deprecated from IBM MQ 9.1.5 as it does not offer the most secure encryption method.

➤ **mqiptPW** — *password* — *file\_name* — **-replace**

## Parameters for deprecated syntax

**Deprecated**

### *password*

The clear text password to encrypt. Passwords can include the space character, but the whole password string must be enclosed in quotes for this to be acceptable. There is no limit to the length or format of the password.

### *file\_name*

The name of a file to create, to contain the encrypted password.

### **-replace**

Overwrite an existing password file with the same name, if it exists. This parameter is optional.

## Return codes

Table 348. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

## mqiptService (manage the MQIPT service)

Install or uninstall the IBM MQ Internet Pass-Thru (MQIPT) service.

### Purpose

Use the **mqiptService** command to install or uninstall the MQIPT service. The MQIPT service lets you manage and run MQIPT as a Windows service, or as a System V init service on AIX and Linux.

### Usage notes

You can only install one MQIPT service on a system, even if there is more than one installation of MQIPT on the system.

**Note:** Only the installation of MQIPT that installed the service can be used to remove it. For example, if you have two MQIPT installations, one in `/opt/mqipt` and one in `/usr/local/mqipt`, and you run the command `/opt/mqipt/bin/mqiptService -install`, then only the **mqiptService** command from the `/opt/mqipt` installation can subsequently be used to remove the service. Attempting to remove the service using a different installation causes error MQCPE083.

**Linux** **AIX** On AIX and Linux, you must run the **mqiptService** command as `root` to ensure that you have the authority required to configure services.

**Linux** On Linux, the MQIPT service is supported only on operating systems which support System V init. On systems which do not support System V init, use another method, such as `systemd`, to manage MQIPT as a service.

**Windows** On Windows, you must run the **mqiptService** command from an administrator command prompt to ensure that you have the authority required to configure Windows services. The MQIPT service runs under the LocalSystem account. The service cannot be configured to run under a different user ID.

## Syntax on AIX and Linux

Linux AIX

```
mqiptService -install
mqiptService -remove
```

## Syntax on Windows

Windows

```
mqiptService -install mqipt_home
mqiptService -remove
```

## Keywords and parameters

### -install

Install and register the MQIPT service.

Linux AIX

On AIX and Linux, the `mqipt.conf` file for the service must be located in the top-level MQIPT installation directory of the installation from which you ran **mqiptService**.

**Windows** On Windows, you must supply the fully-qualified path to the directory containing the `mqipt.conf` configuration file as a parameter. Enclose the path in double quotation marks (") if it contains spaces.

If the MQIPT configuration contains passwords that have been encrypted using an encryption key other than the default, the password encryption key must be present in a file named `mqipt_cred.key` in the same directory as the `mqipt.conf` file. For more information on encrypting passwords in the MQIPT configuration, see [Encrypting stored passwords](#).

Installing the service does not automatically start it. The service starts the next time the system is restarted. Consult your operating system service documentation if you need to start the MQIPT service immediately, without restarting.

### -remove

Remove the MQIPT service so that it no longer starts at system boot time.

On Windows, the MQIPT service is stopped if it is currently active, and all routes are subject to immediate shutdown.

## Return codes

Table 349. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

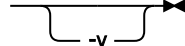
## mqiptVersion (display MQIPT version information)

Display IBM MQ Internet Pass-Thru (MQIPT) version and build information.

## Purpose

Use the **mqiptVersion** command to display MQIPT version and build information.

## Syntax

►► mqiptVersion 

## Optional parameters

**-v**

Display verbose output including build information and the version of the Java runtime environment supplied with MQIPT.

## Return codes

Table 350. Return code identifiers and descriptions

Return code	Description
0	Command successful.
>0	Command not successful.

## Administrative REST API reference

Reference information about the administrative REST API.

For more information about using the administrative REST API, see [Administration using the REST API](#).

For more information about configuring the administrative REST API, see [Configuring the REST API](#).

For more information about securing the administrative REST API, see [Securing the REST API](#).

## REST API resources

This collection of topics provides reference information for each of the administrative REST API resources.

For more information about using the administrative REST API, see [Administration using the REST API](#).

For more information about configuring the administrative REST API, see [Configuring the REST API](#).

For more information about securing the administrative REST API, see [Securing the REST API](#).

### **/admin/action/qmgr/{qmgrName}/mqsc**

You can use the HTTP POST method with the `/admin/action/qmgr/{qmgrName}/mqsc` resource to execute an arbitrary MQSC command on a queue manager.

**Note:** **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

You can use the administrative REST API gateway with this resource URL.

## **POST - plain text MQSC command**

Use the HTTP POST method with this resource to submit administrative commands directly to a queue manager. These administrative commands are submitted in the body of the request, either as a plain text MQSC command, or as a JSON formatted command.

**Note:** **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

You can use the administrative REST API to submit an MQSC command by using either a plain text MQSC command or with a JSON formatted command:

- With a plain text MQSC command, the body of the request contains an MQSC command specified as you would type it on a command line. For example:

```
{
  "type": "runCommand",
  "parameters": {
    "command": "DEFINE CHANNEL(NEWSVRCONN) CHLTYPE(SVRCONN)"
  }
}
```

The response is returned in a plain text format.

- With a JSON formatted command, the body of the request contains an MQSC command in a JSON format. For example:

```
{
  "type": "runCommandJSON",
  "command": "define",
  "qualifier": "channel",
  "name": "NEWSVRCONN",
  "parameters": {
    "chltype": "svrconn"
  }
}
```

The response is returned in JSON format.

For more information about using the JSON formatted MQSC command, see [“POST - JSON formatted command” on page 2166](#).

You can use this REST API command with HTTP to run any MQSC command in plain text format.

On AIX, Linux, and Windows, this REST API command is similar to the [“MQCMD\\_ESCAPE \(Escape\) on Multiplatforms” on page 1153](#) PCF command.

On z/OS, this REST API command is similar to submitting commands directly to the command server:

- Messages are put to a request queue. These messages have MsgType set to MQMT\_REQUEST, Format set to MQFMT\_STRING or MQFMT\_NONE, and the payload set to the text of an MQSC command.
- The command server running in the queue manager reads the messages, validates them, and passes the valid commands to the command processor.
- The command processor then executes the commands, and puts replies to the commands as messages on the reply-to queues that are specified in the incoming messages.
- [“Resource URL” on page 2161](#)
- [“Request headers” on page 2161](#)
- [“Request body format” on page 2161](#)
- [“Security requirements” on page 2162](#)
- [“Response status codes” on page 2162](#)
- [“Response headers” on page 2163](#)
- [“Response body format” on page 2163](#)



- [“Examples” on page 2164](#)

## Resource URL

`https://host:port/ibmmq/rest/v2/admin/action/qmgr/qmgrName/mqsc`

### qmgrName

Specifies the name of the queue manager on which to execute the command.

You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

The queue manager name is case-sensitive.

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.
- A period (.) must be encoded as %2E.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Request headers

The following headers must be sent with the request:

### Content-Type

This header must be sent with a value of `application/json` optionally followed by `; charset=UTF-8`.

### ibm-mq-rest-csrf-token

This header must be set, but the value can be anything, including being blank.

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### ibm-mq-rest-gateway-qmgr

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## Request body format

The request body must be in JSON format in UTF-8 encoding. Within the request body attributes are defined, and named JSON objects are created to specify extra attributes.

The following attributes can be included in the request body:

### type

Required.

String.

Specifies the type of action to be performed.

### runCommand

Specifies that a plain text MQSC command is to be executed

### parameters

Required.

Nested JSON Object.

Specifies the parameters for the action.

This nested object contains only one attribute.

**command**

Required.

A valid plain text MQSC command to be executed.

For more information about the MQSC commands, see [“MQSC commands reference” on page 251](#).

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

The security principal of the caller must be granted the ability to issue MQSC commands against the specified queue manager.

**ALW** On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the **setmqaut** command. For more information, see [setmqaut \(grant or revoke authority\)](#).

**z/OS** On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 200

The specified command was passed successfully to the queue manager for processing.

### 400

Invalid data provided.

For example, an invalid MQSC command is specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. The `ibm-mq-rest-csrf-token` header must also be specified.

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources.
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Queue manager does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

The following headers are returned with the response:

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

### ibm-mq-rest-gateway-qmgr

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

## Response body format

If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

The format of the response body is standardized, with a consistent JSON schema. However, the content is platform-dependent, reflecting the underlying mechanism for executing MQSC commands.

The response body has the following JSON structure:

```
{
  "commandResponse" : [
    {
      "completionCode" : number,
      "reasonCode" : number,
      "text" : [
        "string",
        ...
      ]
    },
    ...
  ]
  "overallCompletionCode" : number,
  "overallReasonCode" : number
}
```

The fields in the response have the following meanings:

### commandResponse

A JSON array of JSON objects that represent individual responses from the execution of the command.

Each response contains the following data:

#### completionCode

The completion code that is associated with the operation.

#### reasonCode

The reason code that is associated with the operation.

#### text

A JSON array of strings that contain the response text that is associated with the operation for this instance. Note that embedded newlines are stripped from this text.

On AIX, Linux, and Windows, this field contains a single string that contains the response from the command, with any newlines escaped in the usual JSON manner.

On z/OS, this field contains multiple entries. For further information, see [Interpreting the reply messages from the command server](#).

### overallCompletionCode

The completion code that is associated with the operation as a whole.

### overallReasonCode

The reason code that is associated with the operation as a whole.

## Examples

**ALW** The following sequence shows how to create a new server-connection channel that is called NEWSVRCONN on AIX, Linux, and Windows queue managers - our example queue manager is called QM\_T1.

- First check that the channel does not exist. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM_T1/mqsc
```

The following JSON payload is sent:

```
{
  "type": "runCommand",
  "parameters": {
    "command": "DISPLAY CHANNEL(NEWSVRCONN)"
  }
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON.

```
{
  "commandResponse": [
    {
      "completionCode": 2,
      "reasonCode": 2085,
      "text": [
        "AMQ8147: IBM MQ object NEWSVRCONN not found."
      ]
    }
  ],
  "overallCompletionCode": 2,
  "overallReasonCode": 3008
}
```

The individual response shows a reason code of 2085 (MQRC\_UNKNOWN\_OBJECT\_NAME) and the MQSC command has an overall reason code of 3008 (MQRCCF\_COMMAND\_FAILED) as it failed to display the requested channel's details.

- Now create the channel. The same URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM_T1/mqsc
```

The following JSON payload is sent:

```
{
  "type": "runCommand",
  "parameters": {
    "command": "DEFINE CHANNEL(NEWSVRCONN) CHLTYPE(SVRCONN)"
  }
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON.

```
{
  "commandResponse": [
    {
      "completionCode": 0,
      "reasonCode": 0,
      "text": [
        "AMQ8014: IBM MQ channel created."
      ]
    }
  ],
  "overallCompletionCode": 0,
  "overallReasonCode": 0
}
```

- Finally, check that the channel does exist. Again the same URL is used with the HTTP POST method:


```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM_T1/mqsc
```

The following JSON payload is sent:

```
{
  "type": "runCommand",
  "parameters": {
    "command": "DISPLAY CHANNEL(NEWSVRCONN) ALL"
  }
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON. The response body is edited for brevity after the CHLTYPE attribute.

```
{
  "commandResponse": [
    {
      "completionCode": 0,
      "reasonCode": 0,
      "text": [
        "AMQ8414: Display Channel details.    CHANNEL(NEWSVRCONN)
CHLTYPE(SVRCONN)"
      ]
    }
  ],
  "overallCompletionCode": 0,
  "overallReasonCode": 0
}
```

 The following sequence shows how to create a new server-connection channel that is called NEWSVRCONN on a z/OS queue manager - our example queue manager is called QM21.

- First check that the channel does not exist. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM21/mqsc
```

The following JSON payload is sent:

```
{
  "type": "runCommand",
  "parameters": {
    "command": "DISPLAY CHANNEL(NEWSVRCONN)"
  }
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON.

```
{
  "commandResponse": [
    {
      "completionCode": 0,
      "reasonCode": 0,
      "text": [
        "CSQN205I  COUNT=          3, RETURN=00000000, REASON=00000000",
        "CSQM297I ]MQ21 CSQMDRTS NO CHANNEL FOUND MATCHING REQUEST CRITERIA ",
        "CSQ9022I ]MQ21 CSQMDRTS ' DISPLAY CHANNEL ' NORMAL COMPLETION "
      ]
    }
  ],
  "overallCompletionCode": 0,
  "overallReasonCode": 0
}
```

The completion and reason codes here are zero, as on z/OS the command is regarded as succeeding, although no matching channel was found.

- Now create the channel. The same URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM21/mqsc
```

The following JSON payload is sent:

```
{
  "type": "runCommand",
  "parameters": {
    "command": "DEFINE CHANNEL(NEWSVRCONN) CHLTYPE(SVRCONN)"
  }
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON.

```
{
  "commandResponse": [
    {
      "completionCode": 0,
      "reasonCode": 0,
      "text": [
        "CSQN205I  COUNT=          2, RETURN=00000000, REASON=00000000",
        "CSQ9022I ]MQ21 CSQMACHL ' DEFINE CHANNEL' NORMAL COMPLETION"
      ]
    }
  ],
  "overallCompletionCode": 0,
  "overallReasonCode": 0
}
```

- Finally, check that the channel does exist. Again the same URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM21/mqsc
```

The following JSON payload is sent:


```
{
  "type": "runCommand",
  "parameters": {
    "command": "DISPLAY CHANNEL(NEWSVRCONN) ALL"
  }
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON. The response body is edited for brevity after the TRPTYPE attribute.

```
{
  "commandResponse": [
    {
      "completionCode": 0,
      "reasonCode": 0,
      "text": [
        "CSQN205I  COUNT=          3, RETURN=00000000, REASON=00000000",
        "CSQM415I ]MQ21 CHANNEL(NEWSVRCONN          ) CHLTYPE(SVRCONN          ) QSGDISP(QMGR          )",
        "DEFCDISP(PRIVATE          ) TRPTYPE(LU62          )",
        "CSQ9022I ]MQ21 CSQMDRTS ' DISPLAY CHANNEL' NORMAL COMPLETION "
      ]
    }
  ],
  "overallCompletionCode": 0,
  "overallReasonCode": 0
}
```

### ***POST - JSON formatted command***

Use the HTTP POST method with this resource to submit administrative commands directly to a queue manager. These administrative commands are submitted in the body of the request, either as a plain text MQSC command, or as a JSON formatted command.

**Note:**  This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

You can use the administrative REST API to submit an MQSC command by using either a plain text MQSC command or with a JSON formatted command:

- With a plain text MQSC command, the body of the request contains an MQSC command specified as you would type it on a command line. For example:

```
{
  "type": "runCommand",
  "parameters": {
    "command": "DEFINE CHANNEL (NEWSVRCONN) CHLTYPE (SVRCONN)"
  }
}
```

The response is returned in a plain text format.

- With a JSON formatted command, the body of the request contains an MQSC command in a JSON format. For example:

```
{
  "type": "runCommandJSON",
  "command": "define",
  "qualifier": "channel",
  "name": "NEWSVRCONN",
  "parameters": {
    "chltype": "svrconn"
  }
}
```

The response is returned in JSON format.

For more information about using the plain text MQSC command, see [“POST - plain text MQSC command” on page 2160](#).

You can use this REST API command with HTTP to run any MQSC command. However, the following MQSC commands are not supported when you use a JSON formatted command in the request body:

- DISPLAY ARCHIVE
- DISPLAY CHINIT
- DISPLAY GROUP
- DISPLAY LOG
- DISPLAY SECURITY
- DISPLAY SYSTEM
- DISPLAY THREAD
- DISPLAY TRACE
- DISPLAY USAGE

On AIX, Linux, and Windows, this REST API command is similar to the [“MQCMD\\_ESCAPE \(Escape\) on Multiplatforms” on page 1153](#) PCF command.

On z/OS, this REST API command is similar to submitting commands directly to the command server:

- Messages are put to a request queue. These messages have MsgType set to MQMT\_REQUEST, Format set to MQFMT\_STRING or MQFMT\_NONE, and the payload set to the text of an MQSC command.
- The command server running in the queue manager reads the messages, validates them, and passes the valid commands to the command processor.
- The command processor then executes the commands, and puts replies to the commands as messages on the reply-to queues that are specified in the incoming messages.
- [“Resource URL” on page 2168](#)
- [“Request headers” on page 2168](#)
- [“Request body format” on page 2168](#)
- [“Security requirements” on page 2171](#)
- [“Response status codes” on page 2172](#)
- [“Response headers” on page 2172](#)

- [“Response body format” on page 2172](#)
- [“Examples” on page 2173](#)

## Resource URL

`https://host:port/ibmmq/rest/v2/admin/action/qmgr/qmgrName/mqsc`

### qmgrName

Specifies the name of the queue manager on which to execute the command.

You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

The queue manager name is case-sensitive.

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.
- A period (.) must be encoded as %2E.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Request headers

The following headers must be sent with the request:

### Content-Type

This header must be sent with a value of `application/json` optionally followed by `; charset=UTF-8`.

### ibm-mq-rest-csrf-token

This header must be set, but the value can be anything, including being blank.

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### ibm-mq-rest-gateway-qmgr

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## Request body format

The request body must be in JSON format in UTF-8 encoding. Within the request body attributes are defined, and named JSON objects are created to specify extra attributes. Any attributes that are not specified use the default value.

The following attributes can be included in the request body:

### type

Required.

String.

Specifies the type of action to be performed.

### runCommandJSON

Specifies that a JSON formatted MQSC command is to be executed



**command**

Required.

String.

Specifies the initial keyword of the MQSC command. The value can be any one of the following values:

- alter
- archive
- backup
- clear
- define
- delete
- display
- move
- ping
- purge
- recover
- refresh
- reset
- resolve
- resume
- rverify
- set
- start
- stop
- suspend

**qualifier**

String.

Specifies the secondary keyword in the MQSC command.

For example, for an **ALTER QLOCAL (qName)** command, the qualifier is **QLOCAL**.

**name**

Optional.

String.

Specifies the primary argument of the MQSC command.

For example, for an **ALTER QLOCAL (qName)** command, the name attribute is qName.

For some commands, this attribute is not required. For example, a **REFRESH SECURITY** command does not require a primary argument.

**responseParameters**

Optional.

String array.

Specifies which parameters are returned in the response to a request where the value of the command attribute is **DISPLAY**.

You can specify a value of [ "all" ] to return all applicable parameters for MQSC commands where the **all** parameter is supported.

**parameters**

Optional.

Nested JSON Object.

Specifies the parameters for the command in name and value pairs.

You can specify the parameters in any order, and in any case. Any double quotation marks or backslash characters used within a value must be escaped:

- A double quotation mark must be represented as \"
- A backslash must be represented as \\

The name and value pairs are constructed based on the following mapping from the MQSC command:

#### **name**

The name part of the name and value pair is the same as the name of the MQSC parameter.

For example, the **TRIGTYPE** parameter on a **DISPLAY QLOCAL** MQSC command maps to **"trigtype"** in the JSON format.

#### **value**

The value part of the name and value pair is the value that is used with the MQSC parameter. The JSON that is used to represent the value depends on the type of the value:

- For an MQSC value that is a string or an enumerated type, then the value used in the JSON format is a JSON string. For example:

```
"chltype" : "SDR",  
"descr" : "A String Description."
```

Unlike using plain-text MQSC, if the string is case-sensitive, or if it contains special characters, you do not need to enclose the string in single quotation marks.

- For an MQSC value that is an integer, then the value that is used in the JSON format is an integer. For example:

```
"maxmsgl" : 50000
```

- For an MQSC parameter that has no associated value, you must specify a value of YES if the attribute applies. For example, for **TRIGGER** on a local queue:

```
"trigger" : "yes"
```

You cannot specify `"trigger" : "no"`. Instead, you must use the attribute **NOTRIGGER**:

```
"nottrigger" : "yes"
```

Similarly, for the attribute **REPLACE**, you must specify the following string:

```
"replace" : "yes"
```

You cannot specify `"replace" : "no"`. To indicate that the MQ object should not be replaced, you must use the attribute **NOREPLACE**:

```
"noreplace" : "yes"
```

- For an MQSC value that is a list, then the value that is used in the JSON format is a JSON array. Each element in the array is a member of the list. A list with no members must be specified as an empty array. For example:

```
"msgexit" : ["exit1", "exit2", "exit3"],  
"rcvexit" : []
```

The following MQSC attributes are lists:

- addrlist
- arcwrtc
- authadd
- authlist

- authrmv
- comphdr
- compmsg
- comprate
- comptime
- connopts
- exclmsg
- exittime
- logs
- msgdata
- msgexit
- names
- nettime
- nid, except on CONN commands
- openopts
- protocol, only on CHANNEL commands
- rcvdata
- rcvexit
- recip
- security, except on REFRESH commands
- senddata
- sendexit
- signer
- suiteb
- userid, only on TRACE commands
- userlist
- xbatchsz
- xqtime

Single quotation marks that are used in the value are automatically escaped. For example, a `descr` attribute with the value *single 'quotation' marks* is represented in the JSON request body as `"descr" : "single 'quotation' marks"`.

For examples of how to format the JSON request, see [“Examples” on page 2173](#)

For more information about the MQSC commands, see [“MQSC commands reference” on page 251](#).

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

The security principal of the caller must be granted the ability to issue MQSC commands against the specified queue manager.

**ALW** On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the **setmqaut** command. For more information, see [setmqaut \(grant or revoke authority\)](#).

**z/OS** On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 200

The specified command was passed successfully to the queue manager for processing.

### 400

Invalid data provided.

For example, an invalid MQSC command is specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. The `ibm-mq-rest-csrf-token` header must also be specified.

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources.
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Queue manager does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

The following headers are returned with the response:

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

### ibm-mq-rest-gateway-qmgr

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

## Response body format

If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

The format of the response body is standardized, with a consistent JSON schema. However, the content is platform-dependent, reflecting the underlying mechanism for executing MQSC commands.

The response body has the following JSON structure:

```
{
  "commandResponse" : [
    {
      "completionCode" : number,
```

```

    "reasonCode" : number,
    "message" : [
      "string",
      ...
    ]
  },
  ...
]
"overallCompletionCode" : number,
"overAllReasonCode" : number
}

```

The fields in the response have the following meanings:

### **commandResponse**

A JSON array of JSON objects that represent individual responses from the execution of the command.

Each response contains the following data:

#### **completionCode**

The completion code that is associated with the operation.

#### **reasonCode**

The reason code that is associated with the operation.

#### **message**

A JSON array of strings that contain any messages that are returned.

#### **parameters**

If an IBM MQ object is returned by the request, this object returns name and value pairs that represent the IBM MQ object. For example, after a **DISPLAY QUEUE** command is sent, a local queue q0 is returned:

```

"parameters": {
  "queue": "q0",
  "type": "QLOCAL",
  "acctq": "QMGR",
  "altdate": "2018-07-16",
  ...
}

```

### **sourceQmgr**

The queue manager from which the response was received.

This object is returned only if the queue manager that the command is issued to is in a queue sharing group and responses are received from other queue managers in the queue sharing group.

### **overallCompletionCode**

The completion code that is associated with the operation as a whole.

### **overallReasonCode**

The reason code that is associated with the operation as a whole.

## **Examples**

- Define a local queue, Q1. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc
```

The following JSON payload is sent:

```

{
  "type": "runCommandJSON",
  "command": "define",
  "qualifier": "qlocal",
  "name": "Q1",
  "parameters": {
    "share": "yes",
    "trigdata": "lowercasetrigdata",
    "trigdpth": 7,
    "usage": "normal"
  }
}

```

```
}  
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON:

**ALW** On AIX, Linux, and Windows:

```
{  
  "commandResponse": [  
    {  
      "completionCode": 0,  
      "message": ["AMQ8006I: IBM MQ queue created."],  
      "reasonCode": 0  
    }  
  ],  
  "overallCompletionCode": 0,  
  "overallReasonCode": 0  
}
```

**z/OS** On z/OS:

```
{  
  "commandResponse": [],  
  "overallCompletionCode": 0,  
  "overallReasonCode": 0  
}
```

- Display the queue. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc
```

The following JSON payload is sent:

```
{  
  "type": "runCommandJSON",  
  "command": "display",  
  "qualifier": "qlocal",  
  "name": "Q1"  
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON:

```
{  
  "commandResponse": [  
    {  
      "completionCode": 0,  
      "parameters": {  
        "acctq": "QMGR",  
        "altdat": "2019-06-06",  
        "alttime": "12.01.21",  
        "boqname": "",  
        "bothresh": 0,  
        "clchname": "",  
        "clusnl": "",  
        "cluster": "xxxx",  
        "clwlprty": 0,  
        "clwlrank": 0,  
        "clwluseq": "QMGR",  
        "share": "YES",  
        "trigtype": "FIRST",  
        "type": "QLOCAL",  
        "usage": "NORMAL"  
      },  
      "reasonCode": 0  
    }  
  ],  
  "overallCompletionCode": 0,  
  "overallReasonCode": 0  
}
```

- Display all the queues on the queue manager, requesting that the `altime` and `trigdpth` parameters are returned. The following URL is used with the HTTP POST method:


```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc
```

The following JSON payload is sent:

```
{
  "type": "runCommandJSON",
  "command": "display",
  "qualifier": "qlocal",
  "name": "*",
  "responseParameters": ["altime", "trigdpth"]
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON:

```
{
  "commandResponse": [
    {
      "completionCode": 0,
      "parameters": {
        "altime": "13.36.31",
        "queue": "Q0",
        "trigdpth": 1,
        "type": "QLOCAL"
      },
      "reasonCode": 0
    },
    {
      "completionCode": 0,
      "parameters": {
        "altime": "13.37.59",
        "queue": "Q1",
        "trigdpth": 7,
        "type": "QLOCAL"
      },
      "reasonCode": 0
    }
  ],
  "overallCompletionCode": 0,
  "overallReasonCode": 0
}
```

-  On z/OS, display the local queue Q0, which is defined on both QMGR1 and QMGR2 in a queue sharing group. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QMGR1/mqsc
```

The following JSON payload is sent:

```
{
  "type": "runCommandJSON",
  "command": "display",
  "qualifier": "qlocal",
  "name": "q0",
  "parameters": {
    "cmdscope": "*"
  }
}
```

A response code of 200 is returned, as the REST command succeeded. The response body that is returned contains the following JSON:

```
{
  "commandResponse": [
    {
      "completionCode": 0,
      "parameters": {
        "acctq": "QMGR",
        "altdat": "2019-01-21",
        "altime": "10.23.43",

```

```

        "boqname": "",
        "bothresh": 0,
        "cfstruct": "",
        "clchname": "",
        "clusnl": "",
        "cluster": "",
        "clwlprty": 0,
        "clwlrank": 0,
        "clwluseq": "QMGR",
        ..
        "trigtype": "FIRST",
        "type": "QLOCAL",
        "usage": "NORMAL"
    },
    "reasonCode": 4,
    "sourceQmgr": "QMGR1"
},
{
    "completionCode": 0,
    "parameters": {
        "acctq": "QMGR",
        "altdat": "2019-03-19",
        "alttime": "13.05.02",
        "boqname": "",
        "bothresh": 0,
        "cfstruct": "",
        "clchname": "",
        "clusnl": "",
        "cluster": "",
        "clwlprty": 0,
        "clwlrank": 0,
        ..
        "trigtype": "FIRST",
        "type": "QLOCAL",
        "usage": "NORMAL"
    },
    "reasonCode": 4,
    "sourceQmgr": "QMGR2"
}
],
"overallCompletionCode": 0,
"overallReasonCode": 0
}

```

## /admin/installation

You can use the HTTP GET method with the `installation` resource to request information about installations.

You cannot use the administrative REST API gateway with this resource URL.

### GET

Use the HTTP GET method with the `installation` resource to request information about the installation that the administrative REST API runs in.

The information that is returned is similar to the information that is returned by the [“dspmqver \(display version information\)”](#) on page 104 control command.

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers”](#) on page 2178
- [Request body format](#)
- [“Security requirements”](#) on page 2178
- [Response status codes](#)
- [“Response headers”](#) on page 2178
- [Response body format](#)
- [Examples](#)



## Resource URL

`https://host:port/ibmmq/rest/v2/admin/installation/{installationName}`

### installationName

Optionally specifies the name of the installation to query. This name must be the name of the installation that the REST API is running in.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Optional query parameters

**attributes={extended}|\*|extended.attributeName,...}**

### extended

Specifies that all extended attributes are returned.

### \*

Specifies all attributes. This parameter is equivalent to **extended**.

### extended.attributeName,...

Specifies a comma-separated list of extended attributes to return:

#### level

String.

IBM MQ build level.

#### operatingSystem

  This attribute is only available on z/OS, AIX, Linux, and Windows.

String.

Full descriptive text of the operating system.

#### description

 This attribute is only available on AIX, Linux, and Windows.

String.

Installation description.

#### installationPath

 This attribute is only available on AIX, Linux, and Windows.

String.

The path to the installation.



#### dataPath

 This attribute is only available on AIX, Linux, and Windows.

String.

The path to where the data for the installation is stored.

#### maximumCommandLevel

  This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

Integer.

Maximum command level that is supported.

#### primary

 This attribute is only available on AIX, Linux, and Windows.

Boolean.

Primary installation status.

## Request headers

The following headers must be sent with the request:

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## Request body format

None.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

There are no specific authorization requirements for an HTTP GET on the `installation` resource.

## Response status codes

### 200

Installation information retrieved successfully.

### 400

Invalid data provided.

For example, invalid installation attributes specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information, see [“Security requirements”](#) on page 2178.

### 404

Installation does not exist.

### 500

Server issue or error code from IBM MQ.

## Response headers

The following headers are returned with the response:

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

## Response body format

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `installation`. Each element in the array is a JSON object that represents information about an installation. Each JSON object contains the following attributes:

### name

 This attribute is only available on AIX, Linux, and Windows.

String.

The installation name.

**version**

String.

The version of IBM MQ for the installation.

**platform**

String.

One of the following values:

- appliance
- ibm-i
- unix
- windows
- z/os

**extended**

JSON object.


If requested, contains one or more of the following extra properties:

**level**

String.

IBM MQ build level.


**operatingSystem**

 This attribute is only available on z/OS, AIX, Linux, and Windows.

String.

Full descriptive text of the operating system.


**description**

 This attribute is only available on AIX, Linux, and Windows.

String.

Installation description.


**installationPath**

 This attribute is only available on AIX, Linux, and Windows.

String.

The path to the installation.


**dataPath**

 This attribute is only available on AIX, Linux, and Windows.

String.

The path to where the data for the installation is stored.


**maximumCommandLevel**

 This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

Integer.

Maximum command level that is supported.

**primary**

 This attribute is only available on AIX, Linux, and Windows.

Boolean.

Primary installation status.

If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

## Examples for AIX, Linux, and Windows

### ALW

- The following example gets basic information about the installation that the REST API is running in. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/installation
```

The following JSON response is returned:

```
{
  "installation":
  [
    {
      "name": "Installation1",
      "platform": "windows",
      "version": "9.1.0.0"
    }
  ]
}
```

- The following example gets extended information about the installation Installation1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/installation/Installation1?attributes=*
```

The following JSON response is returned:

```
{
  "installation":
  [
    {
      "extended": {
        "dataPath": "C:\\Program Files (x86)\\IBM\\WebSphere MQ",
        "description": "My MQ installation",
        "installationPath": "C:\\Program Files\\IBM\\WebSphere MQ",
        "level": "p910-L180501",
        "maximumCommandLevel": 910,
        "operatingSystem": "Windows 7 Professional x64 Edition, Build 7601: SP1",
        "primary": true
      },
      "name": "Installation1",
      "platform": "windows",
      "version": "9.1.0.0"
    }
  ]
}
```

- The following example gets the installation path for Installation1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/installation/Installation1?
attributes=extended.installationPath
```

The following JSON response is returned:

```
{
  "installation": [
    {
      "extended": {
        "installationPath": "C:\\Program Files\\IBM\\MQ"
      },
      "name": "Installation1",
      "platform": "windows",
      "version": "9.1.0.0"
    }
  ]
}
```

## Examples for z/OS

### z/OS

- The following example gets basic information about the installation. The following URL is used with the HTTP GET method:

```
https://REST.example.com:9443/ibmmq/rest/v2/admin/installation
```

The following JSON response is returned:

```
{
  "installation": [{
    "platform": "z/os",
    "version": "9.1.0"
  }]
}
```

- The following example gets extended information about the installation. The following URL is used with the HTTP GET method:

```
https://REST.example.com:9443/ibmmq/rest/v2/admin/installation?attributes=extended
```

The following JSON response is returned:

```
{
  "installation": [{
    "extended": {
      "level": "V910-L180501",
      "operatingSystem": "z/OS 01.00 02"
    },
    "platform": "z/os",
    "version": "9.1.0"
  }]
}
```

## /login

You can use the HTTP GET method together with the `login` resource to get information about the user that is logged in to the REST API. You can use the HTTP POST method to log in a user and get an LTPA token. You can use the HTTP DELETE method to log out a user and end the session.

### **POST**

Use the HTTP POST method with the `login` resource to log in a user and start a token-based authentication session for the REST API. An LTPA token is returned for the user to authenticate further REST requests.

For more information about how to use token based authentication, see [Using token based authentication with the REST API](#).

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2182](#)
- [Request body format](#)
- [Response status codes](#)
- [“Response headers” on page 2182](#)
- [Response body format](#)
- [Examples](#)

### **Resource URL**

```
https://host:port/ibmmq/rest/v2/login
```

### **Optional query parameters**

None.

## Request headers

The following headers must be sent with the request:

### Content-Type

This header must be sent with a value of `application/json` optionally followed by  `; charset=UTF-8`.

## Request body format

The request body must be in JSON format in UTF-8 encoding. Within the request body attributes are defined. The following attributes can be included in the request body:

### username

String.

Specifies the user name to authenticate with.

The user name that is specified must be defined within the mqweb server user registry, and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. This user name is case sensitive.

**Note:** If the user name specified has the MQWebUser role, ensure that the user name has the same case in the user registry as on the IBM MQ system. For example, if the user ID is defined on the IBM MQ system in uppercase, it must be defined in the registry in uppercase. If the user name is specified in different cases, the user might be authenticated by the REST API, but might not be authorized to use IBM MQ resources.

### password

String.

Specifies the password of the user that is specified by the **username** attribute.

## Response status codes

### 204

User logged in successfully.

### 400

Invalid data provided.

For example, an integer value is specified for the user name.

### 401

Not authenticated.

An invalid user name or password was provided.

### 500

Server issue or error code from IBM MQ.

## Response headers

None.

## Response body format

The response body is empty if the login is successful. If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

An LTPA security token is returned in a cookie with a successful login. This token is used to authenticate all further REST requests. By default on z/OS, AIX, Linux, and Windows, the cookie name starts with the prefix `LtpaToken2`, but the name can be changed by setting the **ltpaCookieName** property with the **setmqweb** command. For more information, see [Configuring the LTPA token](#). On the IBM MQ Appliance, the LTPA token cookie name is `LtpaToken2`.

## Examples

The following example logs in a user called mqadmin with the password mqadmin. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/login
```

The following JSON payload is sent:

```
{
  "username" : "mqadmin",
  "password" : "mqadmin"
}
```

In cURL, the log in request might look like the following Windows example. The LTPA token is stored in the cookiejar.txt file by using the -c flag:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data
"{\"username\": \"mqadmin\", \"password\": \"mqadmin\"}"
-c c:\cookiejar.txt
```

After the user is logged in, the LTPA token and ibm-mq-rest-csrf-token HTTP header are used to authenticate further requests. For example, to create a local queue, Q1, the following cURL might be used. The LTPA token is retrieved from the cookiejar.txt file by using the -b flag. The contents of the ibm-mq-rest-csrf-token HTTP header can be anything including blank.

```
curl -k "https://localhost:9443/ibmmq/rest/v2/admin/qmgr/QM1/queue" -X POST
-b c:\cookiejar.txt
-H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/json"
--data "{\"name\": \"Q1\"}"
```

## GET

Use the HTTP GET method with the login resource to request information about the user that is authenticated with the REST API.

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2183](#)
- [Request body format](#)
- [“Security requirements” on page 2184](#)
- [Response status codes](#)
- [“Response headers” on page 2184](#)
- [Response body format](#)
- [Examples](#)

## Resource URL

```
https://host:port/ibmmq/rest/v2/login
```

## Optional query parameters

None.

## Request headers

The following headers must be sent with the request:

## Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## Request body format

None.

## Security requirements

The request must be authenticated by using one of the following authentication mechanisms:

- For HTTP basic authentication, you must provide the user name and password to authenticate. For more information, see [Using HTTP basic authentication with the REST API](#).
- For token based authentication, you must provide the LTPA token to authenticate. For more information, see [Using token based authentication with the REST API](#).
- For client certificate authentication, you must provide the client certificate to authenticate. For more information, see [Using client certificate authentication with the REST API](#).

## Response status codes

### 200

User queried successfully.

### 400

Invalid data provided.

### 401

Not authenticated.

An invalid credential was provided.

### 404

Resource was not found.

### 500

Server issue or error code from IBM MQ.

## Response headers

The following headers are returned with the response:

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

## Response body format

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `user`. This array contains the following attributes:

### name

String.

Specifies the name of the user that is used to check for authorization.

This name might be different from the credentials that are specified using, for example, LDAP user mapping or client certificate user mapping.

### role

JSON array.

Specifies which roles the user is granted.

The value is one or more of the following values:

- `MQWebAdmin`



- MQWebAdminRO
- MQWebUser

## Examples

The following example queries the user. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/login
```

The following JSON response is returned:

```
{
  "user" :
  [ {
    "name" : "reader",
    "role" : [
      "MQWebAdminRO",
      "MQWebUser"
    ]
  } ]
}
```

In cURL, the log in query might look like the following Windows example that uses token based authentication. The LTPA token is retrieved from the `cookiejar.txt` file by using the `-b` flag:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X GET
-b c:\cookiejar.txt
```

## DELETE

Use the HTTP DELETE method with the `login` resource to log out a user and end a token-based authentication session for the REST API.

For more information about how to use token based authentication, see [Using token based authentication with the REST API](#).

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2185](#)
- [Request body format](#)
- [“Security requirements” on page 2186](#)
- [Response status codes](#)
- [“Response headers” on page 2186](#)
- [Response body format](#)
- [Examples](#)

## Resource URL

```
https://host:port/ibmmq/rest/v2/login
```

## Optional query parameters

None.

## Request headers

The following headers must be sent with the request:

### **ibm-mq-rest-csrf-token**

This header must be set, but the value can be anything, including being blank.

## Request body format

None.

## Security requirements

The LTPA token that is used to authenticate the user must be provided with the request as a cookie. By default, this token starts with the prefix `LtpaToken2`.

With the response to the REST request, an instruction to delete the LTPA token from the local cookie store is included. Ensure that you process this instruction. If the instruction is not processed, and the LTPA token remains in the local cookie store, then the LTPA token can be used to authenticate future REST requests. That is, when the user attempts to authenticate with the LTPA token after the session is ended, a new session is created that uses the existing token.

## Response status codes

### 204

User logged out successfully.

### 400

Invalid data provided.

### 401

Not authenticated.

An invalid LTPA token was provided, or the `ibm-mq-rest-csrf-token` header was missing.

### 404

Resource was not found.

### 500

Server issue or error code from IBM MQ.

## Response headers

None.

## Response body format

The response body is empty if the logout is successful. If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

## Examples

The following cURL example for Windows logs out a user.

The LTPA token is retrieved from the `cookiejar.txt` file by using the `-b` flag. CSRF protection is provided by the presence of the `ibm-mq-rest-csrf-token` HTTP header. The location of the `cookiejar.txt` file is specified by the `-c` flag so that the LTPA token is deleted from the file:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X DELETE
-H "ibm-mq-rest-csrf-token: value" -b c:\cookiejar.txt
-c c:\cookiejar.txt
```

## /admin/qmgr

You can use the HTTP GET method with the `qmgr` resource to request information about queue managers, including status information.

**Note:** **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

You can use the administrative REST API gateway with this resource URL.

For more information about the PCF equivalents to the queue manager REST API parameters and attributes, see [“REST API and PCF equivalents for queue managers” on page 2379](#).

## GET

Use the HTTP GET method with the `qmgr` resource to request basic information and status information about queue managers.

**Note:** **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

The information that is returned is similar to the information that is returned by the [“dspmq \(display queue managers\)” on page 73](#) control command, the **DISPLAY QMSTATUS** MQSC command, and the **Inquire Queue Manager Status** PCF command. You can also obtain status for high availability (HA) and disaster recovery (DR) configurations on the IBM MQ Appliance as returned by the **dspmq -o HA** and **dspmq -o DR** commands. For more information, see [dspmq \(display queue managers\)](#) in the IBM MQ Appliance documentation.

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2189](#)
- [Request body format](#)
- [“Security requirements” on page 2189](#)
- [Response status codes](#)
- [“Response headers” on page 2190](#)
- [Response body format](#)
- [Examples](#)

## Resource URL

`https://host:port/ibmmq/rest/v2/admin/qmgr/{qmgrName}`

### qmgrName

Optionally specifies the name of the queue manager to query.

You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

If you specify a remote queue manager, only the following attributes are returned:

- name
- started
- channelInitiatorState
- ldapConnectionState
- connectionCount
- publishSubscribeState

The queue manager name is case-sensitive.



If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.
- A period (.) must be encoded as %2E.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Optional query parameters

**attributes={extended|\*|extended.attributeName,...}**

  This parameter is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

This parameter is not valid if you specify a remote queue manager in the resource URL.

### **extended**

Specifies that all extended attributes are retrieved.

**\***

Specifies all attributes. This parameter is equivalent to **extended**.

### **extended.attributeName,...**

Specifies a comma-separated list of extended attributes to return.

For example, to return the `installationName` attribute, specify `extended.installationName`.

For a full list of extended attributes, see [Extended attributes for queue managers](#).

**status={status|\*|status.attributeName,...}**

### **status**

Specifies that all status attributes are returned.

**\***

Specifies all attributes. This parameter is equivalent to **status**.

### **status.attributeName,...**

Specifies a comma-separated list of queue manager status attributes to return.

The queue manager must be running to return the status attributes.

For example, to return the `connectionCount` attribute, specify `status.connectionCount`.

For a full list of status attributes, see [Status attributes for queue managers](#).

**state=state**

Specifies that only queue managers with the specified state are returned. The following values are valid values:

On all platforms:

- running
- ended

 On AIX, Linux, and Windows:

- endedImmediately
- endedPreemptively
- endedUnexpectedly
- starting
- quiescing
- endingImmediately
- endingPreemptively
- beingDeleted
- stateNotAvailable
- runningAsStandby
- runningElsewhere

You can specify the `state=state` optional query parameter only if you do not specify a queue manager name within the resource URL. That is, you cannot request information about a specific queue manager in a specific state.

V 9.3.0 MQ Appliance **ha={\*|ha|attributeName,...}**

Specifies that HA information is returned for HA queue managers. You can specify that all HA attributes are returned (\* or ha), or specify particular attributes (one or more of `ha.type`, `ha.floatingIPAddress`, and `ha.floatingIPInterface`). You can combine DR attributes with any other attributes of the `qmgr` resource.

If you specify this attribute on a platform other than IBM MQ Appliance, the response is bad request 400.

V 9.3.0 MQ Appliance **dr={\*|dr|attributeName,...}**

Specifies that DR information is returned for DR queue managers. You can specify that all DR attributes are returned (\* or dr), or specify particular attributes (`dr.replicationPort` or `dr.remoteIPAddress`). You can combine DR attributes with any other attributes of the `qmgr` resource.

If you specify this attribute on a platform other than IBM MQ Appliance, the response is bad request 400.

## Request headers

The following headers must be sent with the request:

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### ibm-mq-rest-gateway-qmgr

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## Request body format

None.

## Security requirements

The caller must be authenticated to the `mqweb` server and must be a member of one or more of the `MQWebAdmin`, `MQWebAdminRO`, or `MQWebUser` roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

When the **status** optional query parameter is specified, the ability to issue certain PCF commands is required. If only a subset of the status attributes is to be returned, only the permissions for the corresponding PCF commands are required. The security principal of the caller must be granted the ability to issue the following PCF commands for the specified queue manager:

- MQ Appliance ALW On the IBM MQ Appliance, AIX, Linux, and Windows:
  - To return the `started`, `channelInitiatorState`, `ldapConnectionState`, or `connectionCount` attributes, authority to issue the **MQCMD\_INQUIRE\_Q\_MGR\_STATUS** PCF command must be granted.

- To return the `publishSubscribeState` attribute, authority to issue the **MQCMD\_INQUIRE\_PUBSUB\_STATUS** PCF command must be granted.

▶ **z/OS** On z/OS:

- To return the `started` attribute, authority to issue the **MQCMD\_INQUIRE\_LOG** PCF command must be granted.
- To return the `channelInitiatorState` attribute, authority to issue the **MQCMD\_INQUIRE\_CHANNEL\_INIT** PCF command must be granted.
- To return the `connectionCount` attribute, authority to issue the **MQCMD\_INQUIRE\_CONNECTION** PCF command must be granted.
- To return the `publishSubscribeState` attribute, authority to issue the **MQCMD\_INQUIRE\_PUBSUB\_STATUS** PCF command must be granted.

▶ **ALW** On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the **setmqaut** command. For more information, see [setmqaut \(grant or revoke authority\)](#).

▶ **z/OS** On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 200

Queue manager information retrieved successfully.

### 400

Invalid data provided.

For example, invalid queue manager specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the `MQWebAdmin`, `MQWebAdminRO`, or `MQWebUser` roles. For more information, see [“Security requirements” on page 2189](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources. For more information about the access that is required, see [“Security requirements” on page 2189](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Queue manager does not exist.

### 500

Server issue or error code from IBM MQ.

## Response headers

The following headers are returned with the response:

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

### ibm-mq-rest-gateway-qmgr

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

## Response body format

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `qmgr`. Each element in the array is a JSON object that represents information about a queue manager. Each JSON object contains the following attributes:

### name

String.

The queue manager name.

### state

String.

This attribute is not returned if the queue manager that is specified in the resource URL is a remote queue manager.

One of the following values:

On all platforms:

- `running`
- `ended`

 **ALW** On AIX, Linux, and Windows:


- `endedImmediately`
- `endedPreemptively`
- `endedUnexpectedly`
- `starting`
- `quiescing`
- `endingImmediately`
- `endingPreemptively`
- `beingDeleted`
- `stateNotAvailable`
- `runningAsStandby`
- `runningElsewhere`

The following objects can be included in the JSON object that represents information about a queue manager. Which objects and attributes are returned depends on the URL that was specified for the request:

### status

Contains attributes that are related to status information for the queue manager.



### extended

 **ALW** These attributes are only available on the IBM MQ Appliance, AIX, Linux, and Windows.

These attributes are not returned if the queue manager that is specified in the resource URL is a remote queue manager.



Contains extended attributes.

### ha

  These attributes are only available on the IBM MQ Appliance.

Contains high availability attributes.

### dr

  These attributes are only available on the IBM MQ Appliance.

Contains disaster recovery attributes.

For more information, see [“Response body attributes for queue managers”](#) on page 2194.

If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

## Examples for AIX, Linux, and Windows



- The following example gets basic information about all queue managers. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/qmgr
```

The following JSON response is returned:

```
{
  "qmgr": [
    {
      "name": "QM_T1",
      "state": "endedImmediately"
    },
    {
      "name": "RESTQM0",
      "state": "endedUnexpectedly"
    }
  ]
}
```

- The following example gets extended information about the queue manager QM\_T1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/qmgr/QM_T1?attributes=extended
```

The following JSON response is returned:

```
{
  "qmgr": [
    {
      "extended": {
        "installationName": "Installation1",
        "isDefaultQmgr": false,
        "permitStandby": "notApplicable"
      },
      "name": "QM_T1",
      "state": "endedImmediately"
    }
  ]
}
```

- The following example gets specific information about all queue managers. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/qmgr?attributes=extended.permitStandby
```

The following JSON response is returned:

```
{
  "qmgr": [
    {
      "extended": {
        "permitStandby": "notApplicable"
      },
      "name": "QM_T1",
      "state": "endedImmediately"
    },
    {
      "extended": {
        "permitStandby": "notApplicable"
      },
      "name": "RESTQM0",
      "state": "endedUnexpectedly"
    }
  ]
}
```

- The following example gets status for the queue manager QM1. The following URL is used with the HTTP GET method:



```
http://localhost:9443/ibmmq/rest/v2/admin/qmgr/QM1?status=*
```

The following JSON response is returned:

```
{
  "qmgr":
  [ {
    "name": "QM1",
    "state": "running",
    "status":
    {
      "started": "2016-11-08T11:02:29.000Z",
      "channelInitiatorState": "running",
      "ldapConnectionState": "disconnected",
      "connectionCount": 23,
      "publishSubscribeState": "running"
    }
  }
]
```

## Examples for IBM MQ Appliance

V 9.3.0 MQ Appliance

- The following example gets HA information from all queue managers on the appliance. The following URL is used with the HTTP GET method:

```
https://appliance.example.com:5554/ibmmq/rest/v2/admin/qmgr/?ha=*
```

In the example configuration, there is one HA queue manager named HAQM1. The following JSON response is returned:

```
{
  "qmgr": [
    {
      "name": "HAQM1",
      "ha": {
        "floatingIPAddress": "172.20.37.16",
        "floatingIPInterface": "eth22",
        "type": "replicated"
      },
      "state": "endedImmediately"
    }
  ]
}
```

- The following example gets DR information from all queue managers on the appliance. The following URL is used with the HTTP GET method:

```
https://appliance.example.com:5554/ibmmq/rest/v2/admin/qmgr/?dr=*
```

In the example configuration, there is one DR queue manager named DRQM1. The following JSON response is returned:

```
{
  "qmgr": [
    {
      "name": "DRQM1",
      "state": "endedUnexpectedly",
      "dr": {
        "remoteIPAddress": [
          "172.20.39.0"
        ],
        "replicationPort": 1419
      }
    }
  ]
}
```

- The following example gets extended information about the queue manager QM1. The following URL is used with the HTTP GET method:

```
https://appliance.example.com:5554/ibmmq/rest/v2/admin/qmgr/QM1?attributes=extended
```

The following JSON response is returned:

```
{
  "qmgr": [
    {
      "extended": {
        "installationName": "MQAppliance",
        "isDefaultQmgr": false,
        "encryptedFileSystem": "yes"
      },
      "name": "QM1",
      "state": "endedImmediately"
    }
  ]
}
```

## Examples for z/OS

### z/OS

- The following example gets basic information about all queue managers. The following URL is used with the HTTP GET method:

```
https://REST.example.com:9443/ibmmq/rest/v2/admin/qmgr
```



The following JSON response is returned:

```
{
  "qmgr": [
    {
      "name": "MQ5B",
      "state": "ended"
    }
  ]
}
```

### *Response body attributes for queue managers*

When you use the HTTP GET method with the `qmgr` object to request information about queue managers, the following attributes are returned within named JSON objects.

The following objects are available:

- [“status” on page 2194](#)
- [“extended” on page 2196](#)
-  [“ha” on page 2196](#)
-  [“dr” on page 2196](#)

For more information about the PCF equivalents to the queue manager REST API parameters and attributes, see [“REST API and PCF equivalents for queue managers” on page 2379](#).

### **status**

The `status` object contains status information about queue managers:

#### **started**

String.

Specifies the date and time at which the queue manager was started.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).


#### **channelInitiatorState**

String.


Specifies the current state of the channel initiator.

On all platforms, the value is one of the following values:

- stopped
- running

 On the IBM MQ Appliance, AIX, Linux, and Windows, the value can also be one of the following values:


- starting
- stopping

 On z/OS, the value can also be one of the following values:

- unknown

This value indicates that the channel initiator did not return a response to the status request. The channel initiator might be running, but busy. Retry the request after a short time to resolve the issue.

### **ldapConnectionState**

 This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

String.

Specifies the current state of the connection to the LDAP server.

The value is one of the following values:

- connected
- error
- disconnected

### **connectionCount**

Integer.

Specifies the current number of connections to the queue manager.

On z/OS, this attribute includes threads that might be disassociated from a connection, together with connections that are in-doubt and connections where external intervention is required.

### **publishSubscribeState**

String.

Specifies the current state of the publish/subscribe engine of the queue manager.

The value is one of the following values:

#### **stopped**

Specifies that the publish/subscribe engine, and the queued publish/subscribe interface is not running.

#### **starting**

Specifies that the publish/subscribe engine is initializing.

#### **running**

Specifies that the publish/subscribe engine, and the queued publish/subscribe interface are running.

#### **compatibility**

Specifies that the publish/subscribe engine is running, but that the publish/subscribe interface is not running. Therefore, it is possible to publish or subscribe by using the application programming interface. However, any message that is put to the queues that are monitored by the queued publish/subscribe interface are not acted upon.



#### **error**

The publish/subscribe engine failed.

#### **stopping**

The publish/subscribe engine is stopping.

## extended

  This object is only available on the IBM MQ Appliance, AIX, Linux, and Windows. This object is not returned if the queue manager that is specified in the resource URL is a remote queue manager. The extended object contains extended information about queue managers:

### isDefaultQmgr

Boolean.

Specifies whether the queue manager is the default queue manager.

The value is `true` if the queue manager is the default queue manager.

### permitStandby

 This attribute is only available on AIX, Linux, and Windows.

String.

Specifies the permissible standby state.

The value can be one of the following values:

- `permitted`
- `notPermitted`
- `notApplicable`

### installationName

String.

Specifies the name of the installation that the queue manager is associated with.

### **encryptedFileSystem**

This attribute is only available on IBM MQ Appliance.

String.

Set to `yes` if the queue manager file system is encrypted, or `no` if the file system is not encrypted.

## ha

This object is only available on the IBM MQ Appliance. The `ha` object returns information about the high availability (HA) configuration on the appliance:

### type

String.

Specifies whether the queue manager is configured for HA. Is set to `"replicated"` for an HA queue manager, or is an empty string otherwise.

### floatingIPAddress

String.

Specifies the floating IP address if one has been configured for an HA queue manager.

### floatingIPInterface

String.

Specifies the local interface that is used to connect to the queue manager on the two appliances in the HA group, if one has been configured for an HA queue manager.

## dr

This object is only available on the IBM MQ Appliance. The `dr` object returns information about the disaster recovery (DR) configuration on the appliance:

**replicationPort**

Integer.

Specifies the port used by the data replication listener. Contains zero if DR is not configured.

**remoteIPAddress**

String list.

The list can contain up to two elements, zero if DR is not configured, one if DR is configured to a single remote appliance or to a HA pair with a DR floating IP address, or two if DR is configured to a HA pair without using a DR floating IP address.

## **/admin/mft/agent**

You can use the HTTP GET method with the agent resource, to request information about Managed File Transfer agents.

**Note:** **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

**Related tasks**

[Getting started with the REST API for MFT](#)

**Related reference**

[“/admin/mft/transfer” on page 2252](#)

You can use the HTTP GET method with the transfer resource to request information about Managed File Transfer transfers. You can use the HTTP POST method to put a transfer request message to the command queue manager, which will be routed to the source agent queue manager.

**GET**

Use the HTTP GET method with the agent resource to request information about Managed File Transfer agents.

**Note:** **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

The information that is returned is similar to the information returned by the [“fteListAgents \(list the MFT agents for a coordination queue manager\)” on page 2085](#) and [“fteShowAgentDetails \(display MFT agent details\)” on page 2129](#) commands.

For more information about configuring the MFT REST API, see [Configuring the REST API for MFT](#).

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2199](#)
- [Request body format](#)
- [“Security requirements” on page 2200](#)
- [Response status codes](#)
- [“Response headers” on page 2200](#)
- [Response body format](#)
- [Examples](#)

**Resource URL**

`https://host:port/ibmmq/rest/v2/admin/mft/agent/{agentname}`

**agentName**

Optionally specifies the name of the agent to query.

The agent name is not case-sensitive, but agent names that are entered in lowercase or mixed case are converted to uppercase. The agent name value that is received as a response from the REST API is always in uppercase.

The agent name can contain a maximum of 28 characters, and must conform to the IBM MQ [rules for naming objects](#). In addition to the IBM MQ object naming conventions, the percent (%) character cannot be used in agent names.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring the HTTP and HTTPS ports](#).

**Optional query parameters**

**attributes={*object*,...|\*|*object.attributeName*,...}**

**object**

Specifies a comma-separated list of JSON objects that are added to a JSON object, which is a subsection of the complete details.

For example to return:

- All general details of all agents or a particular agent, specify *general*.
- All queue manager connection details of all agents or a particular agent specify *qmgrConnection*.
- Details of connect direct bridge agent, specify *connectDirectBridge*. (applicable only for agent of type "connect direct bridge")
- Details of protocol agent, specify *protocolBridge*. (applicable only for agents of type "protocol bridge")

For a full list of attributes see [“Response body attributes for agents” on page 2203](#)

**\***

Specifies all attributes.

**object.attributeName,...**

Specifies a comma-separated list of agent attributes to return.

Each attribute must specify the JSON object that contains the attribute, in the form *object.attributeName*. For example, to return the *statusAge* attribute, which is contained in the *general* object, specify *general.statusAge*.

You cannot specify the same attribute more than once. If you request attributes that are not valid for a particular agent, the attributes are not returned for that agent.

**name=name**

This parameter cannot be used if you specify an agent name in the resource URL. Specifies a wildcard agent name to filter on.

The name specified must include an \* as a wildcard character. You can specify one of the following combinations:

**\***

Specifies that all agents are returned.

**prefix\***

Specifies that all agents with the specified prefix in the agent name are returned.

**suffix\***

Specifies that all agents with the specified suffix in the agent name are returned.

**prefix\*suffix**

Specifies that all agents with the specified prefix and specified suffix in the agent name are returned.

**type=validAgentType**

Specifies the type of agent to return information about. The value can be one of the following values:

**all**

Specifies that information about all agents is returned. `standard`, `connectDirectBridge`, and `protocolBridge` agent information is returned.

This is the default value.

**standard**

Specifies that information about agent of type `standard` is returned.

**connectDirectBridge**

Specifies that information about agents of type `connect direct bridge` is returned.

**protocolBridge**

Specifies that information about agents of type `protocol bridge` is returned.

**state=validAgentState**

Specifies the state of the agent to return information about. The value can be one of the following values:

**all**

Specifies that information about all agents is returned. This information includes all the valid states listed in the following text.

This is the default value.

**active**

Specifies that information about agents that are in an active state is returned.

**ready**

Specifies that information about agents that are in a ready state is returned.

**starting**

Specifies that information about agents that are in a starting state is returned.

**unreachable**

Specifies that information about agents that are in an unreachable state is returned.

**stopped**

Specifies that information about agents that are in a stopped state is returned.

 **stopping**

Specifies that information about agents that are in a stopping state is returned.

**endedUnexpectedly**

Specifies that information about agents that are in an endedUnexpectedly state is returned.

**noInformation**

Specifies that information about agents that are in a noInformation state is returned.

**unknown**

Specifies that information about agents that are in an unknown state is returned.

**problem**

Specifies that information about agents that are in a problem state is returned.

**Request headers**

The following header must be sent with the request:

**Authorization**

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

**Request body format**

None.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MFTWebAdmin, MFTWebAdminRO or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

The security principal of the caller must be granted authority to subscribe to the SYSTEM.FTE/Agents topic.

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

## Response status codes

### 200

Agent information retrieved successfully.

### 400

Invalid data provided.

For example, invalid agent attributes specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server. For more information, see [“Security requirements” on page 2200](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal is not a member of one or more of the MFTWebAdmin, MFTWebAdminRO, or MQWebUser roles. For more information about the access that is required, see [“Security requirements” on page 2200](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Agent does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

## Response body format

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `agent`. Each element in the array is a JSON object that represents information about an agent. Each of these JSON objects contains the following attributes:

### name

String.

Specifies the name of the agent.

This attribute is always returned.



**type**

String.



Specifies the type of agent.

The value is one of the following values:

- standard
- connectDirectBridge
- protocolBridge

**state**

Specifies the state of the agent. The value can be one of the following values:

- active
- ready
- starting
- unreachable
-   stopping
- stopped

**general**

Contains attributes that are related to general agent properties, such as the agent description, the agent age, and the version and level of the queue manager.

**queueManagerConnection**

This object provides information about the queue manager connections, such as the queue manager name, and transport type.

**connectDirectBridge**

This object provides information about to connect direct bridge type agent, such as the node name, host, and port.

**protocolBridge**

This object provides information about protocol bridge type agent, such as the endpoints and default server.

**standbyInstance**

This object provides information about the status of the standby instances

For more information, see [“ Response body attributes for transfers” on page 2271.](#)

If an error occurs, see [REST API error handling.](#)

**Examples**

The following example returns the basic details of all agents, that is, only the following information is displayed:

- agent name
- agent type
- agent state

The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/agent/
```

The following JSON response is returned:

```
{
  "agent": [{
    "name": "AGENT1",
    "state": "ready",
```

```

    "type": "standard"
  },
  {
    "name": "AGENT2",
    "state": "ready",
    "type": "standard"
  },
  {
    "name": "BRIDGE_AGENT3",
    "type": "protocolBridge",
    "state": "ready"
  },
  {
    "name": "CD_AGENT",
    "type": "connectDirectBridge",
    "state": "ready"
  }
}
}]

```

The following example lists all the agent of type **standard**, along with the **general** object. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/agent?attributes=general&type=standard
```

The following JSON response is returned:

```

{
  "agent": [{
    "name": "SRC",
    "state": "ready",
    "type": "standard",
    "general": {
      "description": "Standard connected to the qmgr in client mode",
      "statusAge": "06:31:00",
      "version": "9.1.5.0",
      "level": "p915-L190514",
      "statusPublicationRate": 300,
      "statusPublishTime": "2019-05-14T06:57:07.000Z",
      "maximumQueuedTransfers": 1000,
      "maximumDestinationTransfers": 25,
      "maximumSourceTransfers": 25,
      "operatingSystem": "Windows10"
    },
    "standbyInstance": [
      {
        "host": "MFTHA1",
        "version": "9.1.5.0"
      },
      {
        "host": "9.122.123.124",
        "version": "9.1.5.0"
      }
    ]
  }]
}

```

Note that the `standbyInstance` attributes are displayed only if the agent is enabled as highly available.

The following example lists all the agents starting with the name AGENT, in a **ready** state, and of type **standard**, along with the **general** object of `statusAge`. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/agent?name=AGENT*&state=ready&type=standard&attributes=general.statusAge
```

The following JSON response is returned:

```

{
  "agent": [{
    "name": "AGENT1",
    "state": "ready",
    "type": "standard",
    "general": {
      "statusAge": "05:00:00"
    }
  }]
}

```

```

    },
    {
      "name": "AGENT2",
      "state": "ready",
      "type": "standard",
      "general": {
        "statusAge": "03:00:00"
      }
    },
    {
      "name": "AGENT3",
      "state": "ready",
      "type": "standard",
      "general": {
        "statusAge": "05:00:00"
      }
    }
  ]
}

```

## Related reference

[“Response body attributes for agents” on page 2203](#)

When you use the HTTP GET method with the agent object to request information about agents, the following attributes are returned within named JSON objects.

### *Response body attributes for agents*

When you use the HTTP GET method with the agent object to request information about agents, the following attributes are returned within named JSON objects.

The following objects are available:

- [“general” on page 2203](#)
- [“qmgrConnection” on page 2204](#)
- [“connectDirectBridge” on page 2205](#)
- [“protocolBridge” on page 2205](#)
- [“standbyInstance” on page 2206](#)

## general

### description

String.

Specifies the description of the agent.

### statusAge

String.

Specifies the age of the agent. The age is calculated as the difference in time between the system time of the machine where the coordination queue manager is running, and the time the last status was published by an agent.

### version

String.

Specifies the version of the queue manager.

### level

String.

Specifies the build level on which the queue manager is running.

### statusPublicationRate

Integer.

Specifies the rate, in seconds, that the agent publishes its status.

The default value for this attribute is 300 seconds.

### statusPublishTime

String.

Specifies the time at which the agent published its status, in Universal Time Constant format.

**maximumQueuedTransfers**

Integer.

Specifies the maximum number of pending transfers that can be queued by an agent until the agent rejects a new transfer request.

The default value for this attribute is 1000.

**maximumQueuedTransfers**

Integer.

Specifies the maximum number of pending transfers that can be queued by an agent until the agent rejects a new transfer request.

The default value for this attribute is 1000

**maximumDestinationTransfers**

Integer.

Specifies the maximum number of concurrent transfers that the destination agent processes at any given point in time.

The default value for this attribute is 25.

**maximumSourceTransfers**

Integer.

Specifies the maximum number of concurrent transfers that the source agent processes at any given point in time.

The default value for this attribute is 25.

**operatingSystem**

String

Specifies the operating system where the agent queue manager is created.

**qmgrConnection**

This object provides information about the queue manager connections.

**qmgrName**

String.

Specifies the name of the agent queue manager.

**transportType**

String.

Specifies the transport type in which the agent is connecting with the queue manager. The transport type can be client or bindings.

The default value is bindings.

**host**

String.

Specifies the agent queue manager host name; applicable only if **transportType** is client.

**port**

Integer.

Specifies the agent queue manager channel communication port; applicable only if **transportType** is client.

**channelName**

String.

Specifies the agent queue manager channel; applicable only if **transportType** is client.

The default value for this attribute is SYSTEM.DEF.SVRCONN

**standbyHost**

String.

Specifies the host name used by client connections to connect to the standby instance of a multi-instance agent queue manager.

**standbyPort**

Integer.

Specifies the port number through which a client can connect to the standby instance of a multi-instance agent queue manager.

The default value for this attribute is -1.

**connectDirectBridge**

This object provides information about to connect direct bridge type agent. For other type of agents this object is not added.

**nodeName**

String.

Specifies the name of the Connect:Direct node to use to transfer messages from this agent to the destination Connect:Direct nodes.

**host**

String.

Specifies the host name or IP address of the system where the Connect:Direct node, specified by the **-cdNode** parameter, is located.

If you do not specify the **-cdNodeHost** parameter, a default of the host name or IP address of the local system is used.

The default value for this attribute are the details of the host where it is configured, for example, localhost.

**port**

Integer.

Specifies the port number of the Connect:Direct node that client applications use to communicate with the node.

The default value for this attribute is 1363.

**protocolBridge**

This object provides information about protocol bridge type agent. For other type of agents this object is not added.

**endpoint**

String.

Specifies the number of endpoints the bridge can support.

The default value for this attribute is *multiple* from version 7.0.1.

**defaultServer**

String.

Specifies the host name or IP address of the default protocol server if it is set. If the default protocol field is not set, this value is blank.

The value is a complete string containing the protocol type, server, and port, in the following format:

```
<protocolType>://<serverName or IP address>:<port>
```

For example:

```
"ftp://localhost:21"
```

## standbyInstance

This object provides information about the status of the standby instance, and is present only if the agent is enabled as highly available.

### host

String

Specifies the agent queue manager host name.

### version

String.

Specifies the version of the queue manager. The version must be 9.1.4.0 or higher.

### Related tasks

[Getting started with the REST API for MFT](#)


### Related reference

“GET” on page 2197

Use the HTTP GET method with the agent resource to request information about Managed File Transfer agents.

### /admin/mft/call

You can use the HTTP GET method with the call resource to request information about the status of a Managed File Transfer managed call. You can use the HTTP POST method to create a managed call.


**Note:**  This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

For more information about managed calls, see [Managed calls](#).

### **GET - managed call**

Use the HTTP GET method with the call resource to request information about the status of a Managed File Transfer managed call. You can query only the managed calls that are initiated after the mqweb server is started.

### Note:

- This resource is available only from version 3 of the IBM MQ REST API.
-  This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

For more information about managed calls, see [Managed calls](#).

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2207](#)
- [Request body format](#)
- [“Security requirements” on page 2207](#)
- [Response status codes](#)
- [“Response headers” on page 2208](#)
- [Response body format](#)
- [Examples](#)

## Resource URL

`https://mqweb.ibm.com:9443/ibmmq/rest/v3/admin/mft/call`

returns the attributes of a specified managed call in the response body.

## Optional query parameters

### attributes

Specifies a comma-separated list of attributes to retrieve.

If you do not specify **attributes**, the default set of attributes is returned. See [“ Response body attributes for transfers”](#) on page 2271 for a list of the available attributes.

You cannot request the same attribute multiple times.

You can specify an asterisk, \*, to specify that all attributes are returned.

You can make a request that specifies attributes that are not valid for some of the managed call. However, if you make a request that specifies a managed call ID and includes attributes that are not valid for that managed call, an error occurs.

### limit

Specifies the maximum number of managed calls to retrieve.

This query parameter is valid only when no managed call ID is specified.

For example, if the `limit=200`, the REST API returns a maximum of 200 managed calls.

### after

Specifies a managed call ID. All managed calls that are initiated after the specified managed call are retrieved. If you specify **after**, you cannot also specify **before**.

This query parameter is valid only when no managed call ID is specified.

### before

Specifies a managed call ID. All managed calls that are initiated before that particular managed call are retrieved. If you specify **before**, you cannot also specify **after**.

This query parameter is valid only when no managed call ID is specified.

## Request headers

The following header must be sent with the request:

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## Request body format

None.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MFTWebAdmin, MFTWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

The security principal of the caller must be granted authority to subscribe to the SYSTEM.FTE topic.

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

## Response status codes

### 200

Manged call information retrieved successfully.

### 400

Invalid data provided.

For example, invalid attributes specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server. For more information, see [“Security requirements” on page 2207](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal is not a member of one or more of the MFTWebAdmin, MFTWebAdminRO, or MQWebUser roles. For more information about the access that is required, see [“Security requirements” on page 2207](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

A managed call with the specified ID does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

### Content-Type

This header is returned with a value of `application/json;charset=utf-8`.

### ibm-mq-rest-mft-total-managed-calls

This header is returned with a value that is the total number of managed calls that have details available in the mqweb server cache.

## Response body format

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `call`. Each element in the array is a JSON object that represents information about a manged call. Each of these JSON objects can contain the following objects and attributes.



**Attention:** Use the appropriate **name** parameter for the variable that your enterprise uses.

Which objects and attributes are returned depends on the URL that was specified for the request:

### job

JSON object.

Name of the group containing the name of the job given to a managed call.

### name

String

Specifies a user-defined job name for the managed call.

### agent

JSON object.



Name of the group containing details of the agent to which a managed call request is submitted

**name**

String

Attribute in the **agent** group and refers to the agent name

**qmgrName**

String

Attribute in the **agent** group and refers to the agent queue manager name

**priority**

Integer

Priority assigned to the managed call. Same as MQMD.**Priority** with a value of 0 to 9.

**userProperties**

JSON object

Name of the group that contains attributes where each attribute refers to user-defined meta data. The name and value of each attribute is defined by the user.

**command**

JSON object

Name of the group containing attributes describing a managed call request.

**retryCount**

Integer

Specifies the number of attempts to run the command before ceasing.

**retryWait**

Integer

Specifies the amount of time to wait, in seconds, between retry attempts.

**successReturnCode**

String

Specifies the condition, based on the return code from the executable, AntScript, or JCL, that must be true in the order for the managed call to be successful.

The condition is specified as an operator, followed by a value. Valid characters for the operator are >, <, !, and =.

It is valid to have a combination of more than one operator. See [“successrc” on page 2058](#) for more information on the operators.

For example, an expression ">2&<7&!5|0|14" indicates that the return codes of 0, 3, 4, 6, and 14 are considered as success.

The default value is zero.

**type**

String

Identifies the type of managed call. *antScript*, *executable*, and *jcl* are the supported values

**executable**

JSON object

A JSON object that can contain attributes related to a platform-specific executable program to be invoked. This object can be specified only when the value of the **type** attribute is *executable*.

**name**

String

Specifies the name of the program to run. This attribute is required if the **executable** JSON object is specified.

**arguments**

String

Specifies a list of user-defined custom data in space separated key=value pairs.

**antScript**

JSON object

A JSON object that can contain attributes related to an Apache Ant script to be invoked. This object can only be specified when the value of the **type** attribute is *antScript*.

**name**

String

Specifies the name of the Ant script to run.

**target**

String

Specifies the target to invoke in the specified Ant script. If this attribute is not specified, the target named **default** is invoked

**arguments**

JSON object

Specifies a list of user-defined custom data in key=value pairs.

**jcl**

JSON object

A JSON object that can contain attributes related to a z/OS JCL job to submit. This object can only be specified when the value of the **type** attribute is *jcl*.

**name**

String

Specifies the name of the JCL to submit.

**results**

JSON array

An array of JSON objects that describes the outcome of the processing of the script or program.

The response can contain multiple result object if the specified command has been retried.

Assuming the processing failed first and succeeded in the next attempt, the first result object has details of the failure and the second one has details of the success.

**returnCode**

Integer

Return code that describes the code returned by the processing of the command.

**time**

String

Date and time when processing of the command completed.

**outcome**

String

Outcome of the processing of a command. The possible values are *success* or *failure*.

**consoleOutput**

JSON array

A JSON array object that describes each line of the console output when the command was processed. This includes output in both `stdout` and `stderr`.

**retries**

Integer

Describes the number of times the command was retried before completing. The value can also be equal to the maximum retry count specified in the request if the command failed to succeed.

**finalOutcome**

String

Provides the description of the overall result of the processing of the command.

**id**

String

Unique identifier of the managed call.

**originator**

JSON object

A JSON object that identifies the initiator of managed call.

**host**

String

Identifies the name of the machine from where the managed call was submitted.

**userId**

String

Identifies the user that submitted the request.

**mqmdUserId**

String

Identifies the IBM MQ user that submitted the request.

**status**

JSON object

Contains attributes that are related to the status of the managed call. The **state** attribute in this object is always returned.**state**

String

Describes the state of the managed call. Possible values are *successful*, *failed*, or *in-progress***lastStatusUpdate**

String

Describes the time (in UTC) when the state of the managed call was last updated.

**statistics**

JSON object

Describes the statistics of a managed call. Attributes of this object include the start and end time of the managed call.

**Note:** The managed call fails to complete if the total length of the console output exceeds 10KB. Message BFGCR0004E is logged by the agent in such a situation.If an error occurs, see [REST API error handling](#).**Examples**

The following is an example of a response body with the default attributes of a managed call:

```

{
  "call": [
    {
      "agent": {
        "name": "SECURITIES.AGENT"
      },
      "id": "414D51204D4654514D2020202020202020987C936103A80140",
      "command": {
        "retryWait": 0,
        "successReturnCode": "0",
        "retryCount": 0,
        "type": "antScript",
        "antScript": {
          "name": "/usr/cmds/hubprocess.xml",
          "arguments": "out.file=c:/temp/Catted.xsd",
          "target": "concatenate"
        }
      },
      "originator": {
        "host": "host.johnwatson.com",

```

```

        "userId": "john.watson"
      },
      "status": {
        "state": "started"
      }
    }
  ]
}

```

The following is an example of a response body with the status of the call in started state:

```

{
  "call": [
    {
      "agent": {
        "qmgrName": "SECURITIES.QM",
        "name": "SECURITIES.AGENT"
      },
      "priority": 0,
      "userProperties": {
        "com.ibm.wmqfte.SourceAgent": "SECURITIES.AGENT",
        "com.ibm.wmqfte.OriginatingUser": "john.watson",
        "com.ibm.wmqfte.OriginatingHost": "host.johnwatson.com",
        "com.ibm.wmqfte.TransferId": "414d51204d4654514d202020202020987c936103a80140",
        "com.ibm.wmqfte.MqmdUser": "john.watson",
        "com.ibm.wmqfte.Priority": "0",
        "com.ibm.wmqfte.DestinationAgent": "SECURITIES.AGENT"
      },
      "command": {
        "retryWait": 0,
        "retryCount": 0,
        "successReturnCode": "0",
        "type": "antScript",
        "antscript": {
          "name": "/usr/sample/hubprocess.xml",
          "target": "concatenate",
          "arguments": "out.file=/usr/out/outfile.txt
            in.file1=/usr/sample/input/infile1.txt
            in.file2=/usr/sample/input/infile2.txt"
        }
      },
      "id": "414D51204D4654514D202020202020987C936103A80140",
      "originator": {
        "host": "host.johnwatson.com",
        "mqmdUserId": "john.watson",
        "userId": "john.watson"
      },
      "job": {
        "name": "pushsecurities"
      },
      "status": {
        "lastStatusUpdate": "2021-11-17T07:12:35.459Z",
        "state": "started"
      },
      "statistics": {
        "startTime": "2021-11-17T07:12:35.459Z"
      }
    }
  ]
}

```

The following is an example of a response body with failed status.

```

{
  "call": [
    {
      "agent": {
        "qmgrName": "SECURITIES.QM",
        "name": "SECURITIES.AGENT"
      },
      "priority": 0,
      "userProperties": {
        "com.ibm.wmqfte.SourceAgent": "SECURITIES.AGENT",
        "com.ibm.wmqfte.OriginatingUser": "john.watson",
        "com.ibm.wmqfte.OriginatingHost": "host.johnwatson.com",
        "com.ibm.wmqfte.TransferId": "414d51204d4654514d2020202020987c936103a80140",
        "com.ibm.wmqfte.MqmdUser": "john.watson",
        "com.ibm.wmqfte.Priority": "0",
        "com.ibm.wmqfte.DestinationAgent": "SECURITIES.AGENT"
      },

```

```

"results": {
  "result": [{
    "returnCode": 1,
    "completionTime": "2021-11-23T03:40:05.794Z",
    "outcome": "failure"
    "consoleOutput": [
      "BFGCL0207E: Target \"concatenate1\" does not exist in the project \"null\"."
    ]
  }],
  "retries": 0,
  "finalOutcome": "failure"
},
"command": {
  "retryWait": 0,
  "retryCount": 0,
  "type": "antScript",
  "antscript": {
    "name": "/usr/sample/hubprocess.xml",
    "target": "concatenate"
    "successReturnCode": "0"
    "arguments": "out.file=/usr/out/outfile.txt
in.file1=/usr/sample/input/infile1.txt
in.file2=/usr/sample/input/infile2.txt"
  }
},
"id": "414D51204D4654514D20202020202020987C936103A80140",
"originator": {
  "host": "host.johnwatson.com",
  "mqmdUserId": "john.watson",
  "userId": "john.watson"
},
"job": {
  "name": "pushsecurities"
},
"status": {
  "lastStatusUpdate": "2021-11-17T07:12:35.459Z",
  "state": "failed"
},
"statistics": {
  "startTime": "2021-11-23T03:40:03.967Z",
  "endTime": "2021-11-23T03:40:05.794Z"
}
}
]
}

```

## Related reference


[“POST - managed call” on page 2213](#)

Use the HTTP POST method with the call resource to create a Managed File Transfer managed call.

## **POST - managed call**

Use the HTTP POST method with the call resource to create a Managed File Transfer managed call.

### Note:

- This resource is available only from version 3 of the IBM MQ REST API.
-  This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

For more information about managed calls, see [Managed calls](#).

- [Resource URL](#)
- [“Request headers” on page 2214](#)
- [Request body format](#)
- [“Security requirements” on page 2216](#)
- [Response status codes](#)
- [“Response headers” on page 2218](#)
- [Response body format](#)

- [Examples](#)

## Resource URL

`https://mqweb.ibm.com:9443/ibmmq/rest/v3/admin/mft/call`

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring the HTTP and HTTPS ports](#).

## Request headers

The following headers must be sent with the request:

### Content-Type

This header must be sent with a value of `application/json` optionally followed by `; charset=UTF-8`.

### ibm-mq-rest-csrf-token

This header must be set, but the value can be anything, including being blank.

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## Request body format

The request body must contain attributes for only one managed call. Submitting multiple managed calls in a single request is not allowed.

The body can contain details of an **AntScript, executable** or a **JCL** (for agents running on z/OS).

The following section describes the request body format for each of the types. Note that the attributes of the command section of a managed call are the same as the attributes described in the [“preSourceCall” on page 2265](#) section of *Request body attributes for transfers with HTTP POST*.



**Attention:** Use the appropriate **name** parameter for the variable that your enterprise uses.

Which objects and attributes are returned depends on the URL that was specified for the request:

### job

JSON object.

Optional

Name of the group containing the name of the job given to a managed call.

### name

String

Required if you specified the **job** attributes

Specifies a user-defined job name for the managed call.

### agent

JSON object.

Required

Name of the group containing details of the agent to which a managed call request is submitted

### name

String

Required

Attribute in the **agent** group and refers to the agent name

### qmgrName

String

Required

Attribute in the **agent** group and refers to the agent queue manager name

**priority**

Integer

Optional

Priority assigned to the managed call. Same as MQMD.**Priority** with a value of 0 to 9.

**userProperties**

JSON object

Optional

Name of the group that contains attributes where each attribute refers to user-defined meta data. The name and value of each attribute is defined by the user.

**command**

JSON object

Required

Name of the group containing attributes describing a managed call request.

**retryCount**

Integer

Optional - zero is assumed if not specified.

Specifies the number of attempts to run the command before ceasing.

**retryWait**

Integer

Optional - zero is assumed if not specified.

Specifies the amount of time to wait, in seconds, between retry attempts.

**successReturnCode**

String

Required

Specifies the condition, based on the return code from the executable, AntScript, or JCL, that must be true in the order for the managed call to be successful.

The condition is specified as an operator, followed by a value. Valid characters for the operator are >, <, !, and =.

It is valid to have a combination of more than one operator. See [“successrc” on page 2058](#) for more information on the operators.

For example, an expression ">2&<7&!5|0|14" indicates that the return codes of 0, 3, 4, 6, and 14 are considered as success.

The default value is zero.

**type**

String

Required

Identifies the type of managed call. *antScript*, *executable*, and *jcl* are the supported values

**executable**

JSON object

Required if the value of the **type** attribute is *executable*, and can be specified only when the value of the **type** attribute is *executable*

A JSON object that can contain attributes related to a platform-specific executable program to be invoked.

**name**

String

Required if the value of the **type** attribute is *executable*, and can be specified only when the value of the **type** attribute is *executable*

Specifies the name of the program to run.

#### arguments

String

Optional

Specifies a list of user-defined custom data in space separated key=value pairs.

#### antScript

JSON object

Required if the value of the **type** attribute is *antScript*, and can be specified only when the value of the **type** attribute is *antScript*

A JSON object that can contain attributes related to an Apache Ant script to be invoked.

#### name

String

Required if the value of the **type** attribute is *antScript*, and can be specified only when the value of the **type** attribute is *antScript*

Specifies the name of the Ant script to run.

#### target

String

Specifies the target to invoke in the specified Ant script. If this attribute is not specified, the target named **default** is invoked

#### arguments

JSON object

Optional

Specifies a list of user-defined custom data in key=value pairs.

#### jcl

JSON object

Required if the value of the **type** attribute is *jcl*, and can be specified only when the value of the **type** attribute is *jcl*

A JSON object that can contain attributes related to a z/OS JCL job to submit.

#### name

String

Required if the value of the **type** attribute is *jcl*, and can be specified only when the value of the **type** attribute is *jcl*

Specifies the name of the JCL to submit.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of the MFTWebAdmin or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

If you have set up a [user sandbox](#), grant additional authority to the mqweb server user ID to access the specified file system location. For example, to restrict file system or queue access to only user IDs that contain the characters a, A or b, B:

```
<tns:userSandboxes
  xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
```



```

xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
<tns:agent>
  <tns:sandbox user="^[a-bA-B]*$" userPattern="regex">
    <tns:read>
      <tns:include name="/mountpath/**"/>
      <tns:include name="**" type="queue"/>
    </tns:read>
    <tns:write>
      <tns:include name="/mountpath/**"/>
      <tns:include name="**" type="queue"/>
    </tns:write>
  </tns:sandbox>
</tns:agent>
</tns:userSandboxes>

```

If you have MFT authority checking turned on, grant additional authorities as described in [Restricting user authorities on MFT agent actions](#).

For the MFTWebAdmin role, managed call requests are submitted under the context of the mqweb server user ID. To distinguish between different principals of the MFTWebAdmin role, and for audit purposes, the managed call request submitted contains the name of the authenticated user as the managed call originator. This method ensures that there is a record of who initiated the managed call request.

For example, if the user mftadminusr, of the MFTWebAdmin role, initiates a managed call, the originator data in the XML that is created to describe the managed call has mftadminusr in the userID element, as shown in this example:

```

<originator>
  <hostName>example.com</hostName>
  <userID>mftadminusr</userID>
  <mqmdUserId>mqm</mqmdUserId>
</originator>

```

where:

**hostName**

Is the name of host where the mqweb server is running.

**userId**

Is the name of the user that is logged in to the mqweb server.

**mqmdUserId**

Is the name of the user under which the mqweb server is running and connects to the command queue manager.

If the caller is a member of the MQWebUser role, the security principal of the caller must be granted one of the following authorities:

1. If the command queue is local, that is, the command queue manager and source agent queue manager are the same, grant put authority to the command queue.
2. If the command queue is remote, that is, the command queue manager and source agent queue manager are different, grant put authority to the transmission queue.

**Notes:**

- If the user ID of a principal that is a member of the MQWebUser role is longer than 12 characters, the request fails. Response status code 403 is returned to the caller.
- If the caller is assigned more than one role, the highest privilege role that is applicable to the operation is used.

If security is disabled on the mqweb server, the transfer request submitted contains the name "UNAUTHENTICATED" as the transfer originator.

## Response status codes

### 202

The managed call request has been accepted by the REST API. It might still get rejected by the MFT agent. You should issue a GET command, using the URL from the `location` response header to ascertain the state of the managed call.

### 400

Invalid data provided.

For example, invalid attributes specified.

### 401

Not authenticated.

The user must be authenticated to the mqweb server. For more information, see [“Security requirements”](#) on page 2216.

The `ibm-mq-rest-csrf-token` header must also be specified.

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ or MFT resources.
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 500

Server issue, or error code from IBM MQ or MFT.

### 503

Queue manager not running.

## Response headers

The following header is returned with the response:

### location

If the request was successful, this header specifies the URL for the new managed call.

## Response body format

The response body is empty if the transfer is created successfully.

If an error occurs, the response body contains an error message; see [REST API error handling](#).

## Examples

An example of a request body format for a managed call for running an *antScript*:

```
{
  "job": {
    "name": "pushsecurities"
  },
  "agent": {
    "name": "SECURITIES.AGENT",
    "qmgrName": "SECURITIES.QM"
  },
  "priority": 0,
  "command": {
    "retryCount": 0,
    "retryWait": 0,
    "successReturnCode": "0",
    "type": "antScript",
    "antScript": {
      "name": "publish.xml",
      "target": "publishsecurities",
      "arguments": "filename=abc.csv updateInterval=5"
    }
  }
}
```

```
}  
}
```

An example of a request body format for a managed call for running an *executable*:

```
{  
  "job": {  
    "name": "compressfiles"  
  },  
  "agent": {  
    "name": "SECURITIES.AGENT",  
    "qmgrName": "SECURITIES.QM"  
  },  
  "priority": 0,  
  "command": {  
    "retryCount": 0,  
    "retryWait": 0,  
    "successReturnCode": "0",  
    "type": "executable",  
    "executable": {  
      "name": "compress.sh",  
      "arguments": "filename zlib"  
    }  
  }  
}
```

An example of a request body format for a managed call with JCL to be run by an agent running z/OS:

```
{  
  "job": {  
    "name": "pushsecurities"  
  },  
  "agent": {  
    "name": "SECURITIES.AGENT",  
    "qmgrName": "SECURITIES.QM"  
  },  
  "priority": 0,  
  "command": {  
    "retryCount": 0,  
    "retryWait": 0,  
    "successReturnCode": "0",  
    "type": "jcl",  
    "jcl": {  
      "name": "publish",  
    }  
  }  
}
```

### Related reference

[“GET - managed call” on page 2206](#)

Use the HTTP GET method with the `call` resource to request information about the status of a Managed File Transfer managed call. You can query only the managed calls that are initiated after the mqweb server is started.

### **/admin/mft/monitor**

You can use the HTTP GET method with the `monitor` resource to request information about a Managed File Transfer resource monitor. You can use the HTTP POST method to create a resource monitor, and the HTTP DELETE method to delete a resource monitor.

**Note:** **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

### Related tasks

[Getting started with the REST API for MFT](#)

### Related reference

[“/admin/mft/agent” on page 2197](#)

You can use the HTTP GET method with the `agent` resource, to request information about Managed File Transfer agents.

[“/admin/mft/transfer” on page 2252](#)

You can use the HTTP GET method with the `transfer` resource to request information about Managed File Transfer transfers. You can use the HTTP POST method to put a transfer request message to the command queue manager, which will be routed to the source agent queue manager.

## **POST**

Use the HTTP POST method with the `monitor` resource to create a Managed File Transfer resource monitor.

### **Note:**

- Set the command queue manager in the configuration before issuing any MFT Create Monitor or Transfer REST API command. See [Configuring the REST API for MFT](#) for more information.
- **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).
- [Resource URL](#)
- [“Request headers” on page 2220](#)
- [Request body format](#)
- [“Security requirements” on page 2221](#)
- [Response status codes](#)
- [“Response headers” on page 2222](#)
- [Response body format](#)
- [Examples](#)

## **Resource URL**

`https://host:port/ibmmq/rest/v2/admin/mft/monitor`

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring the HTTP and HTTPS ports](#).

## **Request headers**

The following headers must be sent with the request:

### **Content-Type**

This header must be sent with a value of `application/json` optionally followed by `; charset=UTF-8`.

### **ibm-mq-rest-csrf-token**

This header must be set, but the value can be anything, including being blank.

### **Authorization**

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## **Request body format**

The request body must be in JSON format in UTF-8 encoding. Attributes marked *required* are mandatory, and if you do not provide values for the other parameters in the request body, the default values are used.

### **name**

JSON string.

Contains the name of the resource monitor.

The name is not case sensitive - lower case characters are folded to upper case characters, and you cannot use the wildcard character (\*).

The name is required.

**type**

JSON string.

Type of the resource to be monitored.

**general**

JSON object.

This JSON object contains details of the poll interval, the units of the poll interval, and the matches per task.

**resource**

JSON object.

This JSON object contains details of the resource, that is the name for both monitoring a queue and a directory, and for a directory resource the recursion level.

The **name** attributes in this object are required.

**triggerCondition**

JSON object.

This JSON object contains the type attribute and various other attribute depending upon whether the resource type is a directory or a queue. See [“Request body attributes for MFT resource monitors” on page 2224](#) for details of this attribute.

The **type** attributes in this object are required.

**userProperties**

JSON object.

Specifies the user-defined metadata that is passed to the exit points of the monitor. The parameter can take one or more name pairs that are separated by commas. Each name pair consists of a name=value.

**transferDefinition**

JSON object.

Contains details about the transfer, for example, source agent and queue manager, destination agent and queue manager, and so on. See [“Request body attributes for MFT resource monitors” on page 2224](#) for details of this attribute.

[“Request body attributes for MFT resource monitors” on page 2224](#) lists all the attributes.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of the MFTWebAdmin or MFTWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

If you have set up a [user sandbox](#), and [MFT authority checking](#), or MFT authority checking, is turned on, you need to grant an additional authority for the user that started the WebSphere Liberty server to access the specified file system location.

For the MFTWebAdmin role, transfer requests are submitted under the context of the user that started the Liberty server. To distinguish between different principals of the MFTWebAdmin role, and for audit purposes, the transfer request submitted contains the name of the authenticated user as the transfer originator. This method ensures that there is a record of who initiated the transfer request.

For example, if the user `mftadminusr`, of the `MFTWebAdmin` role, initiates a transfer, the originator data in the XML has `mftadminusr` in the `userID` element, as shown in this example:

```
<originator>
  <hostName>example.com.</hostName>
  <userID>mftadminusr</userID>
</originator>
```

If the caller is a member of the `MQWebUser` role, the security principal of the caller must be granted one of the following authorities:

1. If the command queue is local, that is, the command queue manager and source agent queue manager are the same, grant put authority to the command queue.
2. If the command queue is remote, that is, the command queue manager and source agent queue manager are different, grant put authority to the transmission queue.

#### Notes:

- If the user ID of a principal that is a member of the `MQWebUser` role is longer than 12 characters, the request fails. Response status code 403 is returned to the caller.
- If the caller is assigned more than one role, the highest privilege role that is applicable to the operation is used.

If security is disabled on the `mqweb` server, the transfer request submitted contains the name "UNAUTHENTICATED" as the transfer originator.

## Response status codes

### 202

The create monitor request has been accepted by the `mqweb` server. It might still get rejected by the MFT agent.

### 400

Invalid or unknown data provided to create resource monitor.  
For example, invalid attributes specified.

### 401

Not authenticated.

The user must be authenticated to the `mqweb` server. For more information, see [“Security requirements”](#) on page 2221.

The `ibm-mq-rest-csrf-token` header must also be specified.

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the `mqweb` server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ or MFT resources.
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 500

Server issue, or error code from IBM MQ or MFT.

## Response headers

The following header is returned with the response:

### location

If the request is successfully submitted, the **location** attribute in the response header is updated with the url, through which details about the resource monitor can be further queried.

## Response body format

The response body is empty if the transfer is created successfully.

If an error occurs, the response body contains an error message; see [REST API error handling](#).

## Examples

The following example creates a resource monitor for monitoring a directory:

```
{
  "name": "DIRMONREGEX",
  "type": "directory",
  "general": { "pollingInterval": 1, "pollingIntervalUnit": "minutes", "matchesPerTask": 5 },
  "userProperties": { "companyName": "IBM", "unit": "ISL" },
  "resource": { "name": "/MFT/TRIGGER", "recursionLevel": 2 },
  "triggerCondition": { "excludePattern": "*.xls", "includePattern":
    "*.txt", "type": "matchAll"
  },
  "transferDefinition" {
    "sourceAgent": { "qmgrName": "srcQmgr", "name": "SRC" },
    "destinationAgent": { "qmgrName": "desQmgr", "name": "DES" },
    "transferSet": {
      "item": [
        { "source": { "name": "C:\src\test.txt", "type": "file" },
          "destination": { "name": "C:\dst\test.txt", "type": "file" } } ],
      "userProperties": { "ARCHIVE_PATH": "C:\\MFT\\ARCHIVE",
        "REJECT_PATH": "C:\\MFT\\REJECT" },
      "postSourceCall": { "name": "posttransfersource.exe",
        "executable": { "arguments": "data1 data2" } },
      "postDestinationCall": { "name": "posttransferdest.exe",
        "executable": { "arguments": "dataDest1 dataDest2" } } },
      "preDestinationCall": { "name": "pretransferdest.exe" },
      "preSourceCall": { "name": "posttransferdest.exe",
        "executable": { "arguments": "predata1 predata2" } },
      "priority": 0,
      "recoveryTimeout": 21600 } }
}
```

The following example creates a resource monitor for monitoring a queue:

```
{
  "name": "QMON", "type": "queue",
  "general": { "pollingInterval": 1, "pollingIntervalUnit": "minutes", "matchesPerTask": 5 },
  "triggerCondition": { "excludePattern": "*.xls", "includePattern": "*.txt", "type":
    "matchAll" },
  "userProperties": { "companyName": "IBM", "unit": "ISL" },
  "resource": { "name": "MSGQ", "matchCondition": "containsMessages" },
  "transferDefinition": {
    "job": { "name": "testJob" },
    "sourceAgent": { "name": "SRC", "qmgrName": "srcQmgr" },
    "destinationAgent": { "name": "DES", "qmgrName": "desQmgr" },
    "transferSet": {
      "item": [ {
        "source": { "name": "C:\temp\src\test.txt", "type": "file",
          "recursive": false, "disposition": "leave" },
        "destination": { "name": "LQ@NYQMGR", "type": "queue",
          "actionIfExists": "error", "delimiterType": "size",
          "messagePersistence": "persistent",
          "queueExtended": { "messageSize": 4, "setMQProperties": "false" } },
        "priority": 1, "recoveryTimeout": -1, "checksum": "md5", "mode": "text" } } ] }
}
```

The following example creates a resource monitor for monitoring a directory with more attributes:

```
{
  "name": "DIRMONREGEX", "type": "directory", "agentName": "SRC",
  "general": { "pollingInterval": 1, "pollingIntervalUnit": "minutes", "matchesPerTask": 5 },
  "userProperties": { "companyName": "IBM", "unit": "ISL" },
  "resource": { "name": "/MFT/TRIGGER", "recursionLevel": 2 },
  "triggerCondition": { "matchPattern": "[a-zA-Z]{3}", "excludePattern": "[d-fD-F]{3}",
    "patternType": "regularExpression",
    "matchCondition": { "matchNoSizeChangeInterval": 5 } } },
```

```

"transferDefinition": {
  "sourceAgent": { "name": "SRC", "qmgrName": "srcQmgr" },
  "destinationAgent": { "name": "NY.AGENT", "qmgrName": "NYQMGR" },
"transferSet": {
  "item": [ { "source": { "name": "C:\temp\src\source.exe", "type": "file" },
            "destination": { "name": "C:\temp\dst", "type": "file"},
            "mode": "binary" } ] } }
}

```

The following example creates a resource monitor, demonstrating variable substitution functionality:

```

{ "name":
"VARSUB-TEST", "type": "directory", "agentName": "SRC",
"general": { "pollInterval": 1, "pollIntervalUnit": "minutes"},
"resource": { "name": "c:\source_dir"},
"triggerCondition": { "excludePattern": "*.exe", "includePattern": "*.txt",
"matchPattern": "wildcard", "type": "matchAll" },
"transferDefinition": {
  "job": { "name": "varSub"},
  "sourceAgent": { "name": "SRC", "qmgrName": "gandhi"},
  "destinationAgent": { "name": "DES", "qmgrName": "gandhi", "actionIfExists": "overwrite"},
  "transferSet": { "item": [ {
    "destination": { "name": "C:\\dest\\${fileName}", "type": "directory"},
    "source": { "name": "C:\\source_dir\\file.txt", "type": "file"},
    "mode": "text" } ] } }
}

```

## Related tasks

[Getting started with the REST API for MFT](#)

## Related reference

[“Request body attributes for MFT resource monitors” on page 2224](#)

The Create Monitor REST API takes the input attributes as JSON objects.

### *Request body attributes for MFT resource monitors*

The Create Monitor REST API takes the input attributes as JSON objects.

The following list shows the attributes that you need to be provide to a REST call:

- [name](#)
- [type](#)
- [“general” on page 2225](#)
- [“resource” on page 2225](#)
- [“transferDefinition” on page 2225](#)
- [“triggerCondition” on page 2232](#)

### **name**

String.

A unique name for the resource monitor or queue.

The name is not case sensitive - lower case characters are folded to upper case characters, and you cannot use the wildcard character (\*).

The name attribute is required.

### **type**

String.

Type of the resource monitor

The value is one of the following values:

#### **directory**

Type of the resource to create is a file system directory.

#### **queue**

Type of the resource to create is an IBM MQ queue.



## general

Group element that defines the basic attributes of the monitor.

### pollInterval

Integer.

Frequency, in units of time, at which a monitor polls a resource.

The default value is 1.

### pollIntervalUnit

String.

Specifies the time interval for the **pollInterval** attribute. Possible values are seconds, minutes, hours, days.

The default value is minutes.

### matchesPerTask

Integer.

Maximum of trigger matches to include in a single task.

The default value is 2.

## resource

Group element that defines the details about the resource to be monitored.

The **name** attribute in this object is always returned.

### name

String.

Specifies the name or the resource to be monitored. It can be absolute path of a file or directory, or the name of a queue.

### recursionLevel

Integer.

Specifies the level in the directory structure that needs to be monitored.

The default value is 1.

**Note:** This attribute is valid only for a directory type of resource monitor.

## transferDefinition

This attribute contains details for the transfer, which is initiated when the trigger condition is satisfied.

### destinationAgent

Group element containing elements that define a destination agent.

The **name** and the **qmgrName** attributes in this object are always returned.

#### qmgrName

String.

The name of the queue manager on the destination system.

#### name

String.

The name of the agent on the destination system.

### job

Contains the name of the transfer job:

#### name

String.

User defined job name for the transfer.

**sourceAgent**

Group element containing elements that define a source agent.

The **name** and the **qmgrName** attributes in this object are always returned.

**qmgrName**

String.

The name of the queue manager on the source system.

**name**

String.

The name of the agent on the source system.

**transferSet**

Group element that contains the elements specifying a transfer request.

The **item** attribute in this object is always returned.

**priority**

Number (optional).

Priority assigned to the transfer request with zero being the default, if no value is set.

**userProperties**

Object (optional).

User defined properties specified in the transfer request.

**item**

Object.

Array of group elements that describes the source and destination item configuration to transfer.

**source**

Object.

Group element that contains the attributes of a source item.

The **name** and **type** attributes in this object are always returned.

**name**

String.

Specifies the absolute path of the file, directory, data set, partitioned data set, or queue at source end.

**type**

String.

The type of source. The value is one of the following values:

**queue**

Specifies an IBM MQ queue as the source.

**file**

Specifies a file as the source.

**directory**

Specifies a directory as the source.

**sequentialDataset**

Specifies a z/OS sequential data set as the source.

**partitionedDataset**

Specifies a z/OS partitioned data set as the source.

**recursive**

Boolean (optional).

Specifies that files are transferred recursively in subdirectories when the source element is a directory, or contains wildcard characters.

**disposition**

String (optional).

Specifies the action that is taken on the source element when a source has successfully been transferred to its destination. possible values are:

**leave**

The source files are left unchanged.

**delete**

The source files are deleted from the source system after the source file is successfully transferred.

**encoding**

String (optional)

Specifies which character encoding to use, to read the source file when performing character conversion. This option is only applicable to text files and the possible value is any valid code page number.

**datasetExtended**

Object (optional).

Group element that defines additional attributes of the source specification, if the source is a z/OS data set in a transfer request.

The **hexDelimiters** and **delimiterPosition** attributes in this object are always returned.

**keepTrailingSpaces**

Boolean (optional).

Describes the action that is taken if there are trailing spaces in the source records that are read from a fixed-length-format record-oriented file (for example, a z/OS data set) as part of a text mode transfer.

**hexDelimiters**

String.

For source files that are record oriented (for example, z/OS data sets), specifies one or more byte values to insert as the delimiter when appending records into a binary file. Each value is represented as two hexadecimal digits in the range 00-FF, prefixed by x.

**delimiterPosition**

String

Specifies the position of insertion for source text and binary delimiters. The value is one of the following values:

**prefix**

The delimiters are inserted at the start of each record.

**postfix**

The delimiters are inserted at the end of each record; this is the default option.

**queueExtended**

Object (optional).

Group element that defines additional attributes of a source specification, if the source is an IBM MQ queue in a transfer request.

The **useMessageGroup** and **groupID** attributes in this object are always returned.

**useMessageGroup**

Boolean.

Specifies that the messages are grouped by IBM MQ group ID. The first complete group is written to the destination file. If this parameter is not specified, all messages on the source queue are written to the destination file.

**groupID**

String.

Group ID to be used when getting messages from a queue.

**textDelimiters**

String (optional).

Specifies a sequence of text to insert as the delimiter, when appending multiple messages to a text file.

**hexDelimiters**

String (optional).

Comma separated string of hexadecimal bytes to use, when appending multiple messages to a file. For example x12 or x03 , x7F.

**delimiterPosition**

String (optional).

Defines where the delimiters are positioned in the message being put to the source queue. The value is one of the following values:

**prefix**

Before the beginning of the message body.

**postfix**

After the end of the message body; this is the default option.

**messageArrivalWaitTime**

Integer.

Time in seconds, to wait for arrival of messages in the source queue.

**destination**

Object.

Group element that contains the attributes of a destination item.

The **name** and **type** attributes in this object are always returned.

**name**

String.

Specifies the absolute path of the file, directory, data set, partitioned data set, or queue at the destination end.

**type**

String.

The type of destination. The value is one of the following values:

**queue**

Specifies an IBM MQ queue as the destination.

**file**

Specifies a file as the destination.

**directory**

Specifies a directory as the destination.

**sequentialDataset**

Specifies a z/OS sequential data set as the destination.

**partitionedDataset**

Specifies a z/OS partitioned data set as the destination.

**actionIfExists**

String(optional).

Specifies the action that is taken if a destination file exists on the destination system. The value is one of the following values:

**error**

Reports an error and the file is not transferred; this is the default value.

**overwrite**

Overwrites the existing destination file.

**encoding**

String (optional).

Specifies which character encoding to use, to read the source file when performing character conversion. This option is only applicable to text files and the possible value is any valid code page number.

**endOfLine**

String (optional).

Specifies the end-of-line characters that are used when the file is written at the destination. This option is applicable to text files only.

**userId**

String (optional).

The name of the user, whose destination file space the files are transferred into.

**datasetExtended**

Object (optional).

Group element that defines additional attributes of the destination specification, if the destination is a z/OS data set in a transfer request.

**truncateRecords**

Boolean.

Specifies that destination records longer than the LRECL data set attribute are truncated. If this parameter is not specified, the records are wrapped. This parameter is valid only for text mode transfers where the destination is a data set.

**queueExtended**

Object (optional).

Group element that defines additional attributes of a destination specification, if the destination is an IBM MQ queue in a transfer request.

The **messagePersistence** and **delimiterType** attributes in this object are always returned.

**messagePersistence**

String.

Defines if the message put to the destination queue is persistent or non-persistent. The value is one of the following values:

**persistent**

Messages are persistent.

**nonPersistent**

Messages are non-persistent.

**asQueueDefault**

Message persistency is set, depending on the queue definition.

**delimiterType**

String.

Defines the type of delimiter to use when splitting incoming data into messages. The value is one of the following values:

**size**

Split based on given size.

**binary**

Split based on given delimiters.

**hexDelimiters**

String (optional).

Comma separated string of hexadecimal bytes to use when splitting messages. For example x12 or x03 , x7F.

**textDelimiters**

String (optional).

Specifies the Java regular expression to use, when splitting a text file into multiple messages.

**includeDelimitersInMessage**

Boolean.

Defines whether delimiters are included in a message being put to the destination queue.

**delimiterPosition**

String

Defines where the delimiters are positioned in the message being put to the destination queue. The value is one of the following values:

**prefix**

Before the beginning of the message body.

**postfix**

After the end of the message body; this is the default option.

**setMQProperties**

Boolean (optional).

Valid only when the destination is a queue. The value is one of the following values:

**true**

Sets message properties on the first message that is created by the transfer.

**false**

Does not set message properties on the first message that is created by the transfer.

**messageSize**

Number.

Defines a size in bytes to split the incoming data into the message.

**checksum**

String (optional).

Checksum method for verifying data integrity. The value is one of the following values:

**md5**

MD5 algorithm used for integrity validation.

**none**

No checksum validation.

**mode**

String (optional).

Specifies the transfer mode as either binary or text. The value is one of the following values:

**text**

Data is transferred as text.

**binary**

Data is transferred in binary.

**recoveryTimeout**

Number (optional).

Time in seconds to wait for a transfer to recover, with -1 being the default if no value is set.

**preSourceCall**

Object (optional).

Group elements that contain the elements for program invocation before a transfer begins at the source.

These group elements are not present if a resource monitor is not configured to use any program invocation.

**type**

String (optional).

Defines the type of the program to be invoked. The value is one of the following values:

**executable**

This value is the default value.

Defines attributes for a platform specific executable program:

**name**

String.

Name of the program to process.

**arguments**

String (optional).

Argument or arguments to be passed to the program being invoked.

**antScript**

Defines attributes for Ant Script:

**name**

String.

Name of the Ant script to process.

**target**

String (optional)

Target to invoke in the specified Ant script. Attribute is not present in the JSON response, if the default target is to be invoked.

**arguments**

String (optional).

A list of user defined custom data in space separated key=value pair of type **String**. For example:

```
"arguments": "coffeeType=Arabica teaChoice=lemon"
```

**jcl**

Defines attributes for z/OS JCL to submit.

**name**

String.

Name of the JCL to submit.

**retryCount**

Number (optional).

A positive number of attempts to run the command before ceasing.

**retryWait**

Number (optional).

Amount of time to wait, in seconds, between two retry attempts.

**successReturnCode**

String (optional).

Reason code that is returned when transfer is complete. This is looked for before running the specified program, script, or JCL. This return code is a combination of an operator and value in the form of "[>|<|!] value". Note that it is valid to have a combination of more than one operator, for example ">= 40".

**postSourceCall**

Object (optional).

Group elements that contain the elements for program invocation after a transfer completes at source. This object contains the same elements as **preSourceCall**.

**preDestinationCall**

Object (optional).

Group elements that contain the elements for program invocation before a transfer begins at the destination. This object contains the same elements as **preSourceCall**.

**postDestinationCall**

Object (optional).

Group elements that contain the elements for program invocation after a transfer completes at the destination. This object contains the same elements as **preSourceCall**.

**triggerCondition**

Group element that defines details of a trigger condition used by a resource monitor.

**type**

String.

Indicates the type of matching done, to decide on triggering a transfer. Possible values are:

For resource type **Directory**:

**matchAll**

Must match the value specified for the **includePattern** and **excludePattern** attributes.

**matchNone**

None of the files in the monitored directory match the value specified for the **includePattern** and **excludePattern** attributes.

**noChangeInSize**

Initiate a transfer, if the size of the file being monitored does not change for a specified number of poll intervals.

**sizeGreaterOrEqualTo**

Initiate a transfer, if the size of the file being monitored is greater than or equal to a specified size.

For resource type **Queue**:

**queueNotEmpty**

Queue must have at least one message.

**completeMessageGroups**

Queue must have at least one group of messages.

**noFileSizeChangePollCount**

Number.

Refers to the number of polling intervals during which the size of the monitored file does not change. Used in conjunction with the **noChangeInSize** attribute

**fileSize**

Number.

Refers to the size of the trigger file being monitored, whose size is equal to or greater. Used in conjunction with the **sizeGreaterOrEqualTo** attribute.

**fileSizeUnit**

String



Defines the unit for the **fileSize** attribute. The value is one of the following values:

**bytes**

File size unit is in bytes

**kilobytes**

File size unit is in kilobytes

**megabytes**

File size unit is in megabytes

**gigabytes**

File size unit is in gigabytes

**includePattern**

String.

A pattern of the name, or names, of files to be included, while doing match for a trigger condition.

**excludePattern**

String.

A pattern of the name, or names, of files to be excluded, while doing match for a trigger condition.

**matchPattern**

String.

Indicates how to interpret the contents of the **includePattern** and **excludePattern** attributes.

The value is one of the following values:

**wildcard**

- Indicates the **includePattern** and **excludePattern** attributes contain wildcard characters, for example, \*.

**regularExpression**

Indicates the **includePattern** and **excludePattern** attributes contain Java regular expressions.

**Related tasks**

[Getting started with the REST API for MFT](#)

**Related reference**

[“/admin/mft/monitor” on page 2219](#)

You can use the HTTP GET method with the `monitor` resource to request information about a Managed File Transfer resource monitor. You can use the HTTP POST method to create a resource monitor, and the HTTP DELETE method to delete a resource monitor.

**GET**

Use the HTTP GET method with the `monitor` resource to list information about the MFT resource monitor status, and other configuration information.

**Note:**

- You must set a coordination queue manager before you can use the `monitor` resource. For more information, see [Configuring the REST API for MFT](#).
- **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

For more information about configuring the MFT REST service, see [Configuring the REST API for MFT](#).

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2236](#)
- [Request body format](#)
- [“Security requirements” on page 2236](#)

- [Response status codes](#)
- [“Response headers” on page 2237](#)
- [Response body format](#)
- [Examples](#)

## Resource URL

`https://host:port/ibmmq/rest/v2/admin/mft/monitor/{monitorName}`

### monitorName

Optionally specifies the name of the monitor to query.

If you do not specify a monitor name, a list of monitors is returned.

If you want to return a list of monitors with a wildcarded monitor name, use the **name** optional query parameter to specify the monitor name instead of specifying the monitor name in the base URL.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring the HTTP and HTTPS ports](#).

## Optional query parameters

### attributes

Specifies a comma-separated list of attributes to retrieve.

If you do not specify **attributes**, the default set of attributes is returned. See [“Response body attributes for list resource monitor” on page 2240](#) for a list of the available attributes.

You cannot request the same attribute multiple times.

You can specify an asterisk, \*, to specify that all attributes are returned.

You can make a request that specifies attributes that are not valid for some of the resource monitor information. However, if you make a request that specifies resource monitor information and includes attributes that are not valid for that information, an error occurs.

You cannot have more than three levels of nesting. For example, you cannot directly query the `transferDefinition.transferSet.postDestCall.retryWait`, only the `transferDefinition.transferSet.postDestCall`. Therefore, when querying the **transferDefinition**, you can query only the following attributes:

#### **transferDefinition**

Returns the complete details of the transfer definition.

#### **transferDefinition.sourceAgent**

Returns the complete details of the **sourceAgent** section of the transfer definition.

#### **transferDefinition.destinationAgent**

Returns the complete details of the **destinationAgent** section of the transfer definition.

#### **transferDefinition.originator**

Returns the complete details of the **originator** section of the transfer definition.

#### **transferDefinition.transferSet**

Returns the complete details of the **transferSet** section of the transfer definition.

#### **transferDefinition.transferSet.item**

Returns the complete details of all transfer items in the **item** section of the transfer definition.

#### **transferDefinition.transferSet.preSourceCall**

Returns the complete details of the **preSourceCall** section of the transfer definition.

#### **transferDefinition.transferSet.postSourceCall**

Returns the complete details of the **postSourceCall** section of the transfer definition.

#### **transferDefinition.transferSet.preDestCall**

Returns the complete details of the **preDestCall** section of the transfer definition.

### **transferDefinition.transferSet.postDestCall**

Returns the complete details of the **postDestCall** section of the transfer definition.

#### **name**

Specifies the name of the resource monitor.

This query parameter is valid only when *monitorName* is not specified in the base resource URL.

By specifying the name of the resource monitor as an optional query parameter instead of in the base URL, you can query a wildcarded resource monitor name, and can combine the query with the **state** and **type** query parameters.

The value can be any string value and \* can be used as a wildcard character. Note that the ? character is not permitted.

#### **agentName**

Name of the agent that owns the resource monitor.

As resource monitors are agent scoped it is possible to have a resource monitor with the same name under more than one agent. In this situation, the REST API returns multiple resource monitor definitions. You can use the **agentName** query parameter to return the resource monitors that are associated with that specific agent.

For example, if a resource monitor with name MONITOR1 exists in more than one agent, the following URL returns more than one resource monitor definition:

```
https://localhost:9443/ibmmq/rest/v1/admin/mft/monitor/MONITOR1
```

Adding the **agentName** query parameter, you can return an agent specific resource monitor:

```
https://localhost:9443/ibmmq/rest/v1/admin/mft/monitor/MONITOR1?agentName=AGENT1
```

The value can be any string value and \* can be used as a wildcard character. Note that the ? character is not permitted.

#### **state**

The status of the resource monitor.

This query parameter is valid only when *monitorName* is not specified in the base resource URL.

The value can be one of the following values:

##### **started**

Only monitors that are in a started state are returned.

##### **stopped**

Only monitors that are in a stopped state are returned.

##### **all**

All monitors, regardless of state, are returned.

The default value is **all**.

#### **type**

The type of the resource monitor.

This query parameter is valid only when *monitorName* is not specified in the base resource URL.

The value can be one of the following values:

##### **directory**

Only directory type monitors are returned.

##### **queue**

Only queue type monitors are returned.

##### **all**

All monitors, regardless of type, are returned.

The default value is **all**.

## Request headers

The following header must be sent with the request:

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## Request body format

None.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MFTWebAdmin, MFTWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

The security principal of the caller must be granted authority to subscribe to the SYSTEM.FTE/Monitor topic.

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

## Response status codes

### 200

Resource monitor information retrieved successfully.

### 400

Invalid data provided.

For example, invalid attributes specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server. For more information, see [“Security requirements” on page 2236](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal is not a member of one or more of the MFTWebAdmin, MFTWebAdminRO, or MQWebUser roles. For more information about the access that is required, see [“Security requirements” on page 2236](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Specified monitor not found.

### 405

Method not allowed.

Returned for any other request apart from GET.

### 500

Server issue or error code from IBM MQ.

### 503

Service unavailable. IBM MQ specific reason code is also returned.

## Response headers

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

## Response body format

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `monitor`.

Each element in the array is a JSON object that represents information about a resource monitor. Each of these JSON objects can contain the following objects and attributes. Which objects and attributes are returned depends on the URL that was specified for the request:

### **name**

String.

Specifies the name of the resource monitor.

### **agentName**

String.

Specifies the name of the agent that runs the resource monitor.

### **type**

String.

Specifies the type of resource monitor:

#### **directory**

The type of the resource to be monitored is the file system directory.

#### **queue**

The type of the resource to be monitored is an IBM MQ queue.

### **state**

String.

Specifies the state of the resource monitor:

#### **started**

The monitor is running.

#### **stopped**

The monitor has stopped.

### **resource**

JSON Object.

Specifies the monitored resource, either a directory or queue.

### **userProperties**

JSON Object.

Specifies a list of user defined custom data in key-value pair of type **String**. For example:

```
"userProperties": {"key1": "value1"}
```

This maps to a metadata attribute in resource monitor definition. An empty array is included in the response, if there are no user properties in the resource monitor configuration.

### **defaultVariables**

JSON Object.

Specifies list of user defined variables and their values in key-value pair of type **String**. Resource monitor uses the values as a "variable substitution" while submitting the transfer request. For example:

```
"defaultVariables": {"groupId": "4F4F4FDEEDF1"}
```

**general**

JSON object.

Specifies other high-level attributes of the resource monitor.

**triggerCondition**

JSON object.

Specifies details of a trigger condition that is used by a resource monitor.

**triggerFileContentFormat**

JSON object.

Specifies a list of files that are transferred when a trigger condition is satisfied.

**transferDefinition**

JSON object.

Specifies details of a list of files to be transferred when a resource monitor trigger condition is satisfied.

This object includes the following nested objects:

**job**

JSON object.

Contains the user-defined job name for the transfer.

**sourceAgent**

JSON object.

Contains attributes that are related to the agent on the destination system.

**destinationAgent**

JSON object.

Contains attributes that are related to the agent on the destination system.

**originator**

JSON object.

Contains attributes that are related to the originator of the request.

**transferSet**

JSON object.

Contains attributes that are related to the group of file transfers.

For more information, see [“ Response body attributes for list resource monitor” on page 2240.](#)

If an error occurs, see [REST API error handling.](#)

**Examples**

The following example returns a default set of data for all resource monitors.

The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/monitor
```

The following JSON response is returned:

```
{ "monitor": [
  {
    "name": "DIRMONWILDCARD",
    "agentName": "SRCWILDCARD",
    "type": "directory",
    "state": "started",
    "resource": {
      "name": "C:\\MFT"
    }
  },
  {
    "name": "DIRMONREGEX",
    "agentName": "SRCDIRREG",
    "type": "directory",
```

```

    "state": "started",
    "resource": {
      "name": "C:\\MFT"
    }
  },
  {
    "name": "DIRMONREGEXFILESIZECHANGE",
    "agentName": "SRCDIR",
    "type": "directory",
    "state": "started",
    "resource": {
      "name": "C:\\MFT"
    }
  }
}
]
}

```

The following example lists the default attributes for a specified resource monitor whose name is DIRMONWILDCARD. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/monitor/DIRMONWILDCARD
```

The following JSON response is returned:

```

{"monitor": [
  {
    "name": "DIRMONWILDCARD",
    "agentName": "SRCWILDCARD",
    "type": "directory",
    "state": "started",
    "resource": {
      "name": "C:\\MFT"
    }
  }
]}

```

The following example lists the default attributes for all the resource monitors whose names begin with DIR. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/monitor?name=DIR*
```

The following JSON response is returned:

```

{"monitor": [
  {
    "name": "DIRMONWILDCARD",
    "agentName": "SRCWILDCARD",
    "type": "directory",
    "state": "started",
    "resource": {
      "name": "C:\\MFT"
    }
  },
  {
    "name": "DIRMONREGEX",
    "agentName": "SRCDIRREG",
    "type": "directory",
    "state": "started",
    "resource": {
      "name": "C:\\MFT"
    }
  },
  {
    "name": "DIRMONREGEXFILESIZECHANGE",
    "agentName": "SRCDIR",
    "type": "directory",
    "state": "started",
    "resource": {
      "name": "C:\\MFT"
    }
  }
]}

```

The following example lists details for all the resource monitors whose type is directory and the state is stopped. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/monitor?type=directory&state=stopped
```

```

{"monitor": [
  {
    "name": "TRIGCONTENTSCSTM",
    "type": "directory",
    "state": "stopped",
    "agentName": "TRIGCONTSCSTM",
    "resource": {
      "name": "C:\\MFT"
    }
  }
]}

```

### Related reference

[“Response body attributes for list resource monitor” on page 2240](#)

When you use the HTTP GET method with the monitor object to request information about resource monitors, the following attributes are returned within named JSON objects.

#### *Response body attributes for list resource monitor*

When you use the HTTP GET method with the monitor object to request information about resource monitors, the following attributes are returned within named JSON objects.

The following objects are available:

- [“general” on page 2240](#)
- [“resource” on page 2241](#)
- [“transferDefinition” on page 2241](#)
- [“triggerCondition” on page 2248](#)
- [“triggerFileContentFormat” on page 2249](#)

#### **name**

String.

The unique name of the resource monitor

#### **Type**

String.

Type of the resource monitor

The value is one of the following values:

#### **directory**

The type of the resource to be monitored is file system directory.

#### **queue**

The type of the resource to be monitored is an IBM MQ queue.

#### **agentName**

String.

Name of the agent that owns the resource monitor.

#### **State**

String.

State of the monitor.

The value is one of the following values:

#### **started**

The monitor is running.

#### **stopped**

The monitor is stopped.

### **general**

Group element that defines the other high level attributes of monitor.



**pollInterval**

Integer.

Frequency at which a monitor polls a resource. The unit of time that is used for this value is specified in the **pollIntervalUnit** attribute.

**pollIntervalUnit**

String.

Specifies the time interval for the **pollInterval** attribute. Possible values are seconds, minutes, hours, days.

**matchesPerTask**

Integer.

Maximum number of trigger matches to include in a single task.

**resource**

Group element that defines the monitored resource.

The **name** attribute in this object is always returned.

**name**

String.

Specifies either the absolute path of a file system directory, or a queue name.

This attribute is always returned.

**recursionLevel**

Integer.

Specifies the number of subdirectories to search to find a matching trigger file. This attribute is valid, only for a directory type of resource monitor.

**transferDefinition**

A list of items that are transferred when a trigger event fires. There is at least one item in the response.

**destinationAgent**

Group element that contains elements that define a destination agent.

The **name** and the **qmgrName** attributes in this object are always returned.

**qmgrName**

String.

The name of the queue manager on the destination system.

**name**

String.

The name of the agent on the destination system.

**job**

A group that contains the following attribute, which is always returned:

**name**

String.

User defined job name for the transfer.

**originator**

Group element that contains the elements that specify the originator of the transfer request.

The **host** and **userid** attributes in this object are always returned.

**host**

String.

The host name of the system where the source file is located.

**userid**

String.

The user ID that originated the file transfer.

**sourceAgent**

Group element that contains the elements that define a source agent.

The **name** and the **qmgrName** attributes in this object are always returned.

**qmgrName**

String.

The name of the queue manager on the source system.

**name**

String.

The name of the agent on the source system.

**transferSet**

Group element that contains the elements that specify a transfer request.

The **item** attribute in this object is always returned.

**priority**

Integer (optional).

Priority assigned to the transfer request. If no value is set, the default is zero.

**userProperties**

Object (optional).

User defined properties specified in the transfer request.

**item**

Object.

Array of group elements that describe the source and destination item configuration to transfer.

**source**

Object.

Group element that contains the attributes of a source item.

The **name** and **type** attributes in this object are always returned.

**name**

String.

Specifies the absolute path of the file, directory, data set, partitioned data set, or queue at source end.

**type**

String.

The type of source. The value is one of the following values:

**queue**

Specifies an IBM MQ queue as the source.

**file**

Specifies a file as the source.

**directory**

Specifies a directory as the source.

**sequentialDataset**

Specifies a z/OS sequential data set as the source.

**partitionedDataset**

Specifies a z/OS partitioned data set as the source.

**recursive**

Boolean (optional).

Specifies that files are transferred recursively in subdirectories when the source element is a directory, or contains wildcard characters.

**disposition**

String (optional).

Specifies the action that is taken on the source element when a source has successfully been transferred to its destination. The value of disposition is one of the following values:

**leave**

The source files are left unchanged.

**delete**

The source files are deleted from the source system after the source file is successfully transferred.

**encoding**

String (optional)

Specifies which character encoding to use to read the source file when performing character conversion. This option is only applicable to text files and the possible value is any valid code page number.

**datasetExtended**

Object (optional).

Group element that defines additional attributes of the source specification, if the source is a z/OS data set in a transfer request.

The **hexDelimiters** and **delimiterPosition** attributes in this object are always returned.

**keepTrailingSpaces**

Boolean (optional).

Describes the action that is taken if there are trailing spaces in the source records that are read from a fixed-length-format record-oriented file (for example, a z/OS data set) as part of a text mode transfer.

**hexDelimiters**

String.

For source files that are record oriented (for example, z/OS data sets), specifies one or more byte values to insert as the delimiter when appending records into a binary file. Each value is represented as two hexadecimal digits in the range 00-FF, prefixed by x.

**delimiterPosition**

String.

Specifies the position of insertion for source text and binary delimiters. The value is one of the following values:

**prefix**

The delimiters are inserted at the start of each record.

**postfix**

The delimiters are inserted at the end of each record. This option is the default option.

**queueExtended**

Object (optional).

Group element that defines additional attributes of a source specification, if the source is an IBM MQ queue in a transfer request.

The **useMessageGroup** and **groupID** attributes in this object are always returned.

**useMessageGroup**

Boolean.

Specifies that the messages are grouped by IBM MQ group ID. The first complete group is written to the destination file. If this parameter is not specified, all messages on the source queue are written to the destination file.

**groupID**

String.

Group ID to be used when getting messages from a queue.

**textDelimiters**

String (optional).

Specifies a sequence of text to insert as the delimiter, when appending multiple messages to a text file.

**hexDelimiters**

String (optional).

Comma separated string of hexadecimal bytes to use, when appending multiple messages to a file. For example x12 or x03 , x7F.

**delimiterPosition**

String (optional).

Defines where the delimiters are positioned in the message being put to the source queue. Possible values are:

**prefix**

Before the beginning of the message body.

**postfix**

After the end of the message body; this is the default option.

**messageArrivalWaitTime**

Integer.

Time in seconds, to wait for arrival of messages in the source queue.

**destination**

Object.

Group element that contains the attributes of a destination item.

The **name** and **type** attributes in this object are always returned.

**name**

String.

Specifies the absolute path of the file, directory, data set, partitioned data set, or queue at the destination end.

**type**

String.

The type of destination. The value is one of the following values:

**queue**

Specifies an IBM MQ queue as the destination.

**file**

Specifies a file as the destination.

**directory**

Specifies a directory as the destination.

**sequentialDataset**

Specifies a z/OS sequential data set as the destination.

**partitionedDataset**

Specifies a z/OS partitioned data set as the destination.

**actionIfExists**

String(optional).

Specifies the action that is taken if a destination file exists on the destination system. The value is one of the following values:

**error**

Reports an error and the file is not transferred; this is the default value.

**overwrite**

Overwrites the existing destination file.

**encoding**

String (optional).

Specifies which character encoding to use to read the source file when performing character conversion. This option is only applicable to text files and the possible value is any valid code page number.

**endOfLine**

String (optional).

Specifies the end-of-line characters that are used when the file is written at the destination. This option is applicable to text files only.

**userId**

String (optional).

The name of the user whose destination file space the files are transferred into.

**datasetExtended**

Object (optional).

Group element that defines additional attributes of the destination specification, if the destination is a z/OS data set in a transfer request.

**truncateRecords**

Boolean.

Specifies that destination records longer than the LRECL data set attribute are truncated. If this parameter is not specified, the records are wrapped. This parameter is valid only for text mode transfers where the destination is a data set.

**queueExtended**

Object (optional).

Group element that defines additional attributes of a destination specification, if the destination is an IBM MQ queue in a transfer request.

The **messagePersistence** and **delimiterType** attributes in this object are always returned.

**messagePersistence**

String.

Defines if the message put to the destination queue is persistent or non-persistent. The value is one of the following values:

**persistent**

Messages are persistent.

**nonPersistent**

Messages are non-persistent.

**asQueueDefault**

Message persistency is set, depending on the queue definition.

**delimiterType**

String.

Defines the type of delimiter to use when splitting incoming data into messages. The value is one of the following values:

**size**

Split based on given size.

**binary**

Split based on given delimiters.

**hexDelimiters**

String (optional).

Comma separated string of hexadecimal bytes to use when splitting messages. For example x12 or x03 , x7F.

**textDelimiters**

String (optional).

Specifies the Java regular expression to use when splitting a text file into multiple messages.

**includeDelimitersInMessage**

Boolean.

Defines whether delimiters are included in a message being put to the destination queue.

**delimiterPosition**

String.

Defines where the delimiters are positioned in the message being put to the destination queue. The value is one of the following values:

**prefix**

Before the beginning of the message body.

**postfix**

After the end of the message body; this is the default option.

**setMQProperties**

Boolean (optional).

Valid only when the destination is a queue. The value is one of the following values:

**true**

Sets message properties on the first message that is created by the transfer.

**false**

Does not set message properties on the first message that is created by the transfer.

**messageSize**

Integer.

Defines a size in bytes to split the incoming data into the message.

**checksum**

String (optional).

Checksum method for verifying data integrity. The value is one of the following values:

**md5**

MD5 algorithm used for integrity validation.

**none**

No checksum validation.

**mode**

String (optional).

Specifies the transfer mode as either binary or text. The value is one of the following values:

**text**

Data is transferred as text.

**binary**

Data is transferred in binary.

**recoveryTimeout**

Integer (optional).

Time in seconds to wait for a transfer to recover.

-1 is the default value.

**preSourceCall**

Object (optional).

Group elements that contain the elements for program invocation before a transfer begins at the source.

These group elements are not present if a resource monitor is not configured to use any program invocation.

**type**

String (optional).

Defines the type of the program to be invoked. The value is one of the following values:

**executable**

This value is the default value.

Defines attributes for a platform specific executable program:

**name**

String.

Name of the program to process.

**arguments**

String (optional).

Argument or arguments to be passed to the program being invoked.

**antScript**

Defines attributes for Ant Script:

**name**

String.

Name of the Ant script to process.

**target**

String (optional)

Target to invoke in the specified Ant script. Attribute is not present in the JSON response, if the default target is to be invoked.

**arguments**

String (optional).

A list of user defined custom data in space separated key=value pair of type **String**. For example:

```
"arguments": "coffeeType=Arabica teaChoice=lemon"
```

**jcl**

Defines attributes for z/OS JCL to submit.

**name**

String.

Name of the JCL to submit.

**retryCount**

Integer (optional).

A positive number of attempts to run the command before ceasing.

**retryWait**

Integer (optional).

Amount of time to wait, in seconds, between two retry attempts.

**successReturnCode**

String (optional).

Reason code that is returned when transfer is complete. This is looked for before running the specified program, script, or JCL. This return code is a combination of an operator and value in the form of "[>|<|!] value". Note that it is valid to have a combination of more than one operator, for example ">= 40".

**postSourceCall**

Object (optional).

Group elements that contain the elements for program invocation after a transfer completes at source. This object contains the same elements as **preSourceCall**.

**preDestinationCall**

Object (optional).

Group elements that contain the elements for program invocation before a transfer begins at the destination. This object contains the same elements as **preSourceCall**.

**postDestinationCall**

Object (optional).

Group elements that contain the elements for program invocation after a transfer completes at the destination. This object contains the same elements as **preSourceCall**.

**triggerCondition**

Group element that defines details of a trigger condition used by a resource monitor.

**type**

String.

Indicates the type of matching done, to decide on triggering a transfer. The value is one of the following values:

For resource type **Directory**:

**matchAll**

Must match the value specified for the **includePattern** and **excludePattern** attributes.

**matchNone**

None of the files in the monitored directory match the value specified for the **includePattern** and **excludePattern** attributes.

**noChangeInSize**

Initiate a transfer, if the size of the file being monitored does not change for a specified number of poll intervals.

**sizeGreaterOrEqualTo**

Initiate a transfer, if the size of the file being monitored is greater than or equal to a specified size.

For resource type **Queue**:

**queueNotEmpty**

Queue must have at least one message.

**completeMessageGroups**

Queue must have at least one group of messages.

**noFileSizeChangePollCount**

Integer.

Refers to the number of polling intervals during which the size of the monitored file does not change. Used in conjunction with the **noChangeInSize** attribute



**fileSize**

Integer.

Refers to the size of the trigger file being monitored, whose size is equal to or greater. Used in conjunction with the **sizeGreaterOrEqualTo** attribute.

**fileSizeUnit**

String.

Defines the unit for the **fileSize** attribute. The value is one of the following values:

**bytes**

File size unit is in bytes

**kilobytes**

File size unit is in kilobytes

**megabytes**

File size unit is in megabytes

**gigabytes**

File size unit is in gigabytes

**includePattern**

String.

A pattern of the name, or names, of files to be included, while doing match for a trigger condition.

**excludePattern**

String.

A pattern of the name, or names, of files to be excluded, while doing match for a trigger condition.

**matchPattern**

String.

Indicates how to interpret the contents of the **includePattern** and **excludePattern** attributes. The value is one of the following values:

**wildcard**

- Indicates the **includePattern** and **excludePattern** attributes contain wildcard characters, for example, \*.

**regularExpression**

Indicates the **includePattern** and **excludePattern** attributes contain Java regular expressions.

**triggerFileContentFormat**

A trigger file defines a list of files transferred when a trigger condition is satisfied. The trigger file might define only the source path, or both source and destination paths. Each line in a trigger file points to a file to transferred.

This object is valid only for **triggerCondition.type** of **matchAll** and **noChangeInSize**.

**groupOrder**

String.

The trigger file contains names of source file names, destination file names, or both. This attribute defines the order of the source file names, destination file names, or both. The value is one of the following values:

**sourceDestination**

Source file name appears first, followed by destination file name.

**destinationSource**

Destination file name appears first followed by source file name.

**customPattern**

String (optional).

A Java regular expression to apply, while parsing trigger file contents for generating a list of files to transfer.

### Related tasks

[Getting started with the REST API for MFT](#)

### Related reference

["/admin/mft/monitor" on page 2219](#)

You can use the HTTP GET method with the `monitor` resource to request information about a Managed File Transfer resource monitor. You can use the HTTP POST method to create a resource monitor, and the HTTP DELETE method to delete a resource monitor.

### DELETE

Use the HTTP DELETE method with the `monitor` resource to delete an existing monitor, or delete the history of an existing monitor.

#### Note:

- Set the command queue manger in the configuration before issuing any MFT Create Monitor or Transfer REST API command. See [Configuring the REST API for MFT](#) for more information.
- **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).
- [Resource URL](#)
- ["Request headers" on page 2250](#)
- [Request body format](#)
- ["Security requirements" on page 2251](#)
- [Response status codes](#)
- ["Response headers" on page 2252](#)
- [Response body format](#)

### Resource URL

To delete an existing monitor:

```
https://host:portibmmq/rest/v2/admin/mft/monitor/  
{monitor name}?agent=<agentName>&agentQmgr=<QmgrName>
```

To delete the history of an existing monitor:

```
https://host:portibmmq/rest/v2/admin/mft/monitor/  
{monitor name}/history?agent=<agentName>&agentQmgr=<QmgrName>
```

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring the HTTP and HTTPS ports](#).

### Request headers

The following headers must be sent with the request:

#### Content-Type

This header must be sent with a value of `application/json` optionally followed by `; charset=UTF-8`.

#### ibm-mq-rest-csrf-token

This header must be set, but the value can be anything, including being blank.

#### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## Request body format

None.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of the MFTWebAdmin or MFTWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

If you have set up a [user sandbox](#), and MFT authority checking, or MFT authority checking, is turned on, you need to grant an additional authority for the user that started the WebSphere Liberty server to access the specified file system location.

For the MFTWebAdmin role, transfer requests are submitted under the context of the user that started the Liberty server. To distinguish between different principals of the MFTWebAdmin role, and for audit purposes, the transfer request submitted contains the name of the authenticated user as the transfer originator. This method ensures that there is a record of who initiated the transfer request.

For example, if the user mftadminusr, of the MFTWebAdmin role, initiates a transfer, the originator data in the XML has mftadminusr in the userID element, as shown in this example:

```
<originator>
  <hostName>example.com.</hostName>
  <userID>mftadminusr</userID>
</originator>
```

If the caller is a member of the MQWebUser role, the security principal of the caller must be granted one of the following authorities:

1. If the command queue is local, that is, the command queue manager and source agent queue manager are the same, grant put authority to the command queue.
2. If the command queue is remote, that is, the command queue manager and source agent queue manager are different, grant put authority to the transmission queue.

### Notes:

- If the user ID of a principal that is a member of the MQWebUser role is longer than 12 characters, the request fails. Response status code 403 is returned to the caller.
- If the caller is assigned more than one role, the highest privilege role that is applicable to the operation is used.

If security is disabled on the mqweb server, the transfer request submitted contains the name "UNAUTHENTICATED" as the transfer originator.

## Response status codes

### 202

The delete request has been accepted by the REST API. It might still get rejected by the MFT agent.

### 400

Invalid or unknown data provided to delete resource monitor.

For example, invalid attributes specified.

### 401

Not authenticated.

The user must be authenticated to the mqweb server. For more information, see [“Security requirements” on page 2251](#).

The `ibm-mq-rest-csrf-token` header must also be specified.

#### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ or MFT resources.
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

#### 500

Server issue, or error code from IBM MQ or MFT.

### Response headers

The following header is returned with the response:

#### location

If the request was successful, this header specifies the URL with the details of the deleted resource monitor. `https://host:port/ibmmq/rest/v1/admin/mft/monitor/{monitorName}` .

### Response body format

The response body is empty if the deletion is successful.

If an error occurs, the response body contains an error message; see [REST API error handling](#).

#### Related tasks

[Getting started with the REST API for MFT](#)

#### Related reference

[“Request body attributes for MFT resource monitors” on page 2224](#)

The Create Monitor REST API takes the input attributes as JSON objects.

### /admin/mft/transfer

You can use the HTTP GET method with the `transfer` resource to request information about Managed File Transfer transfers. You can use the HTTP POST method to put a transfer request message to the command queue manager, which will be routed to the source agent queue manager.

**Note:** **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

### Overview of the HTTP GET method

As a user you can obtain the transfer details of file transfers that have been initiated.

You can retrieve the transfer details of all the transfers that are initiated using the coordination queue manager defined in the `mqwebuser.xml` and a list of all the transfers initiated by yourself. For example, if you initiated 100 transfers and want to know the status of those transfers, the GET method serves the purpose.

See [“GET” on page 2267](#) for more information.

### Overview of the HTTP POST method

As an administrator, you must create a file transfer as necessary for a particular task, or to schedule a new file transfer. This API facilitates in creating the transfer, and also allows you to request a scheduled file transfer.

You can perform a scheduled file transfer once or repeat the transfer multiple times. You can :

- Schedule a file transfer to occur once, or to occur at regular intervals, for example, every minute.

- Specify the occurrences to stop at a defined time and date, or after a defined number of occurrences.
- Specify that the occurrences continue forever.

See [“POST” on page 2253](#) for more information.

### **Related tasks**

[Getting started with the REST API for MFT](#)

### **Related reference**

[“/admin/mft/agent” on page 2197](#)

You can use the HTTP GET method with the agent resource, to request information about Managed File Transfer agents.

## **POST**

Use the HTTP POST method with the `transfer` resource to create a Managed File Transfer transfer.

### **Note:**

- You must set a command queue manager before you can create a transfer with the `transfer` resource. For more information, see [Configuring the REST API for MFT](#).
- **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).
- [Resource URL](#)
- [“Request headers” on page 2253](#)
- [Request body format](#)
- [“Security requirements” on page 2254](#)
- [Response status codes](#)
- [“Response headers” on page 2255](#)
- [Response body format](#)
- [Examples](#)

## **Resource URL**

`https://host:port/ibmmq/rest/v2/admin/mft/transfer/`

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring the HTTP and HTTPS ports](#).

## **Request headers**

The following headers must be sent with the request:

### **Content-Type**

This header must be sent with a value of `application/json` optionally followed by `; charset=UTF-8`.

### **ibm-mq-rest-csrf-token**

This header must be set, but the value can be anything, including being blank.

### **Authorization**

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## **Request body format**

The request body must be in JSON format in UTF-8 encoding. Attributes marked *required* are mandatory. If you do not provide values for the other parameters in the request body, the default values are used.

The following objects can be included in the request body:

**job**

Contains attributes that are related to the transfer job.

**sourceAgent**

Contains attributes that are related to the source agent. This object is required.

**destinationAgent**

Contains attributes that are related to the destination agent. This object is required.

**scheduleTransfer**

Contains attributes that are related to scheduling a transfer.

**transferSet**

Contains attributes that are related to the transfer.

See “[Request body attributes for transfers with HTTP POST](#)” on page 2258 for a list of all the attributes.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of the MFTWebAdmin or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

If you have MFT authority checking turned on, grant additional authorities as described in [Restricting user authorities on MFT agent actions](#). You need to grant the following additional authorities:

1. If the caller is a member of the MFTWebAdmin role, the user ID which started the mqweb server must be granted authority access to the transfer source location.
2. If the caller is a member of the MQWebUser role, the security principal of the caller must be granted access to the transfer source location.

For the MFTWebAdmin role, transfer requests are submitted under the context of the mqweb server user ID. To distinguish between different principals of the MFTWebAdmin role, and for audit purposes, the transfer request submitted contains the name of the authenticated user as the transfer originator. This method ensures that there is a record of who initiated the transfer request.

For example, if the user mftadminusr, of the MFTWebAdmin role, initiates a transfer, the originator data in the XML that is created to describe the transfer has mftadminusr in the userID element, as shown in this example:

```
<originator>
  <hostName>example.com.</hostName>
  <userID>mftadminusr</userID>
</originator>
```

If the caller is a member of the MQWebUser role, the security principal of the caller must be granted one of the following authorities:

1. If the command queue is local, that is, the command queue manager and source agent queue manager are the same, grant put authority to the command queue.
2. If the command queue is remote, that is, the command queue manager and source agent queue manager are different, grant put authority to the transmission queue.

**Notes:**

- If the user ID of a principal that is a member of the MQWebUser role is longer than 12 characters, the request fails. Response status code 403 is returned to the caller.
- If the caller is assigned more than one role, the highest privilege role that is applicable to the operation is used.

If security is disabled on the mqweb server, the transfer request submitted contains the name "UNAUTHENTICATED" as the transfer originator.

## Response status codes

### 202

The file transfer request has been accepted by the REST API. It might still get rejected by the MFT agent. You should issue a GET command, using the URL from the location response header to ascertain the state of the transfer.

### 400

Invalid data provided.

For example, invalid attributes specified.

### 401

Not authenticated.

The user must be authenticated to the mqweb server. For more information, see [“Security requirements”](#) on page 2254.

The `ibm-mq-rest-csrf-token` header must also be specified.

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ or MFT resources.
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 500

Server issue, or error code from IBM MQ or MFT.

### 503

Queue manager not running.

## Response headers

The following header is returned with the response:

### location

If the request was successful, this header specifies the URL for the new transfer.

## Response body format

The response body is empty if the transfer is created successfully.

If an error occurs, the response body contains an error message; see [REST API error handling](#).

## Examples

- The following example creates a simple file transfer. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/transfer/
```

The following JSON payload is sent:

```
{
  "sourceAgent": {
    "qmgrName": "NYQM",
    "name": "NY.AGENT"
  },
  "destinationAgent": {
    "qmgrName": "WASHQM",
```

```

    "name": "WASH.AGENT"
  },
  "transferSet": {
    "item": [
      {
        "source": {
          "name": "C:\\temp\\src\\test.txt",
          "type": "file"
        },
        "destination": {
          "name": "C:\\temp\\dst\\test.txt",
          "type": "file"
        }
      }
    ]
  }
}

```

- The following example creates a transfer from a file to a queue. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/transfer/
```

The following JSON payload is sent:

```

{
  "job": {
    "name": "TESTJOB",
  },
  "sourceAgent": {
    "name": "WASH.AGENT",
    "qmgrName": "WASHQM"
  },
  "destinationAgent": {
    "name": "NY.AGENT",
    "qmgrName": "NYQMGR"
  },
  "transferSet": {
    "priority": 1,
    "recoveryTimeout": -1,
    "item": [
      {
        "checksum": "md5",
        "mode": "text",
        "destination": {
          "actionIfExists": "error",
          "name": "LQ@NYQMGR",
          "type": "queue",
          "delimiterType": "size",
          "messagePersistence": "persistent",
          "queueExtended": {
            "messageSize": 4,
            "setMQProperties": false
          }
        },
        "source": {
          "disposition": "leave",
          "name": "C:\\temp\\src\\test.txt",
          "recursive": false,
          "type": "file"
        }
      }
    ]
  }
}

```

- The following example creates a transfer from a directory to a directory. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/transfer/
```

The following JSON payload is sent:

```

{
  "job": {
    "name": "TESTJOB",
  }
}

```



```

},
"sourceAgent": {
  "name": "WASH.AGENT",
  "qmgrName": "WASHQM"
},
"destinationAgent": {
  "name": "NY.AGENT",
  "qmgrName": "NYQMGR"
},
"transferSet": {
  "item": [
    {
      "checksum": "md5",
      "destination": {
        "actionIfExists": "error",
        "name": "C:\\temp\\dst",
        "type": "directory"
      },
      "source": {
        "disposition": "leave",
        "name": "C:\\temp\\src",
        "recursive": false,
        "type": "directory"
      }
    }
  ]
}
}
}

```

- The following example creates a transfer from a file to a file, using preSourceCall, postSourceCall, preDestinationCall, and postDestinationCall to invoke programs during the transfer. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/transfer/
```

The following JSON payload is sent:

```

{
  "sourceAgent": {
    "qmgrName": "NYQM",
    "name": "NY.AGENT"
  },
  "destinationAgent": {
    "qmgrName": "WASHQM",
    "name": "WASH.AGENT"
  },
  "transferSet": {
    "item": [
      {
        "source": {
          "name": "C:\\temp\\src\\test.txt",
          "type": "file"
        },
        "destination": {
          "name": "C:\\temp\\dst\\test.txt",
          "type": "file"
        }
      }
    ],
    "userProperties": {
      "ARCHIVE_PATH": "C:\\MFT\\ARCHIVE",
      "REJECT_PATH": "C:\\MFT\\REJECT"
    },
    "postSourceCall": {
      "type": "executable",
      "executable": {
        "name": "posttransfersource.exe",
        "arguments": "postdata1 postdata2"
      }
    },
    "postDestinationCall": {
      "type": "executable",
      "executable": {
        "name": "posttransferdest.exe",
        "arguments": "postdataDest1 postdataDest2"
      }
    },
    "preDestinationCall": {
      "type": "executable",

```

```

    "executable": {
      "name": "pretransferdest.exe"
    },
    "preSourceCall": {
      "type": "executable",
      "executable": {
        "name": "posttransferdest.exe",
        "arguments": "predata1 predata2"
      }
    },
    "priority": 0,
    "recoveryTimeout": 21600
  }
}

```

## Related tasks

[Getting started with the REST API for MFT](#)

## Related reference

“Request body attributes for transfers with HTTP POST” on page 2258

When you create the request body for creating a transfer request with the administrative REST API for MFT, you can specify attributes for the transfer within named JSON objects. A number of objects and attributes are available.

### *Request body attributes for transfers with HTTP POST*

When you create the request body for creating a transfer request with the administrative REST API for MFT, you can specify attributes for the transfer within named JSON objects. A number of objects and attributes are available.

The following objects are available:

- [“job” on page 2258](#)
- [“sourceAgent” on page 2258](#)
- [“destinationAgent” on page 2259](#)
- [“scheduleTransfer” on page 2259](#)
- [“transferSet” on page 2260](#)

## job

The job object can contain the following attributes that relate to the transfer job:

### **name**

String.

Specifies a user-defined job name for the transfer.

## sourceAgent

The sourceAgent object can contain the following attributes that relate to the source agent:

### **name**

String.

Specifies the name of the agent on the source system.

This attribute is required.

### **qmgrName**

String.

Specifies the name of the queue manager on the source system.

This attribute is required.

## destinationAgent

The `destinationAgent` object can contain the following attributes that relate to the destination agent:

### name

String.

Specifies the name of the agent on the destination system.

This attribute is required.

### qmgrName

String.

Specifies the name of the queue manager on the destination system.

This attribute is required.

## scheduleTransfer

The `scheduleTransfer` object can contain the following attributes that relate to a scheduled transfer:

### startTime

String.

Specifies the start time and date for the scheduled transfer in the format `yyyy-MM-ddThh:mm` or `hh:mm`. Specify the time by using the 24-hour clock.

### timeBase

String.

Specifies the time base for the start and end time of the scheduled file transfer.

The value must be one of the following values:

#### admin

The start and end time for the scheduled transfer are based on the time and date of the system where the mqweb server is running.

#### source

The start and end time for the scheduled transfer are based on the time and date of the system where the source agent is located.

#### utc

The start and end time for the scheduled transfer are based on Coordinated Universal Time.

### occurrenceInterval

String.

Specifies the interval that the scheduled transfer occurs at.

Use this attribute in conjunction with the `startTime` and `occurrenceFrequency` attributes.

The value must be one of the following values:

- minutes
- hours
- days
- weeks
- months
- years

### occurrenceFrequency

Integer.

Specifies the frequency of a repeating scheduled transfer. Use this attribute in conjunction with the `startTime` and `occurrenceInterval` attributes.

### occurrenceCount

Integer.

Specifies the number of times that the scheduled transfer will occur.

Use this attribute in conjunction with the `startTime` and `occurrenceInterval` attributes.

This attribute can not be specified with the `endTime` attribute.

**endTime**

String.

Specifies the time and date when a repeating scheduled transfer ends in format `yyyy-MM-ddThh:mm` or `hh:mm`. Specify the time by using the 24-hour clock.

Use this attribute in conjunction with the `startTime` and `occurrenceInterval` attributes.

This attribute can not be specified with the `occurrenceCount` attribute.

**transferSet**

The `transferSet` object can contain the following attributes that relate to the transfer:

**priority**

Integer.

Specifies the priority assigned to the transfer request. The default value is zero.

**userProperties**

JSON object.

Specifies user-defined metadata that is passed to exits run by the agents involved in the transfer.

**item**

JSON array.

An array of JSON objects that describe the source and destination item configurations to transfer.

**source**

JSON object.

A JSON object that contains attributes that relate to the source item to transfer.

**name**

String.

Specifies the absolute path of the file, directory, data set, partitioned data set, or queue at the source end.

This attribute is required.

**type**

String.

Specifies the type of source.

The value must be one of the following values:

**queue**

The source is an IBM MQ.

**file**

The source is a file.

**recursive**

Boolean.

Specifies whether files are transferred recursively in subdirectories when the source element is a directory, or contains wildcard characters.

**disposition**

String.

Specifies the action that is taken on the source element when a source has successfully been transferred to its destination.

The value must be one of the following values:

**leave**

The source files are left unchanged.

**delete**

The source files are deleted from the source system when they have been successfully transferred.

**encoding**

String

Specifies which character encoding to use, to read the source file when performing character conversion. This option is only applicable to text files.

The values can be any valid code page number.

**z/OS datasetExtended**

JSON object.

A JSON object that contains additional source attributes, if the source is a z/OS sequential or partitioned data set.

**keepTrailingSpaces**

Boolean.

Specifies whether trailing spaces are kept in the source records that are read from a fixed-length format record oriented file (for example, a z/OS data set) as part of a text mode transfer.

If you do not specify this parameter, trailing spaces are stripped from source records.

**hexDelimiters**

String.

For source files that are record oriented (for example, z/OS data sets), specifies one or more byte values to insert as the delimiter when appending records into a binary file.

You must specify each value as two hexadecimal digits in the range 00-FF, prefixed by x. Separate multiple bytes with commas.

**delimiterPosition**

String

Specifies the position to insert source record delimiters. This attribute is used in conjunction with the hexDelimiters attribute.

The value must be one of the following values:

**prefix**

The delimiters are inserted at the start of each record.

**postfix**

The delimiters are inserted at the end of each record; this is the default option.

**queueExtended**

JSON object.

A JSON object that contains additional source attributes, if the source is an IBM MQ queue.

**messageGroup**

Boolean.

Specifies whether messages are grouped by IBM MQ group ID. The first complete group is written to the destination file.

If this parameter is not specified, all messages on the source queue are written to the destination file.

**groupID**

String.

Specifies the group ID to be used when getting messages from a queue.

**textDelimiters**

String.

Specifies a sequence of text to insert as the delimiter, when appending multiple messages to a text file.

**hexDelimiters**

String.

Specifies one or more byte values to use, when appending multiple messages to a file.

You must specify each value as two hexadecimal digits in the range 00-FF, prefixed by x. Separate multiple bytes with commas. For example x12 or x03 , x7F.

**delimiterPosition**

String.

Specifies where the delimiters are positioned in the message being put to the source queue.

The value must be one of the following values:

**prefix**

The delimiters are inserted at the start of each message.

**postfix**

The delimiters are inserted at the end of each message; this is the default option.

**messageArrivalWaitTime**

Integer.

Specifies the time in seconds to wait for the arrival of messages on the source queue.

**destination**

JSON object.

A JSON object that contains attributes that relate to the destination item.

**name**

String.

Specifies the absolute path of the file, directory, data set, partitioned data set, or queue at the destination.

This attribute is required.

**type**

String.

Specifies the type of destination.

This attribute is required.

The value must be one of the following values:

**queue**

The destination is an IBM MQ queue.

**file**

The destination is a file.

**directory**

The destination is a directory.

 **sequentialDataset**

The destination is a z/OS sequential data set.

 **partitionedDataset**

The destination is a z/OS partitioned data set.

**actionIfExists**

String.

Specifies the action that is taken if a destination file, directory, or data set exists on the destination system.

The value must be one of the following values:

**error**

An error is reported and the file is not transferred; this is the default value.

**overwrite**

The existing destination file is overwritten.

**encoding**

String.

Specifies which character encoding to use to write the file at the destination. This option is only applicable to text files.

The value can be any valid code page number.

**endOfLine**

String.

Specifies the end-of-line characters that are used when the file is written at the destination. This option is applicable to text files only.

The value must be one of the following values:

**LF**

Line feed.

**CRLF**

Carriage return followed by line feed.

 **datasetExtended**

JSON object.

A JSON object that contains additional destination attributes, if the destination is a z/OS data set.

**truncateRecords**

Boolean.

Specifies whether destination records longer than the data set LRECL attribute are truncated. If this parameter is not specified, the records are wrapped.

This parameter is valid only for text mode transfers where the destination is a data set.

**queueExtended**

JSON object.

A JSON object that contains additional destination attributes, if the destination is an IBM MQ queue.

**messagePersistence**

String.

Specifies the persistence of the message put to the destination queue.

The value must be one of the following values:

**persistent**

Messages are persistent.

**notPersistent**

Messages are not persistent.

**asQueue**

The message persistence is as set in the queue definition. This is the default value.

**delimiterType**

String.

Specifies the type of delimiter to use when splitting a file into multiple messages.

The value must be one of the following values:

**size**

Split based on a specified size.

**binary**

Split based on specified delimiters.

**hexDelimiters**

String.

Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages.

You must specify each value as two hexadecimal digits in the range 00-FF, prefixed by x. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes. For example x12 or x03 , x7F.

**textDelimiters**

String.

Specifies the Java regular expression to use, when splitting a text file into multiple messages.

**includeDelimitersInMessage**

Boolean.

Specifies whether delimiters are inserted in the message put to the destination queue.

**delimiterPosition**

String

Specifies where the delimiters are positioned in the message put to the destination queue.

The value must be one of the following values:

**prefix**

The delimiters are inserted at the beginning of the message body.

**postfix**

The delimiters are inserted at the end of the message body.

**setMQProperties**

Boolean.

Specifies whether message properties are set on the first message that is created by the transfer.

**messageSize**

Integer.

Specifies whether to split the file into multiple fixed-length messages of this size in bytes.

**checksum**

String.

Specifies the checksum method for verifying data integrity.

The value must be one of the following values:

**md5**

The MD5 algorithm is used for integrity validation.

**none**

No checksum validation.

**mode**

String.

Specifies the transfer mode.

The value must be one of the following values:

**text**

Data is transferred as text.



**binary**

Data is transferred in binary.

This is the default value.

**recoveryTimeout**

Integer.

Specifies the length of time during which a source agent attempts to recover a stalled file transfer.

The value must be one of the following values:

**-1**

The agent continues to attempt to recover the stalled transfer until the transfer is complete.

This is the default value.

**0**

The agent stops the file transfer as soon as it enters recovery.

**n**

The agent continues to attempt to recover the stalled transfer for the specified amount of time in seconds.

The value must be in the range 1 - 999,999,999.

**preSourceCall**

JSON object.

A JSON object that contains attributes that are related to the program to invoke at the source agent, before a transfer begins.

**type**

String.

Specifies the type of the program to be invoked.

The value must be one of the following values:

**executable**

A platform-specific executable is invoked. This is the default value.

**antScript**

An Apache Ant script is invoked.



A z/OS JCL job is submitted.

**executable**

JSON object.

A JSON object that can contain attributes related to a platform-specific executable program to be invoked. This object can only be specified when the value of the `type` attribute is `executable`.

**name**

String.

Specifies the name of the program to run.

This attribute is required if the `executable` JSON object is specified.

**arguments**

String.

Specifies arguments to be passed to the program that is invoked.

**antScript**

JSON object.

A JSON object that can contain attributes related to an Apache Ant script to be invoked. This object can only be specified when the value of the `type` attribute is `antScript`.

**name**

String.

Specifies the name of the Ant script to run.

This attribute is required if the `antScript` JSON object is specified.

**target**

Specifies the target to invoke in the specified Ant script.

If this attribute is not specified, the target named `default` is invoked.

**arguments**

String.

Specifies a list of user-defined custom data in space separated `key=value` pairs.

**jcl**

JSON object.

A JSON object that can contain attributes related to a z/OS JCL job to submit. This object can only be specified when the value of the `type` attribute is `jcl`.

**name**

String.

Specifies the name of the JCL to submit.

**retryCount**

Integer.

Specifies the number of attempts to run the command before ceasing.

**retryWait**

Integer.

Specifies the amount of time to wait, in seconds, between retry attempts.

**successReturnCode**

String.

Specifies the condition, based on the return code from the transfer, that must be true in order for the specified program, script, or JCL to be run.

The condition is specified as an operator, followed by a value. Valid characters for the operator are `>`, `<`, `!` and `=`. It is valid to have a combination of more than one operator. For example, `">= 40"`.

The default value is zero.

**postSourceCall**

JSON object.

A JSON object that contains attributes that are related to the program to invoke at the source agent, after a transfer completes.

The attributes that can be specified are the same as for the `preSourceCall` object.

**preDestinationCall**

JSON object.

A JSON object that contains attributes that are related to the program to invoke at the destination agent, before a transfer begins.

The attributes that can be specified are the same as for the `preSourceCall` object.

**postDestinationCall**

JSON object.

A JSON object that contains attributes that are related to the program to invoke at the destination agent, after a transfer completes.

The attributes that can be specified are the same as for the `preSourceCall` object.

**Related tasks**

[Getting started with the REST API for MFT](#)

## GET

Use the HTTP GET method with the `transfer` resource to request information about transfers and transfer status. You can query only the transfers that are initiated after the mqweb server is started.

### Note:

- You must set a coordination queue manager before you can use the `transfer` resource. For more information, see [Configuring the REST API for MFT](#).
- The mqweb server caches information about transfers and returns this information when a request is made. This cache is reset when the mqweb server is restarted. You can see if the server has been restarted by viewing the `console.log` and `messages.log` files, or on z/OS, looking at the output from the started task.
- **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

For more information about configuring the MFT REST service, see [Configuring the REST API for MFT](#).

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2268](#)
- [Request body format](#)
- [“Security requirements” on page 2268](#)
- [Response status codes](#)
- [“Response headers” on page 2269](#)
- [Response body format](#)
- [Examples](#)

## Resource URL

`https://host:port/ibmmq/rest/v2/admin/mft/transfer/{transferID}`

### transferID

Optionally specifies the ID of the transfer to query.

If you do not specify a transfer ID, a list of transfers is returned.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring the HTTP and HTTPS ports](#).

## Optional query parameters

### attributes

Specifies a comma-separated list of attributes to retrieve.

This query parameter is valid only when a transfer ID is specified.

If you do not specify **attributes**, the default set of attributes is returned. See [“Response body attributes for transfers” on page 2271](#) for a list of the available attributes.

You cannot request the same attribute multiple times.

You can specify an asterisk, \*, to specify that all attributes are returned.

You can make a request that specifies attributes that are not valid for some of the transfers. However, if you make a request that specifies a transfer ID and includes attributes that are not valid for that transfer, an error occurs.

### limit

Specifies the maximum number of transfers to retrieve.

This query parameter is valid only when no transfer ID is specified.

For example, if the limit=200, the REST API returns a maximum of 200 transfers.

#### **after**

Specifies a transfer ID. All transfers that are initiated after the specified transfer are retrieved. If you specify **after**, you cannot also specify **before**.

This query parameter is valid only when no transfer ID is specified.

#### **before**

Specifies a transfer ID. All transfers that are initiated before that particular transfer are retrieved. If you specify **before**, you cannot also specify **after**.

This query parameter is valid only when no transfer ID is specified.

## **Request headers**

The following header must be sent with the request:

### **Authorization**

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

## **Request body format**

None.

## **Security requirements**

The caller must be authenticated to the mqweb server and must be a member of one or more of the MFTWebAdmin, MFTWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

The security principal of the caller must be granted authority to subscribe to the SYSTEM.FTE/Transfer topic.

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

## **Response status codes**

### **200**

Transfer information retrieved successfully.

### **400**

Invalid data provided.

For example, invalid attributes specified.

### **401**

Not authenticated.

The caller must be authenticated to the mqweb server. For more information, see [“Security requirements” on page 2268](#).

### **403**

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal is not a member of one or more of the MFTWebAdmin, MFTWebAdminRO, or MQWebUser roles. For more information about the access that is required, see [“Security requirements” on page 2268](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

**404**

A transfer with the specified ID does not exist.

**500**

Server issue or error code from IBM MQ.

**503**

Queue manager not running.

**Response headers****Content-Type**

This header is returned with a value of `application/json; charset=utf-8`.

**ibm-mq-rest-mft-total-transfers**

This header is returned with a value that is the total number of transfers that have details available in the mqweb server cache.

**Response body format**

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `transfer`. Each element in the array is a JSON object that represents information about a transfer. Each of these JSON objects can contain the following objects and attributes. Which objects and attributes are returned depends on the URL that was specified for the request:

**id**

String.

Specifies the unique transfer or transaction ID. The ID can be a maximum of 48 alphanumeric characters.

This attribute is always returned.

**job**

JSON object.

Contains the job name for the transfer.

**userProperties**

JSON object.

Contains additional meta data about the transfer. For example: `"userProperties": {"key1": "value1"}`

**sourceAgent**

JSON object.

Contains attributes that are related to the agent on the source system.

The **name** attribute in this object is always returned.

**destinationAgent**

JSON object.

Contains attributes that are related to the agent on the destination system.

The **name** attribute in this object is always returned.

**originator**

JSON object.

Contains attributes that are related to the originator of the request.

The **host** and **host** attributes in this object are always returned.

**transferSet**

JSON object.

Contains attributes that are related to the group of file transfers.

**status**

JSON object.

Contains attributes that are related to the status of the transfer.

The **state** attribute in this object is always returned.

### statistics

JSON object.

Contains attributes that are related to the statistics of the transfer.

The **startTime**, **numberOfFileFailures**, **numberOfFileSuccesses**, **numberOfFileWarnings**, **numberOfFiles** and **endTime** attributes in this object are always returned.

For more information, see [“ Response body attributes for transfers” on page 2271.](#)

If an error occurs, see [REST API error handling.](#)

## Examples

The following example returns a default set of data in the response.

The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/ibmmq/rest/v2/admin/mft/transfer/414d512050524d465444454d4f312020f5189c5921f22302
```

The following JSON response is returned:

```
{
  "transfer": [{
    "id": "414D512050524D465444454D4F312020F5189C5921F22302",
    "destinationAgent": {
      "name": "AGENT.TRI.BANK"
    },
    "originator": {
      "host": "192.168.99.1",
      "userId": "johndoe"
    },
    "sourceAgent": {
      "name": "TESTAGENT"
    },
    "statistics": {
      "endTime": "2018-01-08T16:22:15.569Z",
      "numberOfFileFailures": 0,
      "numberOfFileSuccesses": 2,
      "numberOfFileWarnings": 0,
      "numberOfFiles": 2,
      "startTime": "2018-01-08T16:22:15.242Z"
    },
    "status": {
      "state": "successful"
    }
  }
}]
}
```

The following example lists all the attributes for the specified transfer ID, on the coordination queue manager. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v2/admin/mft/transfer/414d512050524d465444454d4f312020c5c6705924cf9e02?attributes=*
```

The following JSON response is returned:

```
{
  "transfer": [{
    "id": "414D512050524D465444454D4F312020C5C6705924CF9E02",
    "sourceAgent": {
      "qmgrName": "PRMFTDEM01",
      "name": "AGENT2"
    },
    "destinationAgent": {
```



- [“destinationAgent” on page 2272](#)
- [“originator” on page 2272](#)
- [“sourceAgent” on page 2272](#)
- [“statistics” on page 2272](#)
- [“status” on page 2273](#)
- [“transferSet” on page 2274](#)

## **destinationAgent**

### **name**

String.

Specifies the name of the agent on the destination system.

This attribute is always returned.

### **qmgrName**

String.

Specifies the name of the queue manager on the destination system.

## **originator**

### **host**

String.

Specifies the host name of the system where the source file is located.

This attribute is always returned.

### **mqmdUserId**

String.

Specifies the IBM MQ user ID that was supplied in the message descriptor (MQMD).

### **userID**

String.

Specifies the user ID that originated the file transfer.

This attribute is always returned.

## **sourceAgent**

### **name**

String.

Specifies the name of the agent on the source system.

This attribute is always returned.

### **qmgrName**

String.

Specifies the name of the queue manager on the source system.

## **statistics**

### **endTime**

String.

Specifies the time when the transfer completed. This field is updated only when the transfer is complete. If the transfer is in any other state, then **endTime** is an empty string.

This attribute is always returned.

### **numberOfFileFailures**

Integer.



Specifies the number of files that failed to transfer successfully.

This attribute is always returned.

**numberOfFileSuccesses**

Integer.

Specifies the number of files that successfully transferred.

This attribute is always returned.

**numberOfFileWarnings**

Integer.

Specifies the number of files that generated warnings, but otherwise transferred successfully.

This attribute is always returned.

**numberOfFiles**

Integer.

Specifies the total number of files included in the transfer request. This number includes all the files considered for the transfer operation.

This attribute is always returned.

**retryCount**

Integer.

Specifies the number of times that the transfer went into the recovery state and was retried by the agent.

A transfer can go into a recovery state because the source and destination agents lose communication, either because of an IBM MQ network error, or because the agents are not receiving data or acknowledgment messages for a period. This period is determined by the agent properties:

**transferAckTimeout** and **transferAckTimeoutRetries**.

**startTime**

String.

Specifies the time when the transfer was submitted in UTC format.

This attribute is always returned.

**status**

**description**

String.

Specifies detailed information about the status upon completion, such as whether it was partially successful, successful, or failed.

**lastStatusUpdate**

String.

Specifies the most recent time when the transfer status was captured, in UTC format.

**state**

String.

Specifies the state of the transfer. The value can be one of the following values:

- started
- inProgress
- successful
- failed
- partiallySuccessful
- cancelled
- malformed
- notAuthorized

- deleted
- inProgressWithFailures
- inProgressWithWarnings

This attribute is always returned.

## **transferSet**

### **bytesSent**

Integer.

Specifies the total bytes sent.

### **item**

JSON object.

Contains elements that specify the source and destination file names and locations:

#### **destination**

JSON object.

##### **actionIfExists**

String.

Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:

##### **error**

Reports an error and the file is not transferred.

##### **overwrite**

Overwrites the existing destination file.

#### **checksum**

JSON object.

This object does not appear if a checksum was not performed.

Specifies the type of hash algorithm that generated the message digest to create the digital signature. Managed File Transfer supports Message Digest algorithm 5 (md5) only. The checksum provides a way for you to confirm the integrity of transferred files is intact.

The JSON object includes the following elements:

##### **method**

String.

Specifies the method that is used for generating the checksum.

##### **value**

String.

Specifies the checksum value generated.

#### **dataset**

JSON object.

This object is not returned if the `file` or `queue` object is returned.

Specifies a z/OS dataset with the following elements:

##### **attributes**

String.

Specifies attributes related to the dataset.

##### **name**

String.

Specifies the name of the dataset.

##### **size**

Integer.

Specifies the file size.

## **file**

JSON object.

This object is not returned if the `queue` or `dataset` object is returned.

Specifies information about the file that was transferred in the following elements:

### **encoding**

String.

Specifies the encoding for a text file transfer.

### **endOfLine**

Specifies the end of line marker. This value can be either of the following values:

- LF - line feed character only.
- CRLF - carriage return and line feed character sequence.

### **lastModified**

String.

Specifies the last modified date and time for the file, in UTC format.

### **path**

String.

Specifies the path location of the file.

### **size**

Integer.

Specifies the file size.

## **queue**

JSON object.

This object is not returned if the `file` or `dataset` object is returned.

Specifies information about the queue that messages were transferred to, in the following elements:

### **delimiter**

String.

Specifies the delimiter used.

If **delimiterType** is set to *size*, this element specifies the delimiter size. If

**delimiterType** is set to *binary*, the value is the number of delimiter bytes.

If **delimiter** is an empty string, the field is not set while initiating the transfer.

### **delimiterPosition**

String.

This element is valid only when **delimiterType** is *binary*. The value is one of the following values:

#### **"prefix"**

Before each message.

#### **"postfix"**

After each message.

If **delimiterPosition** is an empty string, the field is not set while initiating the transfer.

### **delimiterType**

String.

Specifies the type of delimiter that is used to split the messages. The value can be one of the following values:

#### **binary**

Split by delimiter bytes.

**size**

Split by size.

If **delimiterType** is an empty string, the field is not set while initiating the transfer.

**includeDelimiterInMessage**

Boolean.

This element is valid only when **delimiterType** is *binary*.

Specifies whether the delimiter is included in the message.

**messageCount**

Integer.

Specifies the number of messages that were written to the queue.

**messageLength**

Integer.

Specifies the length of the message written to the queue.

**messageOrGroupId**

String.

If the transfer request did not specify that the file is split into multiple messages, the value of this attribute is the IBM MQ message ID of the message written to the queue.

If the transfer request specified that the file is split into multiple messages, the value of this attribute is the IBM MQ group ID of the messages written to the queue.

**name**

String.

Specifies the name of the queue and the queue manager, in the following format:

```
queueName@queueManagerName
```

**type**

String.

Specifies the type of destination. The destination is one of the following destinations:

**queue**

Specifies an IBM MQ queue as the destination.

**file**

Specifies a file as the destination.

**dataset**

Specifies a z/OS dataset as the destination.

**mode**

String.

Specifies the transfer mode as either binary or text.

**source**

JSON object.

**checksum**

JSON object.

This object does not appear if a checksum was not performed.

Specifies the type of hash algorithm that generated the message digest to create the digital signature. Managed File Transfer supports Message Digest algorithm 5 (md5) only. The checksum provides a way for you to confirm the integrity of transferred files is intact.

The JSON object includes the following elements:

**method**

String.

Specifies the method that is used to generate the checksum.

**value**

String.

Specifies the checksum value that is generated.

**disposition**

String.

Specifies the action that is taken on the source element when source has successfully been transferred to its destination. This string is one of the following options:

**leave**

Specifies that the source files are left unchanged

**delete**

Specifies that the source files are deleted from the source system after the source file is successfully transferred

**dataset**

JSON object.

This object is not returned if the `file` or `queue` object is returned.

Specifies a z/OS dataset with the following elements:

**attributes**

String.

Specifies attributes related to the dataset.

**name**

String.

Specifies the name of the dataset.

**size**

Integer.

Specifies the file size.

**file**

JSON object.

This object is not returned if the `queue` or `dataset` object is returned.

This object contains the following elements:

**encoding**

String.

Specifies the encoding for a text file transfer.

**endOfLine**

Specifies the end of line marker. This value can be either of the following values:

- LF - line feed character only.
- CRLF - carriage return and line feed character sequence.

**lastModified**

String.

Specifies the last modified date and time for the file, in UTC format.

**path**

String.

Specifies the path location for the file.

**size**

Integer.

Specifies the size of the file.

**queue**

JSON object.

This object is not returned if the `file` or `dataset` object is returned.

Specifies information about the queue that transferred messages were retrieved from, in the following elements:

**messageCount**

Integer.

Specifies the number of messages that were read from the queue.

**name**

String.

Specifies the name of the queue and the queue manager, in the following format:

```
queueName@queueManagerName
```

**setMqProperties**

Boolean.

Specifies whether IBM MQ message properties are set on the first message in a file, and any messages written to the queue when an error occurs.

**type**

String.

Specifies the type of source. The source is one of the following sources:

**queue**

Specifies an IBM MQ queue as the source.

**file**

Specifies a file as the source, if the source is a file or directory.

**dataset**

Specifies a z/OS dataset as the source.

**status**

JSON object.

Specifies the status of a single item in the transfer set. The status object contains the following elements:

**description**

String.

Specifies detailed information about the status completion, such as whether it was partially successful, successful, or failed.

**state**

String.

Specifies the state of the transfer. The value can be one of the following values:

- started
- inProgress
- successful
- failed
- partiallySuccessful
- cancelled
- malformed
- notAuthorized
- deleted
- inProgressWithFailures
- inProgressWithWarnings

## Related tasks

[Getting started with the REST API for MFT](#)

## Related reference

[“/admin/mft/agent” on page 2197](#)

You can use the HTTP GET method with the agent resource, to request information about Managed File Transfer agents.

## **/admin/qmgr/{qmgrName}/channel**

You can use the HTTP GET method with the channel resource to request information about channels.

### Note:

- This resource URL is available only in version 1 of the REST API. To query channels using version 2 of the REST API, use the [“/admin/action/qmgr/{qmgrName}/mqsc” on page 2159](#) resource.
- **V 9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

You can use the administrative REST API gateway with this resource URL.

For more information about the PCF equivalents to the channel REST API parameters and attributes, see [“REST API and PCF equivalents for channels” on page 2390](#).

## **GET**

Use the HTTP GET method with the channel resource to request information about channels.

### Note:

- This resource URL is available only in version 1 of the REST API. To query channels using version 2 of the REST API, use the [“/admin/action/qmgr/{qmgrName}/mqsc” on page 2159](#) resource.
- **V 9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

The information that is returned is similar to the information returned by the [“MQCMD\\_INQUIRE\\_CHANNEL \(Inquire Channel\)” on page 1191](#) and [“MQCMD\\_INQUIRE\\_CHANNEL\\_STATUS \(Inquire Channel Status\)” on page 1216](#) PCF commands, and the [“DISPLAY\\_CHANNEL \(display channel definition\)” on page 661](#) and [“DISPLAY\\_CHSTATUS \(display channel status\)” on page 685](#) MQSC commands.

**Note:** **z/OS** On z/OS, the channel initiator must be running before you use the channel resource with the HTTP GET method specifying the **status** parameter.

**Note:** The REST API supports only the following channels:

- Channels that have a transport type of TCP.
- Sender, receiver, server, requester, cluster-sender, and cluster-receiver channels.

Other channels are not returned.

- [“Resource URL” on page 2280](#)
- [“Optional query parameters” on page 2280](#)
- [“Request headers” on page 2284](#)
- [“Request body format” on page 2284](#)
- [“Security requirements” on page 2284](#)
- [“Response status codes” on page 2285](#)
- [“Response headers” on page 2285](#)
- [Response body format](#)

- [“Examples” on page 2287](#)

## Resource URL

`https://host:port/ibmmq/rest/v1/admin/qmgr/{qmgrName}/channel/{channelName}`

### qmgrName

Specifies the name of the queue manager on which to query the channels.

You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

The queue manager name is case-sensitive.

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.
- A period (.) must be encoded as %2E.

### channelName

Optionally specifies the name of a channel to query. This channel must exist on the specified queue manager.

The channel name is case-sensitive.

If the channel name includes a forward slash or a percent sign, these characters must be URL encoded:

- A forward slash, /, must be encoded as %2F.
- A percent sign, %, must be encoded as %25.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Optional query parameters

**attributes={object,...[\*|object.attributeName,...]}**

### object,...

Specifies a comma-separated list of JSON objects that contain related channel configuration attributes to return.

For example, to return all channel configuration attributes that are related to time stamps, specify `timestamps`. To return all channel configuration attributes that are related to compression and to connection management, specify `compression,connectionManagement`.

The `status` objects cannot be specified with this query parameter. Use the **status** query parameter to return these attributes.

You cannot specify the same object more than once. If you request objects that are not valid for a particular channel, the attributes are not returned for that channel. However, if you specify a value for the **type** parameter that is not `all`, and request objects that are not valid for that channel type, an error is returned.

For a full list of objects and associated attributes, see [Attributes for channels](#).

### \*

Specifies all attributes.

### object.attributeName,...

Specifies a comma-separated list of channel configuration attributes to return.

Each attribute must specify the JSON object that contains the attribute, in the form `object.attributeName`. For example, to return the `keepAliveInterval`



attribute, which is contained in the `connectionManagement` object, specify `connectionManagement.keepAliveInterval`.

Attributes can be nested inside multiple JSON objects, such as `exits.message.name`, which is an attribute inside a message object inside an exits object.

The keyword `[type]` can be used as a wildcard to include multiple channel-type-specific sections that contain the same attribute. For example, `[type].clusterName` is equivalent to `clusterSender.clusterName,clusterReceiver.clusterName`.

Attributes from the status object cannot be specified with this query parameter. Use the **status** query parameter to return these attributes.

You cannot specify the same attribute more than once. If you request attributes that are not valid for a particular channel, the attributes are not returned for that channel. However, if you specify the **type** parameter and request attributes that are not valid for that channel type, an error is returned.

For a full list of attributes and associated objects, see [Attributes for channels](#).

**status={\*|currentStatus|savedStatus|currentStatus.attributeName,savedStatus.attributeName,...}**

**\***

Specifies that all savedStatus and currentStatus attributes are returned.

**currentStatus**

Specifies that all currentStatus attributes are returned.

**savedStatus**

Specifies that all savedStatus attributes are returned.

**currentStatus.attributeName,savedStatus.attributeName,...**

Specifies a comma-separated list of current status and saved status attributes to return.

For example, to return the state attribute, specify `currentStatus.state`.

For a full list of status attributes, see [Current status attributes for channels](#) and [Saved status attributes for channels](#).

**filter=filterValue**

Specifies a filter for the channel definitions that are returned.

If you specify a channel name in the resource URL, you can only filter on status attributes.

If you filter on a current status attribute, the only current status objects returned are those that match the filter parameter. All saved status objects for the corresponding channels are returned, if requested.

If you filter on a saved status attribute, the only saved status objects returned are those that match the filter parameter. All current status objects for the corresponding channels are returned, if requested.

You can specify only one filter. If you filter on a status attribute, you must specify the corresponding **status** query parameter.


*filterValue* has the following format:

```
attribute:operator:value
```

where:

**attribute**

Specifies one of the applicable attributes. For a full list of attributes, see [Attributes for channels](#). The following attributes cannot be specified:

- name
- type
-  queueSharingGroup.disposition
- `[type].connection.port`

- `connectionManagement.localAddress.port`
- `connectionManagement.localAddress.portRange`
- `currentStatus.general.connection.port`
- `currentStatus.connectionManagement.localAddress.port`

The keyword [type] can be used as a wildcard to include multiple channel-type-specific sections that contain the same attribute, such as `sender.connection` and `clusterReceiver.connection`.

To filter on any attributes that are time stamps, the filter can specify any portion of the time stamp, with a trailing asterisk, \*. The format of a time stamp is, YYYY-MM-DDThh:mm:ss. For example, you can specify `2001-11-1*` to filter on dates in the range `2001-11-10` to `2001-11-19`, or `2001-11-12T14:*` to filter any minute in the specified hour of the specified day.

Valid values for the YYYY section of the date are in the range 1900 - 9999.

The time stamp is a string. Therefore, only the `equalTo` and `notEqualTo` operators can be used with the time stamp.

### operator

Specifies one of the following operators:

#### **lessThan**

Use this operator only with integer attributes.

#### **greaterThan**

Use this operator only with integer attributes.

#### **equalTo**

Use this operator with any attribute except string array attributes and integer array attributes.

#### **notEqualTo**

Use this operator with any attribute except string array attributes and integer array attributes.

#### **lessThanOrEqualTo**

Use this operator only with integer attributes.

#### **greaterThanOrEqualTo**

Use this operator only with integer attributes.

#### **contains**

Use this operator only with integer array attributes and string array attributes.

#### **doesNotContain**

Use this operator only with integer array attributes and string array attributes.

### value

Specifies the constant value to test against the attribute.

The value type is determined by the attribute type.

For string and boolean attributes, you can omit the value field after the colon. For string attributes, omit the value to return channels with no value for the specified attribute. For boolean attributes, omit the value to return any channels that have the specified attribute set to false. For example, the following filter returns all channels where the description attribute is not specified:

```
filter=general.description:equalTo:
```

You can use a single asterisk, \*, at the end of the value as a wildcard. You cannot use only an asterisk.

If the value includes a space, a forward slash, a percent sign, or an asterisk that is not a wildcard, these characters must be URL encoded:

- A space must be encoded as %20
- A plus, +, must be encoded as %2B
- A forward slash, /, must be encoded as %2F.

- A percent sign, %, must be encoded as %25.
- An asterisk, \*, must be encoded as %2A.

**name=*name***

This query parameter cannot be used if you specify a channel name in the resource URL.

Specifies a wildcard channel name to filter on.

The *name* specified must include an asterisk, \*, as a wildcard. You can specify one of the following combinations:

**\***

Specifies that all channels are returned.

**prefix\***

Specifies that all channels with the specified prefix in the channel name are returned.

**\*suffix**

Specifies that all channels with the specified suffix in the channel name are returned.

**prefix\*suffix**

Specifies that all channels with the specified prefix and the specified suffix in the channel name are returned.

**type=*type***

Specifies the type of channel to return information about.

The value can be one of the following values:

**all**

Specifies that information about all channels is returned.

**sender**

Specifies that information about sender channels is returned.

**receiver**

Specifies that information about receiver channels is returned.

**server**

Specifies that information about server channels is returned.

**requester**

Specifies that information about requester channels is returned.

**clusterSender**

Specifies that information about cluster sender channels is returned.

**clusterReceiver**

Specifies that information about cluster receiver channels is returned.

The default value is all.

**queueSharingGroupDisposition=*disposition***

 This parameter is only available on z/OS.

Specifies the disposition of the channels for which information is to be returned.

The value can be one of the following values:

**live**

Return channels defined with qmgr or copy disposition.

**all**

Return channels defined with qmgr, copy or group disposition.

**copy**

Return channels defined with copy disposition.

**group**

Return channels defined with group disposition.

**private**

Return channels defined with copy or qmgr disposition.

## qmgr

Return channels defined with qmgr disposition.

The default value is live.

## Request headers

The following headers must be sent with the request:

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### ibm-mq-rest-gateway-qmgr

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## Request body format

None.

## Security requirements


The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

The security principal of the caller must be granted the ability to issue the following PCF commands for the specified queue manager:

- If the **status** query parameter is not specified:
  - For the channel that is specified by the *{channelName}* portion of the resource URL, or for channels that match the specified query parameters, authority to issue the **MQCMD\_INQUIRE\_CHANNEL** PCF command must be granted.
- If the **status** query parameter is specified:
  - For the channel that is specified by the *{channelName}* portion of the resource URL, or for channels that match the specified query parameters, authority to issue the **MQCMD\_INQUIRE\_CHANNEL** PCF command must be granted.
  - For the channel that is specified by the *{channelName}* portion of the resource URL, or for channels that match the specified query parameters, authority to issue the **MQCMD\_INQUIRE\_CHSTATUS** PCF command must be granted.

A principal has display authority if the principal can issue one or both of the **MQCMD\_INQUIRE\_CHANNEL** and **MQCMD\_INQUIRE\_CHSTATUS** PCF commands. If the principal has display authority for only some of the channels that are specified by the resource URL and query parameters, then the array of channels that is returned from the REST request is limited to those channels that the principal has authority to display. No information is returned about channels that cannot be displayed. If the principal does not have display authority for any of the channels that are specified by the resource URL and query parameters, an HTTP status code of 403 is returned.

 On Multiplatforms, if the attribute `currentStatus.monitoring.messagesAvailable` is to be returned, authority to issue the **MQCMD\_INQUIRE\_Q** on the transmission queues used by cluster sender channels is required.

**ALW** On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the **setmqaut** command. For more information, see [setmqaut \(grant or revoke authority\)](#).

**z/OS** On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 200

Channel information retrieved successfully.

### 400

Invalid data provided.

For example, invalid channel attributes specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information, see [“Security requirements” on page 2284](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources. For more information about the access that is required, see [“Security requirements” on page 2284](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Channel does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

The following headers are returned with the response:

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

### ibm-mq-rest-gateway-qmgr

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

## Response body format

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `channel`. Each element in the array is a JSON object that represents information about a channel. Each of these JSON objects contains the following attributes:

### name

String.

Specifies the name of the channel.

This attribute is always returned.

**type**

String.

Specifies the type of channel.

The value is one of the following values:

- `sender`
- `receiver`
- `server`
- `requester`
- `clusterSender`
- `clusterReceiver`

This attribute is always returned.

The following objects can be included in the JSON object that represents information about a channel. Which objects and attributes are returned depends on the URL that was specified for the request:

**sender**

Contains attributes that are related to sender channels.

**server**

Contains attributes that are related to server channels.

**requester**

Contains attributes that are related to requester channels.

**clusterSender**

Contains attributes that are related to cluster sender channels.

**clusterReceiver**

Contains attributes that are related to cluster receiver channels.

**clusterRouting**

Contains attributes that are related to the routing of messages in a cluster.

**connectionManagement**

Contains attributes that are related to connection management including:

- A JSON array of connection objects that are labeled `connectionManagement`, which contain host and port information
- `longRetry` and `shortRetry` objects, containing count and interval attributes

**compression**

Contains attributes that are related to compression

**dataCollection**

Contains attributes that are related to monitoring and statistics

**exits**

Contains exit objects and arrays of exit objects, each containing:

- Exit name attribute
- User data attribute

**extended**

Contains attributes that are related to extended channel properties, such as data conversion, and sequence numbers.

**failedDelivery**

Contains attributes that are related to message delivery failure, such as retry options.

**general**

Contains attributes that are related to general channel properties, such as the description of the channel.

**batch**

Contains attributes that are related to message batches.

**queueSharingGroup**

Contains attributes that are related to queue sharing groups on z/OS.

**receiverSecurity**

Contains attributes that are related to security for receiving channels.

**transmissionSecurity**

Contains attributes that are related to transmission security and encryption.

For more information, see [“Response body attributes for channels” on page 2289](#).

If a damaged object is found, and the REST request did not specify a channel name within the resource URL, an extra JSON array that is called `damaged` is returned. This JSON array contains a list of the objects that are damaged, specifying the object names. If the REST request specifies a channel name within the resource URL, but the object is damaged, an error is returned.

If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

**Examples**

- The following example lists all channels on the queue manager QM1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/channel
```

The following JSON response is returned:

```
{
  "channel": [
    {
      "name": "RECEIVER.CHL",
      "type": "receiver"
    },
    {
      "name": "SENDER.CHL",
      "type": "sender",
      "sender": {
        "connection": [
          {
            "host": "example.com",
            "port": "1414"
          }
        ],
        "transmissionQueueName": "XMIT.Q"
      }
    },
    {
      "name": "SERVER.CHL",
      "type": "server",
      "server": {
        "transmissionQueueName": "XMIT.Q"
      }
    },
    {
      "name": "REQUESTER.CHL",
      "type": "requester",
      "requester": {
        "connection": [
          {
            "host": "example.com",
            "port": 1414
          }
        ]
      }
    },
    {
      "name": "CLUSSDR.CHL",
      "type": "clusterSender",
      "clusterSender": {
        "connection": [
          {
            "host": "example.com",
            "port": 1414
          }
        ],
        "clusterName": "CUSTER1"
      }
    },
    {
      "name": "CLUSRCVR.CHL",
      "type": "clusterReceiver",
    }
  ]
}
```

```

    "clusterReceiver": {
      "connection": [{
        "host": "example.com",
        "port": 1414
      }],
      "clusterName": "CUSTER1"
    }
  }
}

```

- The following example lists all receiver channels on the queue manager QM1, showing their connection retry attempts information. The following URL is used with the HTTP GET method:

```

https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QMGR2/channel?
type=sender&attributes=connectionManagement.shortRetry,connectionManagement.longRetry

```

The following JSON response is returned:

```

{
  "channel":
  [
    {
      "name": "SENDER.CHL",
      "type": "sender",
      "connectionManagement": {
        "longRetry": {
          "count": 999999999,
          "interval": 1200
        },
        "shortRetry": {
          "count": 10,
          "interval": 60
        }
      },
      "sender": {
        "connection": [{
          "host": "example.com",
          "port": 1414
        }],
        "transmissionQueueName": "XMIT.Q"
      }
    },
    {
      "name": "SYSTEM.DEF.SENDER",
      "type": "sender",
      "connectionManagement": {
        "longRetry": {
          "count": 999999999,
          "interval": 1200
        },
        "shortRetry": {
          "count": 10,
          "interval": 60
        }
      },
      "sender": {
        "connection": [],
        "transmissionQueueName": ""
      }
    }
  ]
}

```

- The following example lists some status attributes for the channel CHL1, on channel manager QM1. The following URL is used with the HTTP GET method:

```

https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/channel/CHL1?
status=currentStatus.timestamps,currentStatus.batch.currentMessages,savedStatus.batch.currentM
essages

```

The following JSON response is returned:

```

{
  "channel":
  [
    {
      "name": "CHL1",
      "type": "sender",
      "currentStatus": {
        "inDoubt": false,
        "state": "running",
        "batch": {

```



```

        "currentMessages": 10
      },
      "timestamps": {
        "lastMessage": "2017-10-02T09:17:42.314Z",
        "started": "1993-12-31T23:59:59.000Z"
      }
    }
  ],
  "savedStatus": [
    {
      "inDoubt": false,
      "batch": {
        "currentMessages": 5
      }
    },
    {
      "inDoubt": false,
      "batch": {
        "currentMessages": 7
      }
    }
  ]
}

```

- The following example shows how to get all information, including current status and saved status, for the channel CHL2 on queue manager QM1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/channel/CHL2?attributes=*&status=*
```

- The following example shows how to get all channel configuration and status information for channels that are currently running, for the queue manager QM1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/channel?attributes=*&status=*&filter=currentStatus.state:equalTo:running
```

#### *Response body attributes for channels*

When you receive the response body from using the HTTP verb GET with the channel object to request information about channels, attributes for the channels are returned within named JSON objects.

The following objects are available:

- [“sender” on page 2290](#)
- [“server” on page 2290](#)
- [“requester” on page 2290](#)
- [“clusterSender” on page 2291](#)
- [“clusterReceiver” on page 2291](#)
- [“clusterRouting” on page 2292](#)
- [“connectionManagement” on page 2292](#)
- [“compression” on page 2294](#)
- [“dataCollection” on page 2294](#)
- [“exits” on page 2295](#)
- [“extended” on page 2296](#)
- [“failedDelivery” on page 2297](#)
- [“general” on page 2298](#)
- [“batch” on page 2298](#)
- [“queueSharingGroup” on page 2299](#)
- [“receiverSecurity” on page 2299](#)
- [“transmissionSecurity” on page 2300](#)
- [“currentStatus” on page 2300](#)
- [“savedStatus” on page 2310](#)

For more information about the PCF equivalents to the queue REST API parameters and attributes, see [“REST API and PCF equivalents for channels” on page 2390](#).

**Note:** The REST API supports only channels that have TCP as their transport type, and are of type sender, receiver, server, requester, cluster-sender, or cluster-receiver. Other channels are not returned.

## sender

The `sender` object contains information about sender channels and is returned only for sender channels:

### connection

An array of JSON objects that can contain the following attributes that define the channel connection:

#### host

String.

Specifies the host that this channel connects to.

#### port

Integer.

Specifies the port that this channel uses on this host.

This attribute cannot be used to filter results.

These attributes are always returned if they are available. If no connection information is available, an empty array is returned. If the connection does not conform to the expected syntax, an array containing a single host attribute having the value of the entire connection is returned.

### transmissionQueueName

String.

Specifies the name of the transmission queue in use by this channel.

This attribute is always returned.

## server

The `server` object contains information about server channels and is returned only for server channels:

### connection

An array of JSON objects that can contain the following attributes that define the channel connection:

#### host

String.

Specifies the host that this channel connects to.

#### port

Integer.

Specifies the port that this channel uses on this host.

This attribute cannot be used to filter results.

These attributes are always returned if they are available. If no connection information is available, an empty array is returned. If the connection does not conform to the expected syntax, an array containing a single host attribute having the value of the entire connection is returned.

### transmissionQueueName

String.

Specifies the name of the transmission queue in use by this channel.

This attribute is always returned.

## requester

The `requester` object contains information about requester channels and is returned only for requester channels:

**connection**

An array of JSON objects that can contain the following attributes that define the channel connection:

**host**

String.

Specifies the host that this channel connects to.

**port**

Integer.

Specifies the port that this channel uses on this host.

This attribute cannot be used to filter results.

If no connection information is available, an empty array is returned.

If the connection does not conform to the expected syntax, an array containing a single host attribute having the value of the entire connection is returned.

**clusterSender**

The `clusterSender` object contains information about cluster sender channels and is returned only for cluster sender channels:

**connection**

An array of JSON objects that can contain the following attributes that define the channel connections:

**host**

String.

Specifies the host that this channel connects to.

**port**

Integer.

Specifies the port that this channel uses on this host.

This attribute cannot be used to filter results.

These attributes are always returned if they are not empty. If no connection information is available, an empty array is returned.

If the connection does not conform to the expected syntax, an array containing a single host attribute having the value of the entire connection is returned.

**clusterName**

String.

Specifies the name of the cluster to which the channel belongs.

This attribute is always returned if it is not empty.

**clusterNameList**

String.

Specifies a list of clusters to which the channel belongs.

This attribute is always returned if it is not empty.

**clusterReceiver**

The `clusterReceiver` object contains information about cluster receiver channels and is returned only for cluster receiver channels:

**connection**

An array of JSON objects that can contain the following attributes that define the channel connections:

**host**

String.

Specifies the host that this channel connects to.

**port**

Integer.

Specifies the port that this channel uses on this host.

This attribute cannot be used to filter results.

These attributes are always returned if they are not empty. If no connection information is available, an empty array is returned.

If the connection does not conform to the expected syntax, an array containing a single host attribute having the value of the entire connection is returned.

**clusterName**

String.

Specifies the name of the cluster to which the channel belongs.

This attribute is always returned if it is not empty.

**clusterNameList**

String.

Specifies a list of clusters to which the channel belongs.

This attribute is always returned if it is not empty.

**clusterRouting**

The `clusterRouting` object contains information about routing within clusters and is returned only for cluster receiver and cluster sender channels:

**workloadPriority**

Integer.

Specifies the channel priority for cluster workload distribution.

A value of 0 specifies the lowest priority and a value of 9 specifies the highest priority.

**workloadRank**

Integer.

Specifies the channel rank for cluster workload distribution.

A value of 0 specifies the lowest rank and a value of 9 specifies the highest rank.

**workloadWeight**

Integer.

Specifies channel weighting for cluster workload distribution.

A value of 1 specifies the lowest weight and a value of 99 specifies the highest weight.

**networkPriority**

Integer.

Specifies priority for the network connection. If there are multiple paths available, distributed queuing selects the path with the highest priority.

A value of 0 specifies the lowest priority and a value of 9 specifies the highest priority.

**connectionManagement**

The `connectionManagement` object contains information about connection management:

**heartbeatInterval**

Integer.

Specifies the time, in seconds, between heartbeat flows that are passed from the sending MCA when there are no messages on the transmission queue. This interval gives the receiving MCA the opportunity to quiesce the channel.

**disconnectInterval**

Integer.

Specifies the maximum number of seconds that the channel waits for messages to be put on a transmission queue before the channel ends.

A value of zero causes the message channel agent to wait indefinitely.

**keepAliveInterval**

Integer.

Specifies the value that is passed to the communications stack for KeepAlive timing for the channel.

**localAddress**

An array of JSON objects that can contain the following attributes that define the local communications address of the channel:

**host**

String.

Specifies the local IP address or host name.

This value is returned if the local address in the channel definition contains a host name or IP address.

**port**

Integer.

Specifies the local port number.

This value is returned if the local address in the channel definition contains a port number.

This attribute cannot be used to filter results.

**portRange**

JSON object that contains a range of local ports:

**low**

Integer.

Specifies the start of the port range.

**high**

Integer.

Specifies the end of the port range.

Returned if a port range is specified in the local address in the channel definition.

This attribute cannot be used to filter results.

If no local address information is available, an empty array is returned.

If the local address does not conform to the expected syntax, an array containing a single host attribute having the value of the entire local address is returned.

**shortRetry**

JSON object.

Specifies the maximum number and interval of attempts that are made to establish a connection to the remote machine before the `longRetry.count` and `longRetry.interval` are used:

**count**

Integer.

Specifies the maximum number of attempts to connect to the remote machine.

**interval**

Integer.

Specifies the interval in seconds between attempts to connect to the remote machine.

**longRetry**

JSON object.

Specifies the maximum number of attempts and interval of attempts that are made to establish a connection to the remote machine after the count by `shortRetry.count` is exhausted:

**count**

Integer.

Specifies the maximum number of attempts to connect to the remote machine.

**interval**

Integer.

Specifies the interval in seconds between attempts to connect to the remote machine.

## compression

The `compression` object contains attributes that are related to data compression:

**header**

String array.

Specifies the header data compression techniques that are supported by the channel. The values that are returned are in order of preference.

The value is one of the following values:

**none**

Specifies that no header data compression is performed.

**system**

Specifies that header data compression is performed.

**message**

String array.

Specifies the message data compression techniques that are supported by the channel. The values that are returned are in order of preference.

The value is one of the following values:

**none**

Specifies that no header data compression is performed.

**runLengthEncoding**

Specifies that message data compression is performed by using run-length encoding.

**zlibFast**

Specifies that message data compression is performed by using ZLIB encoding with speed prioritized.

**zlibHigh**

Specifies that message data compression is performed by using ZLIB encoding with compression prioritized.

**any**

Specifies that any compression technique that is supported by the queue manager can be used.

This value is only valid for channels of type receiver and requester.

## dataCollection

The `dataCollection` object contains attributes that are related to data collection, monitoring, and statistics:

**monitoring**

String.

Specifies whether online monitoring data is collected, and if so, the rate at which the data is collected.

The value is one of the following values:

**off**

Specifies that online monitoring data is not collected for the channel.

**asQmgr**

Specifies that the queue inherits the value from the queue manager MONCHL MQSC parameter.

**low**

Specifies that online monitoring data is collected for the channel if the MONCHL MQSC parameter on the queue manager is not set to none. The rate of data collection is low.

**medium**

Specifies that online monitoring data is collected for the channel if the MONCHL MQSC parameter on the queue manager is not set to none. The rate of data collection is moderate.

**high**

Specifies that online monitoring data is collected for the channel if the MONCHL MQSC parameter on the queue manager is not set to none. The rate of data collection is high.

**statistics**

String.

Specifies whether statistics data is collected for the channel.

The value is one of the following values:

**off**

Specifies that statistics data is not collected for the channel.

**asQmgr**

Specifies that the channel inherits the value from the queue manager STATCHL MQSC parameter.

**low**

Specifies that statistics data is collected for the channel if the STATCHL MQSC parameter on the channel manager is not set to none. The rate of data collection is low.

**medium**

Specifies that statistics data is collected for the channel if the STATCHL MQSC parameter on the channel manager is not set to none. The rate of data collection is moderate.

**high**

Specifies that statistics data is collected for the channel if the STATCHL MQSC parameter on the channel manager is not set to none. The rate of data collection is high.

**exits**

The `exits` object contains information about channel exits:

**message**

An array of JSON objects that contain the following attributes that define the channel message exits:

**name**

String.

Specifies the message exit name.

**userData**

String.

Specifies the user data that is passed to the message exit.

**messageRetry**

A JSON object that contains the following attributes that define the channel message retry exit:

**name**

String.

Specifies the message retry exit name.

**userData**

String.

Specifies the user data that is passed to the message retry exit.

**receive**

An array of JSON objects that contain the following attributes that define the channel receive exits:

**name**

String.

Specifies the receive exit name.

**userData**

String.

Specifies the user data that is passed to the receive exit.

**security**

A JSON object that contains the following attributes that define the channel security exit:

**name**

String.

Specifies the security exit name.

**userData**

String.

Specifies the user data that is passed to the security exit.

**send**

An array of JSON objects that contain the following attributes that define the channel send exits:

**name**

String.

Specifies the send exit name.

**userData**

String.

Specifies the user data that is passed to the send exit.

**extended**

The extended object contains attributes that are related to extended channel properties, such as data conversion and sequence number settings:

**channelAgentType**

String.

Specifies the type of the message channel agent program.

The value is one of the following values:

**process****thread****messagePropertyControl**

String.

Specifies what happens to message properties when the message is about to be sent to a V6 or earlier queue manager, which does not understand the concept of a property descriptor.

The value is one of the following values:

**compatible**

If the message contains a property with a prefix of mcd., jms., usr. or mqext., all message properties are delivered to the application in an MQRFH2 header. Otherwise, all properties of the message, except those properties that are contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

**none**

All properties of the message, except those properties in the message descriptor (or extension), are removed from the message before the message is sent to the remote queue manager.



**all**

All properties of the message are included with the message when it is sent to the remote queue manager. The properties, except those properties in the message descriptor (or extension), are placed in one or more MQRFH2 headers in the message data.

**senderDataConversion**

Boolean.

Specifies whether the sender must convert application data.

**sequenceNumberWrap**

Integer.

Specifies the maximum message sequence number.

When the maximum is reached, sequence numbers wrap to start again at 1.

**resetSequenceNumber**

Integer.

Specifies the pending reset sequence number.

A nonzero value indicates that a reset channel request is outstanding. The value is in the range 1 - 999999999.

 **z/OS securityPolicyProtection**

String

Specifies what happens to messages across the channel when AMS is active and an applicable policy exists.

This parameter is applicable to Sender, Server, Receiver and Requester channels.

The value is one of the following:

**passThrough**

Pass through, unchanged, any messages sent or received by the MCA for this channel.

This value is valid for channels with a channel type of sender, server, receiver, or requester, and is the default value.

**remove**

Remove any AMS protection from messages retrieved from the transmission queue by the MCA, and send the messages to the partner.

When the message channel agent gets a message from the transmission queue, if an AMS policy is defined for the transmission queue, it is applied to remove any AMS protection from the message prior to sending the message across the channel. If an AMS policy is not defined for the transmission queue, the message is sent as is.

This value is valid for channels with a channel type of sender or server only.

**asPolicy**

Based on the policy defined for the target queue, apply AMS protection to inbound messages prior to putting them on to the target queue.

When the message channel agent receives an inbound message, if an AMS policy is defined for the target queue, AMS protection is applied to the message prior to the message being put to the target queue. If an AMS policy is not defined for the target queue, the message is put to the target queue as is.

This value is valid for channels with a channel type of receiver or requester only.

**failedDelivery**

The `failedDelivery` object contains attributes that are related to channel behavior when delivery of a message fails:

**retry**

JSON object.

Specifies the maximum number of attempts and the interval of attempts that are made to establish a connection to the remote machine before the `longRetry.count` and `longRetry.interval` are used:

**count**

Integer.

Specifies the maximum number of attempts to redeliver the message.

**interval**

Integer.

Specifies the interval, in milliseconds, between attempts to redeliver the message.

This attribute is only returned for channels of type receiver, requester, and clusterReceiver.

**useDeadLetterQueue**

Boolean.

Specifies whether the dead-letter queue is used when messages cannot be delivered by channels:

**false**

Specifies that messages that cannot be delivered by a channel are treated as a failure. The channel either discards the message, or the channel ends, in accordance with the `nonPersistentMessageSpeedFast` setting.

**true**

Specifies that when the DEADQ attribute of a queue manager provides the name of a dead-letter queue, then the dead letter queue is used. Otherwise, the behavior is as for `false`.

**general**

The `general` object contains attributes that are related to more generic channel properties, such as description:

**description**

String.

Specifies the description of the channel.

**maximumMessageLength**

Integer.

Specifies the maximum message length that can be transmitted on the channel. This value is compared with the value for the remote channel, and the actual maximum is the lower of the two values.

**batch**

The `batch` object contains attributes that are related to batches of messages that are sent through the channel:

**preCommitHeartbeat**

Integer.

Specifies whether batch heartbeats are used.

The value is the length of the heartbeat in milliseconds.

**timeExtend**

Integer.

Specifies the approximate time, in milliseconds, that a channel keeps a batch open if fewer than `batch.messageLimit` messages have been transmitted in the current batch.

**dataLimit**

Integer.

Specifies the limit, in KB, of the amount of data that can be sent through a channel before a sync point is taken.

**messageLimit**

Integer.

Specifies the maximum number of messages that can be sent through a channel before a sync point is taken.

**nonPersistentMessageSpeedFast**

Boolean.

Specifies whether fast speed is used to send nonpersistent messages.


Fast speed means that nonpersistent messages on a channel need not wait for a syncpoint before the messages are made available for retrieval.

**queueSharingGroup**

The `queueSharingGroup` object contains attributes that are related to queue sharing groups on z/OS:

**disposition**

String.

 This attribute is only available on z/OS.

Specifies the disposition of the channel. That is, where it is defined and how it behaves.

This value is always returned if the queue manager is a member of the queue sharing group.

The value is one of the following values:

**qmgr**

Specifies that the channel definition exists on the page set of the queue manager that runs the command.

**group**

Specifies that the channel definition exists in the shared repository.


**copy**

Specifies that the channel definition exists on the page set of the queue manager that runs the command, copying its definition from the channel of the same name defined in the shared repository.

This attribute cannot be used to filter results.

**defaultChannelDisposition**

String.

 This attribute is only available on z/OS.

Specifies the intended disposition of a channel when it is activated or started.

The value is one of the following values:

**private**

Specifies that the intended use of the object is as a private channel.

**fixShared**

Specifies that the intended use of the object is as a fixshared channel.

**shared**

Specifies that the intended use of the object is as a shared channel.

**receiverSecurity**

The `receiverSecurity` object contains attributes that are related to security for receiving channels:

**channelAgentUserId**

String.

Specifies the user identifier that is to be used by the message channel agent for authorization to access IBM MQ resources, including authorization to put the message to the destination queue for receiver or requester channels.

If the value is blank, the message channel agent uses its default user identifier.

### **putAuthority**

String.

Specifies which user identifiers are used to establish authority to put messages to the destination queue.

The value is one of the following values:

#### **default**


Specifies that the default user identifier is used.

#### **context**

Specifies that the user ID from the `UserIdentifier` field of the message descriptor is used.

#### **alternateOrChannelAgent**

Specifies that the user ID from the `UserIdentifier` field of the message descriptor is used.

 This value is only supported on z/OS.

#### **onlyChannelAgent**

Specifies that the user ID derived from MCAUSER is used.

## **transmissionSecurity**

The `transmissionSecurity` object contains attributes that are related to security for message transmission:

### **certificateLabel**

String.

Specifies which personal certificate in the key repository is sent to the remote peer.

If this attribute is blank, the certificate is determined by the queue manager **CERTLABL** parameter.

### **cipherSpecification**

String.

Specifies the name of the cipher that the channel uses.

### **requirePartnerCertificate**

Boolean.

Specifies whether IBM MQ requires a certificate from the TLS client.

### **certificatePeerName**

String.

Specifies the filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel. A Distinguished Name is the identifier of the TLS certificate.

## **currentStatus**

The `currentStatus` object contains attributes that are related to current status information:

### **inDoubt**

Boolean.

Specifies whether the channel is in doubt.

A sending channel is in doubt only while the sending message channel agent is waiting for an acknowledgment that a batch of sent messages has been successfully received.

### **state**

String.

Specifies the current status of the channel.

The value is one of the following values:

**binding**

Specifies that the channel is negotiating with the partner.

**starting**

Specifies that the channel is waiting to become active.

**running**

Specifies that the channel is transferring or waiting for messages.

**paused**

Specifies that the channel is paused.

**stopping**

Specifies that the channel is in process of stopping.

**retrying**

Specifies that the channel is reattempting to establish connection.

**stopped**

Specifies that the channel is stopped.

**requesting**

Specifies that the requester channel is requesting connection.

**switching**

Specifies that the channel is switching transmission queues.

**initializing**

Specifies that the channel is initializing.

**agent**

A JSON object that contains attributes that are related to the message channel agent:

**jobName**

String.

Specifies the name of the MCA job.

**running**

Boolean.

Specifies whether the MCA is running or not.

**state**

String.

Specifies the current action being performed by the MCA.

The value is one of the following values:

**runningChannelAutoDefinitionExit**

Specifies that the MCA is running a channel auto-definition exit.

**compressingData**

Specifies that the MCA is compressing or decompressing data.

**processingEndOfBatch**

Specifies that the MCA is performing end of batch processing.

**performingSecurityHandshake**

Specifies that the MCA is performing TLS handshaking.

**heartbeating**

Specifies that the MCA is heartbeating with a partner.

**executingMQGET**

Specifies that the MCA is performing an MQGET.

**executingMQI**

Specifies that the MCA is executing an IBM MQ API call, other than an MQPUT or MQGET.

**executingMQPUT**

Specifies that the MCA is performing an MQPUT.

**runningRetryExit**

Specifies that the MCA is running a retry exit.

**runningMessageExit**

Specifies that the MCA is running a message exit.

**communicatingWithNameServer**

Specifies that the MCA is processing a name server request.

**connectingToNetwork**

Specifies that the MCA is connecting to the network.

**undefined**

Specifies that the MCA is in an undefined state.

**runningReceiveExit**

Specifies that the MCA is running a receive exit.

**receivingFromNetwork**

Specifies that the MCA is receiving from the network.

**resynchingWithPartner**

Specifies that the MCA is resynching with a partner.

**runningSecurityExit**

Specifies that the MCA is running a security exit.

**runningSendExit**

Specifies that the MCA is running a send exit.

**sendingToNetwork**

Specifies that the MCA is performing a network send.

**serializingAccessToQmgr**

Specifies that the MCA is serialized on queue manager access.

**userId**

Specifies the user ID that is in use by the MCA.

This attribute is only applicable to receiver, requester, and cluster receiver channels.

**batch**

JSON Object containing attributes that are related to batches of messages:

**count**

Integer.

Specifies the number of completed batches.

**currentMessages**

Integer.

Specifies the number of messages that are sent or received in the current batch.

When a sending channel becomes in-doubt, it specifies the number of the messages that are in-doubt.

The number is reset to 0 when the batch is committed.

**luwid**

JSON object that contains attributes that are related to logical units of work:

**current**

String.

This identifier is represented as 2 hexadecimal digits for each byte.

Specifies the logical unit of work identifier that is associated with the current batch.

For a sending channel, when the channel is in-doubt it is the LUWID of the in-doubt batch.

**last**

String.

This identifier is represented as 2 hexadecimal digits for each byte.

Specifies the logical unit of work identifier that is associated with the last committed batch.

**nonPersistentMessageSpeedFast**

Boolean.

Specifies whether non-persistent messages are to be sent at fast speed.

**sequenceNumber**

JSON object that contains attributes that are related to sequence numbers:

**current**

Integer.

Specifies the message sequence number of the last message sent or received.

When a sending channel becomes in-doubt, it is the message sequence number of the last message in the in-doubt batch.

**last**

Integer.

Specifies the sequence number of last message in last committed batch.

**size**

Integer.

Specifies the negotiated batch size.

**compression**

JSON Object that contains attributes that are related to data compression:

**header**

JSON object that contains attributes that are related to header data compression:

**default**

String.

Specifies the default header data compression value that is negotiated for this channel.

The value is one of the following values:

**none**

Specifies that no header data compression is performed.

**system**

Specifies that header data compression is performed.

**lastMessage**

String.

Specifies the header data compression value that was used for the last message sent.

The value is one of the following values:

**none**

Specifies that no header data compression was performed.

**system**

Specifies that header data compression was performed.

**unavailable**

Specifies that no message was sent.

**message**

JSON object that contains attributes that are related to message data compression:

**default**

String.

Specifies the default message data compression value that was negotiated for this channel.

The value is one of the following values:

**none**

Specifies that no message data compression is performed.

**runLengthEncoding**

Specifies that message data compression is performed by using run-length encoding.

**zlibFast**

Specifies that message data compression is performed by using ZLIB encoding with speed prioritized.

**zlibHigh**

Specifies that message data compression is performed by using ZLIB encoding with compression prioritized.

**lastMessage**

String.

Specifies the message data compression value that was used for the last message sent.

The value is one of the following values:

**none**

Specifies that no message data compression was performed.

**runLengthEncoding**

Specifies that message data compression was performed by using run-length encoding.

**zlibFast**

Specifies that message data compression was performed by using ZLIB encoding with speed prioritized.

**zlibHigh**

Specifies that message data compression was performed by using ZLIB encoding with compression prioritized.

**unavailable**

Specifies that no message was sent.

**connectionManagement**

JSON Object that contains attributes that are related to connection management:

**heartbeatInterval**


Integer.

Specifies the heartbeat interval in seconds.

**keepAliveInterval**

Integer.

Specifies the value that is passed to the communications stack for KeepAlive timing for the channel.

 This parameter is only available on the z/OS

**localAddress**

An array of JSON objects that can contain the following attributes that define the local communications address of the channel:

**host**

String.

Specifies the IP address or host name that is used for local communications.

**port**

Integer.

Specifies the port number that is used for local communications.

This attribute cannot be used to filter results.

If no local address information is available, an empty array is returned.

**remainingRetries**

JSON object that contains attributes that are related to connection retry attempts:



**long**

Integer.

Specifies the number of long retry attempts remaining.

**last**

Integer.

Specifies the number of short retry attempts remaining.

This object is applicable only to sender, server, and cluster-sender channels.

**extended**

JSON object that contains attributes that are related to extended channel status properties:

**buffers**

JSON object that contains the following attributes that are related to buffers:

**received**

Integer.

Specifies the number of buffers received.

**sent**

Integer.

Specifies the number of buffers sent.

**bytes**

JSON object that contains the following attributes that are related to data transmission:

**received**

Integer.

Specifies the number of bytes received.

**sent**

Integer.

Specifies the number of bytes sent.

**messageCount**

Integer.

Specifies the total number of messages that are sent or received, or the number of MQI calls handled.

**general**

JSON Object containing more generic attributes that are related to channels:

**heartbeatInterval**


Integer.

Specifies the heartbeat interval in seconds.

**keepAliveInterval**

Integer.

Specifies the value that is passed to the communications stack for KeepAlive timing for the channel.

 This parameter is only available on the z/OS

**connection**

An array of JSON objects that can contain the following attributes that define the remote communications address of the channel:

**host**

String.

Specifies the remote IP address or host name.

**port**

Integer.

Specifies the remote port number.

This attribute cannot be used to filter results.

If no connection information is available, an empty array is returned.

If the connection does not conform to the expected syntax, an array containing a single host attribute having the value of the entire connection is returned.

**maximumMessageLength**

Integer.

Specifies the maximum length of a message.

**statistics**

String.

Specifies the rate at which statistics data is collected for the channel.

The value is one of the following values:

**off**

Specifies that no data is collected.

**low**

Specifies a low rate of data collection.

**medium**

Specifies a medium rate of data collection.

**high**

Specifies a high rate of data collection.

**stopRequested**

Boolean.

Specifies whether a stop request from the user has been received.

**transmissionQueueName**

String.

Specifies the name of the transmission queue in use by the channel.

**monitoring**

JSON object that contains more generic attributes that are related to channel monitoring:

**messagesInBatch**

JSON object that contains information about the number of messages in a batch:

**shortSamplePeriod**

Specifies the number of messages in a batch, based on recent activity over a short period.

**longSamplePeriod**

Specifies the number of messages in a batch, based on activity over a long period.

**rate**

String.

Specifies the rate at which monitoring data is collected for the channel.

The value is one of the following values:

**off**

Specifies that no data is collected.

**low**

Specifies a low rate of data collection.

**medium**

Specifies a medium rate of data collection.

**high**

Specifies a high rate of data collection.

**compressionRate**

JSON object that contains information about data compression rates:

**shortSamplePeriod**

Specifies the compression rate as a percentage, based on recent activity over a short period.

If no measurement is available, a value of -1 is returned.

**longSamplePeriod**

Specifies the compression rate as a percentage, based on activity over a long period.

If no measurement is available, a value of -1 is returned.

**compressionTime**

JSON object that contains information about data compression rates:

**shortSamplePeriod**

Specifies the compression speed as the time in microseconds spent compressing or decompressing each message, based on recent activity over a short period.

If no measurement is available, a value of -1 is returned.

**longSamplePeriod**

Specifies the compression speed as the time in microseconds spent compressing or decompressing each message, based on activity over a long period.

If no measurement is available, a value of -1 is returned.

**exitTime**

JSON object that contains information about exit processing speed:

**shortSamplePeriod**

Specifies the exit processing speed as the time in microseconds spent processing user exits for each message, based on recent activity over a short period.

If no measurement is available, a value of -1 is returned.

**longSamplePeriod**

Specifies the exit processing speed as the time in microseconds spent processing user exits for each message, based on activity over a long period.

If no measurement is available, a value of -1 is returned.

**messagesAvailable**

Integer.

Specifies the number of messages currently queued on the transmission queue and available for MQGETs.

**networkTime**

JSON object that contains information about network performance:

**shortSamplePeriod**

Specifies the time, in microseconds, to send a request to the remote end of the channel and receive a response, based on recent activity over a short period.

If no measurement is available, a value of -1 is returned.

**longSamplePeriod**

Specifies the time, in microseconds, to send a request to the remote end of the channel and receive a response, based on activity over a long period.

If no measurement is available, a value of -1 is returned.

**transmissionQueueTime**

JSON object that contains information about transmission queue delay:

**shortSamplePeriod**

Specifies the time, in microseconds, that messages remain on the transmission queue before being retrieved, based on recent activity over a short period.

If no measurement is available, a value of -1 is returned.

**longSamplePeriod**

Specifies the time, in microseconds, that messages remain on the transmission queue before being retrieved, based on activity over a long period.

If no measurement is available, a value of -1 is returned.

This attribute is only applicable to sender, server, and cluster sender channels.

**partner**

JSON Object that contains attributes that are related to the remote end queue manager:

**productIdentifier**

String.

Specifies the product identifier for the IBM MQ version that is running at the remote end of the channel.

The value is one of the following values:

**MQMM**

Queue Manager (non z/OS Platform)

**MQMV**

Queue Manager on z/OS

**MQCC**

IBM MQ C client

**MQNM**

IBM MQ .NET fully managed client

**MQJB**

IBM MQ Classes for Java

**MQJM**

IBM MQ Classes for JMS (normal mode)

**MQJN**

IBM MQ Classes for JMS (migration mode)

**MQJU**

Common Java interface to the MQI

**MQXC**

XMS client C/C++ (normal mode)

**MQXD**

XMS client C/C++ (migration mode)

**MQXN**

XMS client .NET (normal mode)

**MQXM**

XMS client .NET (migration mode)

**MQXU**

IBM MQ .NET XMS client (unmanaged/XA)

**MQNU**

IBM MQ .NET unmanaged client

**qmgrName**

String.

Specifies the name of the remote queue manager or queue sharing group.

**version**

String.

Specifies the version of IBM MQ running at the remote end of the channel, in the form V.R.M.F.

**maximumMessageLength**

Integer.


Specifies the maximum length of a message.

### **queueSharingGroup**

JSON Object that contains attributes that are related to the queue sharing group this channel belongs to:

#### **channelDisposition**

String.

 This attribute is only available on z/OS.

Specifies the disposition of the channel. That is, where it is defined and how it behaves.

The value is one of the following values:

#### **qmgr**

Specifies that the channel definition exists on the page set of the queue manager that runs the command.

#### **group**

Specifies that the channel definition exists in the shared repository.

#### **copy**

Specifies that the channel definition exists on the page set of the queue manager that runs the command, copying its definition from the channel of the same name defined in the shared repository.

### **timestamps**

JSON object that contains attributes that are related to date and time information:

#### **started**

String.

Specifies the date and time at which the channel was started.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

#### **lastMessage**

String.

Specifies the date and time at which the last message was sent over the channel.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

### **transmissionSecurity**

JSON object that contains attributes that are related to transmission security:

#### **certificateIssuerName**

String.

Specifies the full Distinguished Name of the issuer of the remote certificate.

#### **certificateUserId**

String.

Specifies the local user ID that is associated with the remote certificate.

#### **cipherSpecification**

String.

Specifies the name of the cipher that the channel uses.

#### **keyLastReset**

String.

Specifies the date and time of the last successful TLS secret key reset.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).



#### **keyResetCount**


String.

Specifies the number of successful TLS secret key resets since the channel started.

**protocol**

String.

  This parameter is available on AIX, Linux, and Windows platforms and on the IBM MQ Appliance.

 From IBM MQ 9.1.1, this parameter is also available on z/OS.

Specifies the security protocol currently in use.

The value is one of the following values:

**none**

Specifies that no security protocol is in use.

**sslV30**

Specifies that SSL 3.0 is in use.

**tlsV10**

Specifies that TLS 1.0 is in use.

**tlsV12**

Specifies that TLS 1.2 is in use.

**shortPeerName**

String.

Specifies the Distinguished Name of the peer queue manager or client at the other end of the channel.

**savedStatus**

The savedStatus object contains attributes that are related to saved status information:

**inDoubt**

Boolean.

Specifies whether the channel was in doubt.

A sending channel is only in doubt while the sending message channel agent is waiting for an acknowledgment that a batch of messages, which it has sent, has been successfully received.

**batch**

JSON Object that contains attributes that are related to batches of messages:

**currentMessages**

Integer.

Specifies the number of messages that are sent or received in the current batch or, if the channel was in-doubt, the number of messages that were in-doubt.

In the context of saved status, this number is only meaningful if the channel was in-doubt, but this value is returned regardless.

**luwid**

JSON object that contains attributes that are related to logical units of work:

**current**

String. This identifier is represented as 2 hexadecimal digits for each byte.

Specifies the logical unit of work identifier that is associated with the current batch.

For a sending channel, if the channel was in-doubt, it specifies the LUWID of the in-doubt batch.

In the context of saved status, this number is only meaningful if the channel was in-doubt, but this value is returned regardless.

**last**

Hex string.

Specifies the logical unit of work identifier that is associated with the last committed batch.

### **sequenceNumber**

JSON object that contains attributes that are related to sequence numbers:

#### **current**

Integer.

Specifies the message sequence number of the last message that is sent or received.

When a sending channel is in-doubt, it specifies the sequence number of the last message in the in-doubt batch.

#### **last**

Integer.

Specifies the sequence number of the last message in the last committed batch.

### **general**

JSON Object that contains more generic attributes that are related to channels:

#### **connection**

An array of JSON objects that can contain the following attributes that define the remote communications address of the channel:

##### **host**

String.

Specifies the remote IP address or host name.

##### **port**

Integer.

Specifies the remote port number.

This attribute cannot be used to filter results.

If no connection information is available, an empty array is returned.

If the connection does not conform to the expected syntax, an array containing a single host attribute having the value of the entire connection is returned.

#### **transmissionQueueName**

String.

Specifies the name of the transmission queue in use by the channel.

### **queueSharingGroup**

JSON Object that contains attributes that are related to the queue sharing group this channel belonged to:

#### **channelDisposition**

String.

 This attribute is only available on z/OS.

Specifies the disposition of the channel. That is, where it was defined and how it behaved.

The value is one of the following values:

##### **qmgr**

Specifies that the channel definition existed on the page set of the queue manager that runs the command.

##### **group**

Specifies that the channel definition existed in the shared repository.

##### **copy**

Specifies that the channel definition existed on the page set of the queue manager that runs the command, copying its definition from the channel of the same name defined in the shared repository.

## **/admin/qmgr/{qmgrName}/queue**

You can use the HTTP GET method with the queue resource to request information about queues. You can use the HTTP POST method to create queues, the PATCH method to modify queues, and the DELETE method to delete queues.

### **Note:**

- This resource URL is available only in version 1 of the REST API. To create, update, delete, or display queues using version 2 of the REST API, use the [“/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 resource](#).
- **V 9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

You can use the administrative REST API gateway with this resource URL.

For more information about the PCF equivalents to the queue REST API parameters and attributes, see [REST API and PCF equivalents for queues](#).

### **POST**

Use the HTTP POST method with the queue resource to create a queue on a specified queue manager.

### **Note:**

- This resource URL is available only in version 1 of the REST API. To create queues using version 2 of the REST API, use the [“/admin/action/qmgr/{qmgrName}/mqsc” on page 2159 resource](#).
- **V 9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

This REST API command is similar to the [“Change, Copy, and Create Queue” on page 1064 PCF command](#), and the [“DEFINE queues” on page 547 MQSC commands](#).

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers” on page 2314](#)
- [Request body format](#)
- [“Security requirements” on page 2315](#)
- [Response status codes](#)
- [“Response headers” on page 2316](#)
- [Response body format](#)
- [Examples](#)

### **Resource URL**

`https://host:port/ibmmq/rest/v1/admin/qmgr/{qmgrName}/queue`

### **qmgrName**

Specifies the name of the queue manager on which to create the queue.

You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.




- A period (.) must be encoded as %2E.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Optional query parameters

### **commandScope=scope**

 This parameter is only available on z/OS.

Specifies how the command is run when the queue manager is a member of a queue sharing group. You cannot specify this parameter if the queue manager is not a member of a queue sharing group. *scope* can be one of the following values:

#### **The name of a queue manager**

Specifies that the command is run on the queue manager that is named. The queue manager must be active within the same queue sharing group as the queue manager that is specified in the resource URL.

You cannot specify the queue manager name that is the queue manager that is specified in the resource URL.

If the queue manager name includes a percent sign, %, this character must be URL encoded as %25.

★


Specifies that the command is run on the local queue manager and also passed to every active queue manager in the queue sharing group.

If this option is used, an `ibm-mq-qmgrs` response header is returned with a comma-separated list of the queue managers that generated a response. For example, the header might look like the following header:

```
ibm-mq-qmgrs: MQ21, MQ22
```

### **like=qName**

Specifies an existing queue definition to copy.

 On z/OS, the way that a queue is copied depends on the value that is specified for the **disposition** parameter in the request body:

- If copy is specified, the **like** parameter is ignored. The queue to copy is a queue with the name that is specified by the **name** parameter in the request body and with a disposition of `group`.
- If copy is not specified, the queue to copy is a queue with the name that is specified by the **like** parameter and a disposition of `qmgr`, `copy`, or `shared`.

### **noReplace**

Specifies that the queue is not replaced if it exists. If this flag is not specified, the queue is replaced.

If a queue is replaced, any messages that are on the existing queue are retained.

The queue is not replaced in the following scenarios:

- The queue is a local queue. **allowedSharedInput** is changed to `false`, and more than one application has the local queue open for input.
- The queue is a local queue. The value of **isTransmissionQueue** is changed, and one or more applications has the local queue open, or if one or more messages are on the queue.
- The queue is a remote queue. The value of **transmissionQueueName** is changed, and an application has a remote queue open that would be affected by this change.
- The queue is a remote queue. The value of **queueName**, **qmgrName**, or **transmissionQueueName** is changed, and one or more applications has a queue open that resolved through this definition as a queue manager alias.

## Request headers

The following headers must be sent with the request:

### Content-Type

This header must be sent with a value of `application/json` optionally followed by `; charset=UTF-8`.

### ibm-mq-rest-csrf-token

This header must be set, but the value can be anything, including being blank.

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### ibm-mq-rest-gateway-qmgr

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## Request body format

The request body must be in JSON format in UTF-8 encoding. Within the request body attributes are defined, and named JSON objects are created to specify extra attributes. Any attributes that are not specified use the default value. These default values are as specified for the `SYSTEM.DEFAULT` queues on the queue manager. For example, a local queue inherits the values that are defined in `SYSTEM.DEFAULT.LOCAL.QUEUE`.

For example, the following JSON contains some attributes, and then the named JSON objects, `events` and `storage`. These named JSON objects define the extra attributes to create a local queue with queue depth high events enabled, and a maximum queue depth of 1000:

```
{
  "name": "queue1",
  "type": "local",
  "events": {
    "depth": {
      "highEnabled": true,
      "highPercentage": 75
    }
  },
  "storage": {
    "maximumDepth": 1000
  }
}
```

For more examples, see [examples](#).

The following attributes can be included in the request body:

### name

Required.

String.

Specifies the name of the queue to create.

### type

String.

Specifies the type of queue.

The value can be one of the following values:

- local
- alias
- model

- remote

The default value is local.

The following objects can be included in the request body to specify extra attributes:

**remote**

Contains attributes that are related to remote queues. The attributes in this object are supported only for remote queues.

**alias**

Contains attributes that are related to alias queues. The attributes in this object are supported only for alias queues.

**model**

Contains attributes that are related to model queues. The attributes in this object are supported only for model queues.

**cluster**

Contains attributes that are related to clusters.

**trigger**

Contains attributes that are related to triggering.

**events**

Contains two objects, one for queue depth and one for queue service interval events. Each object contains attributes that are related to the event type.

**applicationDefaults**

Contains attributes that are related to default behavior such as message persistence, message priority, shared input settings, and read ahead settings.

**queueSharingGroup**

Contains attributes that are related to queue sharing groups on z/OS.

**dataCollection**

Contains attributes that are related to data collection, monitoring, and statistics.

**storage**

Contains attributes that are related to message storage, such as the maximum depth of the queue, and the maximum length of messages that are allowed on the queue.

**general**

Contains attributes that are related to general queue properties, such as whether get or put operations are inhibited, the description of the queue, and transmission queue settings.

**extended**

Contains attributes that are related to extended queue properties, such as backout queue settings, and shared input settings.

For more information, see [“Request body attributes for queues” on page 2318](#).

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

The security principal of the caller must be granted the ability to issue the following PCF commands for the specified queue manager:

- If the **like** optional query parameter is not specified:
  - For the queue that is specified by the **name** attribute in the request body, authority to issue the **MQCMD\_CREATE\_Q** PCF command must be granted.

- For the relevant `SYSTEM.DEFAULT.*.QUEUE`, authority to issue the `MQCMD_INQUIRE_Q` PCF command must be granted.
- If the **like** optional query parameter is specified:
  - For the queue that is specified by the **name** attribute in the request body, authority to issue the `MQCMD_COPY_Q` PCF command must be granted.
  - For the queue that is specified by the **like** optional query parameter, authority to issue the `MQCMD_INQUIRE_Q` PCF command must be granted.

**ALW** On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the `setmqaut` command. For more information, see [setmqaut \(grant or revoke authority\)](#).

**z/OS** On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 201

Queue created successfully.

### 400

Invalid data provided.

For example, invalid queue data is specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the `MQWebAdmin`, `MQWebAdminRO`, or `MQWebUser` roles. The `ibm-mq-rest-csrf-token` header must also be specified. For more information, see [“Security requirements” on page 2315](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources. For more information about the access that is required, see [“Security requirements” on page 2315](#).
- **V 9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

The following headers are returned with the response:

### location

If the request was successful, this header specifies the URL for the new queue.

If the optional query parameter `commandScope=*` is used, the URL that is returned is the URL for the local copy of the queue. If the optional query parameter `commandScope=qmgrName` is used, the URL that is returned is a partial URL that does not include information about the host and port.

On z/OS, if the optional query parameter `commandScope=*` is used, this header is returned with a comma-separated list of the queue managers that generated a response. For example, the header might look like the following header:

```
ibm-mq-qmgrs: MQ21, MQ22
```

If an error occurs before the command is issued to the queue managers, the response header does not contain the list of queue managers. For example, a request that generates a 200 or 201 status code has the header because the command was successful. A request that generates a 401 (not authenticated) status code does not have the header because the request was rejected. A request that generates a 403 (not authorized) status code has the header because individual queue managers decide whether the command is authorized.

### **ibm-mq-rest-gateway-qmgr**

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

## **Response body format**

The response body is empty if the queue is created successfully. If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

## **Examples**

- The following example creates a local queue called `localQueue`. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/
```

The following JSON payload is sent:

```
{
  "name": "localQueue"
}
```

- The following example creates a remote queue called `remoteQueue`. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/
```

The following JSON payload is sent:

```
{
  "name": "remoteQueue",
  "type": "remote",
  "remote": {
    "queueName": "localQueue",
    "qmgrName": "QM2"
  }
}
```

- The following example creates an alias queue called `aliasQueue`. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/
```

The following JSON payload is sent:

```
{
  "name": "aliasQueue",
  "type": "alias",
  "alias": {
    "targetName": "localQueue"
  }
}
```

```
}  
}
```

- The following example creates a model queue called `modelQueue`. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/
```

The following JSON payload is sent:

```
{  
  "name": "modelQueue",  
  "type": "model",  
  "model": {  
    "type": "permanentDynamic"  
  }  
}
```

- The following example creates a clustered remote queue that is called `remoteQueue1`. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/
```

The following JSON payload is sent:

```
{  
  "name": "remoteQueue1",  
  "type": "remote",  
  "remote": {  
    "queueName": "aLocalQueue1",  
    "qmgrName": "QM2",  
    "transmissionQueueName": "MY.XMITQ"  
  },  
  "general": {  
    "description": "My clustered remote queue"  
  },  
  "cluster": {  
    "name": "Cluster1",  
    "workloadPriority": 9  
  }  
}
```

- The following example creates a clustered remote queue, `remoteQueue2`, based on another queue, `remoteQueue1`. All the attributes from `remoteQueue1` are used, except for the queue name and the remote queue name. The following URL is used with the HTTP POST method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/?like=remoteQueue1
```

The following JSON payload is sent:

```
{  
  "name": "remoteQueue2",  
  "type": "remote",  
  "remote": {  
    "queueName": "aLocalQueue2"  
  }  
}
```

#### *Request body attributes for queues*

When you create the request body for creating or modifying a queue with the administrative REST API, you can specify attributes for the queue within named JSON objects. A number of objects and attributes are available.

The following objects are available:

- [“remote” on page 2319](#)
- [“alias” on page 2319](#)
- [“model” on page 2320](#)

- [“cluster” on page 2320](#)
- [“trigger” on page 2321](#)
- [“events” on page 2322](#)
- [“applicationDefaults” on page 2324](#)
- [“queueSharingGroup” on page 2325](#)
- [“dataCollection” on page 2327](#)
- [“storage” on page 2328](#)
- [“general” on page 2329](#)
- [“extended” on page 2330](#)

For more information about the PCF equivalents to the queue REST API parameters and attributes, see [“REST API and PCF equivalents for queues” on page 2380](#).

## remote

**Note:** The `remote` object and the `qmgrName` attribute are required when you create a remote queue by using the HTTP POST method. You cannot use the `remote` object unless you are creating a remote queue, or updating a remote queue.

The `remote` object can contain the following attributes that relate to remote queues:

### queueName

String.

Specifies the name of the queue as it is known on the remote queue manager.

If this attribute is omitted, a queue manager alias or reply-to queue alias is created.

### qmgrName

String.

Specifies the name of the remote queue manager.

Required when you create a queue by using the HTTP POST method, unless you use the **like** optional query parameter.

If this remote queue is used as a queue manager alias, this attribute is the name of the queue manager. The value can be the name of the queue manager in the resource URL.

If this remote queue is used as a reply-to queue alias, this attribute is the name of the queue manager that is to be the reply-to queue manager.

### transmissionQueueName

String.

Specifies the name of the transmission queue that is to be used for messages that are destined for either a remote queue or for a queue manager alias definition.

This attribute is ignored in the following cases:

- The remote queue is used as a queue manager alias and **qmgrName** attribute is the name of the queue manager in the resource URL.
- The remote queue is used as a reply-to queue alias.

If this attribute is omitted, a local queue with the name that is specified by the **qmgrName** attribute must exist. This queue is used as the transmission queue.

## alias

**Note:** The `alias` object and the `targetName` attribute are required when you create an alias queue by using the HTTP POST method. You cannot use the `alias` object unless you are creating an alias queue, or updating an alias queue.

The `alias` object can contain the following attributes that relate to alias queues:

**targetName**

String.

Specifies the name of the queue or topic that the alias resolves to.

Required when you create a queue by using the HTTP POST method, unless you use the **like** optional query parameter.

**targetType**

String.

Specifies the type of object that the alias resolves to.

The value must be one of the following values:

**queue**

Specifies that the object is a queue.

**topic**

Specifies that the object is a topic.

The default value is queue.

**model**

**Note:** The `model` object and the `type` attribute are required when you create a model queue by using the HTTP POST method. You cannot use the `model` object unless you are creating a model queue, or updating a model queue.

The `model` object can contain the following attributes that relate to model queues:

**type**

String.


Specifies the model queue definition type.

The value must be one of the following values:

**permanentDynamic**

Specifies that the queue is a dynamically defined permanent queue.

**sharedDynamic**

 This attribute is only available on z/OS.

Specifies that the queue is a dynamically defined shared queue.

**temporaryDynamic**

Specifies that the queue is a dynamically defined temporary queue.

The default value is `temporaryDynamic`.

**cluster**

The `cluster` object can contain the following attributes that relate to clusters:

**name**

String.

Specifies the name of the cluster that the queue belongs to.

Specify either the **name** or **namelist** cluster attributes. You cannot specify both attributes.

**namelist**

String.

Specifies the namelist that lists the clusters that the queue belongs to.

Specify either the **name** or **namelist** cluster attributes. You cannot specify both attributes.

**transmissionQueueForChannelName**

String.



Specifies the generic name of the cluster-sender channels that use the queue as a transmission queue. The attribute specifies which cluster-sender channels send messages to a cluster-receiver channel from the cluster transmission queue.

You can also set this attribute to a cluster-sender channel manually. Messages that are destined for the queue manager that is connected by the cluster-sender channel are stored in the transmission queue that identifies the cluster-sender channel. The messages are not stored in the default cluster transmission queue.

If you set the **transmissionQueueForChannelName** attribute to blanks, the channel switches to the default cluster transmission queue when the channel restarts. The default cluster transmission queue is `SYSTEM.CLUSTER.TRANSMIT.QUEUE` if the queue manager **DefClusterXmitQueueType** attribute is set to `SCTQ`. A specific cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.ChannelName`, is used for each cluster-sender channel if the queue manager **DefClusterXmitQueueType** attribute is set to `CHANNEL`.

By specifying asterisks, `*`, in **transmissionQueueForChannelName**, you can associate a transmission queue with a set of cluster-send channels. The asterisks can be at the beginning, end, or any number of places in the middle of the channel name string.

### **workloadPriority**

Integer.

Specifies the priority of the queue in cluster workload management.

The value must be in the range 0 - 9, where 0 is the lowest priority and 9 is the highest.

### **workloadRank**

Integer.

Specifies the rank of the queue in cluster workload management.

The value must be in the range 0 - 9, where 0 is the lowest priority and 9 is the highest.

### **workloadQueueUse**

String.

Specifies whether remote and local instances of the clustered queues are to be used in cluster workload distribution.

The value must be one of the following values:

#### **asQmgr**

Use the value that is defined on the queue manager.

#### **any**

Use remote and local instances of the queues.

#### **local**

Use only local instances of the queues.

## **trigger**

The **trigger** object can contain the following attributes that relate to triggering:

### **data**

String.

Specifies the user data that is included in the trigger message. This data is made available to the monitoring application that processes the initiation queue and to the application that is started by the monitor.

### **depth**

Integer.

Specifies the number of messages that initiates a trigger message to the initiation queue.

The value must be in the range 1 - 999,999,999.

This attribute is required when **type** is set to `depth`.

**enabled**

Boolean.

Specifies whether trigger messages are written to the initiation queue.

If the value is set to `true`, trigger messages are written to the initiation queue.

**initiationQueueName**

String.

Specifies the local queue for trigger messages that relate to the queue. The queues must be on the same queue manager.

**messagePriority**

Integer.

Specifies the minimum priority that a message must have before it can cause, or be counted for, a trigger event.

The value must be in the range 0 - 9.

**processName**

String.

Specifies the local name of the IBM MQ process that identifies the application to be started when a trigger event occurs.

If the queue is a transmission queue, the process definition contains the name of the channel to be started.

**type**

String.

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

The value must be one of the following values:

**none**

Send no trigger messages.

**every**

Send a trigger message for every message that arrives on the queue.

**first**

Send a trigger message when the queue depth goes from 0 to 1.

**depth**

Send a trigger message when the queue depth exceeds the value of the **depth** attribute.

**events**

The `events` object can contain the following objects and attributes that relate to queue depth and queue service interval events:

**depth**

JSON object.

A JSON object that can contain the following attributes that related to queue depth events:

**fullEnabled**

Boolean.

Specifies whether queue full events are generated.

A queue full event indicates that no more messages can be put on a queue because the queue is full. That is, the queue depth reached the maximum queue depth, as specified by the **maximumDepth** attribute in the `storage` object.

If the value is set to `true`, queue full events are enabled.

**highEnabled**

Boolean.

Specifies whether queue depth high events are generated.

A queue depth high event indicates that the number of messages on the queue is greater than or equal to the queue depth high limit, **highPercentage**.

If the value is set to `true`, queue depth high events are enabled.

#### **highPercentage**

Integer.

Specifies the threshold against which the queue depth is compared to generate a queue depth high event.

This value is expressed as a percentage of the maximum queue depth, as specified by the **maximumDepth** attribute in the `storage` object. The value must be a value in the range 0 - 100.

#### **lowEnabled**

Boolean.

Specifies whether queue depth low events are generated.

A queue depth low event indicates that the number of messages on the queue is less than or equal to the queue depth low limit, **lowPercentage**.

If the value is set to `true`, queue depth low events are enabled.

#### **lowPercentage**

Integer.

Specifies the threshold against which the queue depth is compared to generate a queue depth low event.

This value is expressed as a percentage of the maximum queue depth, as specified by the **maximumDepth** attribute in the `storage` object. The value must be a value in the range 0 - 100.

#### **serviceInterval**

JSON object.

A JSON object that can contain the following attributes that are related to queue service interval events:

##### **duration**

Integer.

Specifies the service interval duration that is used for comparison to generate queue service interval high and queue service interval OK events.

The value must be a value in the range 0 - 999,999,999 milliseconds.

##### **highEnabled**

Boolean.

Specifies whether queue service interval high events are generated.

A queue service interval high event is generated when a check indicates that no messages were put to, or retrieved from, the queue for at least the amount of time specified by the **duration** attribute.

If the value is set to `true`, queue service interval high events are enabled.

If you set the **highEnabled** attribute to `false`, you must also specify a value for the **okEnabled** attribute. You cannot set both the **highEnabled** attribute and the **okEnabled** attribute to `true` at the same time.

##### **okEnabled**

Boolean.

Specifies whether queue service interval OK events are generated.

A queue service interval OK event is generated when a check indicates that a message was retrieved from the queue within the amount of time that is specified by the **duration** attribute.

If the value is set to `true`, queue service interval OK events are enabled.

If you set the **okEnabled** attribute to `false`, you must also specify a value for **highEnabled**. You cannot set both the **highEnabled** attribute and the **okEnabled** attribute to `true` at the same time.

## applicationDefaults

The `applicationDefaults` object can contain the following attributes that relate to default behavior such as message persistence:

### clusterBind

String.

Specifies the binding to be used when `MQOO_BIND_AS_Q_DEF` is specified on the `MQOPEN` call.

The value must be one of the following values:

#### onOpen

Specifies that the binding is fixed by the `MQOPEN` call.

#### notFixed

Specifies that the binding is not fixed.

#### onGroup

Specifies that the application can request that a group of messages is allocated to the same destination instance.

### messagePersistence

String.

Specifies the default for message persistence on the queue. Message persistence determines whether messages are preserved across restarts of the queue manager.

The value must be one of the following values:

#### persistent

Specifies that the messages on the queue are persistent, and are preserved when the queue manager restarts.

#### nonPersistent

Specifies that the messages on the queue are not persistent, and are lost when the queue manager restarts.

### messagePriority

Integer.

Specifies the default priority of messages that are put on the queue.

The value must be in the range 0 - 9, where 0 represents the lowest priority, and 9 represents the highest priority.

### messagePropertyControl

String.

Specifies how message properties are handled when messages are retrieved from queues when `MQGMO_PROPERTIES_AS_Q_DEF` is specified on the `MQGET` call.

This attribute is applicable to local, alias, and model queues.

The value must be one of the following values:

#### all

Specifies that all properties of the message are included when the message is sent to the remote queue manager. The properties, except those properties in the message descriptor or extension, are placed in one or more `MQRFH2` headers in the message data.

#### compatible

Specifies that if the message contains a property with the prefix `mcd.`, `jms.`, `usr.`, or `mqext.`, all message properties are delivered to the application in an `MQRFH2` header. Otherwise, all properties, except those properties in the message descriptor or extension, are discarded and are no longer accessible.

**force**

Specifies that properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle. A valid message handle that is included in the MsgHandle field of the MQGMO structure on the MQGET call is ignored. Properties of the message are not accessible by using the message handle.

**none**

Specifies that all properties of the message are removed from the message before the message is sent to the remote queue manager. Properties in the message descriptor, or extension, are not removed.

**version6Compatible**

Any application MQRFH2 header is received as it was sent. Any properties set by using MQSETMP must be retrieved by using MQINQMP. They are not added to the MQRFH2 created by the application. Properties that were set in the MQRFH2 header by the sending application cannot be retrieved by using MQINQMP.

**putResponse**

String.

Specifies the type of response that is to be used for put operations to the queue when an application specifies MQPMO\_RESPONSE\_AS\_Q\_DEF.

The value must be one of the following values:

**synchronous**

The put operation is run synchronously, returning a response.

**asynchronous**

The put operation is run asynchronously, returning a subset of MQMD fields.

**readAhead**

String.

Specifies the default read-ahead behavior for non-persistent messages that are delivered to the client.

The value must be one of the following values:

**no**

Specifies that non-persistent messages are not read ahead unless the client application is configured to request read ahead.

**yes**

Specifies that non-persistent messages are sent ahead to the client before an application requests them. Non-persistent messages can be lost if the client ends abnormally or if the client does not consume all the messages that it is sent.

**disabled**

Specifies that non-persistent messages are not read ahead, regardless of whether read ahead is requested by the client application.


**sharedInput**

Boolean.

Specifies the default share option for applications that open this queue for input.

If the value is set to true, queues are enabled to get messages with shared access.


**queueSharingGroup**

 The queueSharingGroup object can contain the following attributes that relate to queue sharing groups:



**disposition**

String.

 This attribute is only available on z/OS.

Specifies where the queue is defined and how it behaves. That is, it specifies the disposition of the queue.

The value must be one of the following values:

#### **copy**

Specifies that the queue definition exists on the page set of the queue manager that runs the command. The group object of the same name as the **name** attribute is used to create the queue.

For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.

#### **group**

Specifies that the queue definition exists in the shared repository.

This value is allowed only in a shared queue manager environment.

If the creation is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group. The command attempts to make or refresh local copies on page set zero:

```
DEFINE queue(q-name) REPLACE QSGDISP(COPY)
```

The creation of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

#### **qmgr**

Specifies that the queue definition exists on the page set of the queue manager that runs the command.

For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.

#### **shared**

This value is only valid for local queues.

Specifies that the queue exists in the shared repository.


Messages are stored in the coupling facility and are available to any queue manager in the queue sharing group. You can specify shared only if the following things are true:

- The value of **structureName** is not blank.
- The value of **indexType** is not messageToken.
- The queue is not SYSTEM.CHANNEL.INITQ or SYSTEM.COMMAND.INPUT.

The default value is qmgr.

#### **structureName**

String.

 This attribute is only available on z/OS.

Specifies the name of the coupling facility structure where you want to store messages when you used shared queues.

The value cannot have more than 12 characters, it must start with an uppercase letter (A - Z), and can include only the characters A - Z and 0 - 9.

The name of the queue sharing group to which the queue manager is connected is prefixed to the name you supply. The name of the queue sharing group is always 4 characters, padded with the at sign, @, if necessary. For example, if you use a queue sharing group that is named NY03 and you supply the name PRODUCT7, the resultant coupling facility structure name is NY03@PRODUCT7. Note the administrative structure for the queue sharing group (in this case NY03CSQ\_ADMIN) cannot be used for storing messages.

For local and model queues, the following rules apply. The rules apply if you create a queue without specifying the **noReplace** optional query parameter, or if you change the queue:

- On a local queue with a **disposition** value of shared, **structureName** cannot change. If you need to change the **structureName** or the **disposition**, you must delete and redefine the queue. To preserve any of the messages on the queue, you must offload the messages before you delete the queue. Reload the messages after you redefine the queue, or move the messages to another queue.
- On a model queue with a **definitionType** value of sharedDynamic, the **structureName** cannot be blank.

For local and model queues, when you create a queue with the **noReplace** optional query parameter, the following rules apply:

- On a local queue with a **disposition** value of shared, or a model queue with a **definitionType** value of sharedDynamic, the **structureName** cannot be blank.

## dataCollection

The dataCollection object can contain the following attributes that relate to the collection of data, monitoring, and statistics:

### accounting

String.

Specifies whether accounting data is collected for the queue.

The value must be one of the following values:

#### asQmgr

Specifies that the queue inherits the value from the queue manager MQSC parameter ACCTQ.

#### off

Specifies that accounting data is not collected for the queue.

#### on

Specifies that accounting data is collected for the queue if the ACCTQ MQSC parameter on the queue manager is not set to none.

### monitoring

String.

Specifies whether online monitoring data is to be collected, and if so, the rate at which the data is collected.

The value must be one of the following values:

#### off

Specifies that online monitoring data is not collected for the queue.

#### asQmgr

Specifies that the queue inherits the value from the queue manager MQSC parameter MONQ.

#### low

Specifies that online monitoring data is collected for the queue if the MONQ MQSC parameter on the queue manager is not set to none. The rate of data collection is low.

#### medium

Specifies that online monitoring data is collected for the queue if the MONQ MQSC parameter on the queue manager is not set to none. The rate of data collection is moderate.

#### high

Specifies that online monitoring data is collected for the queue if the MONQ MQSC parameter on the queue manager is not set to none. The rate of data collection is high.

### statistics

This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

String.

Specifies whether statistics data is to be collected for the queue.

The value must be one of the following values:

**asQmgr**

Specifies that the queue inherits the value from the queue manager STATQ MQSC parameter.

**off**

Specifies that statistics data is not collected for the queue.


**on**

Specifies that statistics data is collected for the queue if the STATQ MQSC parameter on the queue manager is not set to none.

## storage

The `storage` object can contain the following attributes that relate to message storage:

### indexType

 This attribute is only available on z/OS.

String.

Specifies the type of index that is maintained by the queue manager to expedite MQGET operations on the queue. For shared queues, the type of index determines what type of MQGET calls can be used.

The value must be one of the following values:

**none**

Specifies that there is no index. Messages are retrieved sequentially.

**correlationId**

Specifies that the queue is indexed by using correlation identifiers.

**groupId**

Specifies that the queue is indexed by using group identifiers.

**messageId**

Specifies that the queue is indexed by using message identifiers.

**messageToken**

Specifies that the queue is indexed by using message tokens.

The default value is none.

### maximumDepth

Integer.

Specifies the maximum number of messages that are allowed on the queue.

The value must be in the range 0 - 999,999,999.

### maximumMessageLength

Integer.

Specifies the maximum message length that is allowed for messages on the queue.

Do not set a value that is greater than the **maximumMessageLength** attribute for the queue manager.

The value must be in the range 0 - 104,857,600 bytes.

### messageDeliverySequence

String.

Specifies whether messages are delivered in priority order or by sequence.

The value must be one of the following values:

**priority**

Specifies that messages are returned in priority order.

**fifo**

Specifies that messages are returned in first in, first out order.



## **nonPersistentMessageClass**

 MQ Appliance

 ALW

This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

String.

This attribute is valid only on local and model queues.

Specifies the level of reliability to be assigned to non-persistent messages that are put to the queue.

The value must be one of the following values:

### **normal**

Specifies that non-persistent messages persist for the lifetime of the queue manager session. They are discarded if the queue manager restarts.

### **high**

Specifies that the queue manager attempts to retain non-persistent messages for the lifetime of the queue. Non-persistent messages might still be lost if a failure occurs.

## **storageClass**

 z/OS

This attribute is only available on z/OS.

String.

Specifies the name of the storage class.

## **general**

The `general` object can contain the following attributes that relate to general queue properties:

### **description**

String.

Specifies a description for the queue.

The characters in the description field are converted from UTF-8 into the CCSID of the queue manager. Ensure that you use only the characters that can be converted. Certain characters must be escaped:

- Double quotation marks, `"`, must be escaped as `\ "`
- A backslash, `\`, must be escaped as `\\`
- A forward slash, `/`, must be escaped as `\/`

### **inhibitGet**

Boolean.

Specifies whether `get` operations are allowed on the queue.

If the value is set to `true`, `get` operations are not allowed on the queue.

### **inhibitPut**

Boolean.

Specifies whether `put` operations are allowed on the queue.

If the value is set to `true`, `put` operations are not allowed on the queue.

### **isTransmissionQueue**

String.

Specifies whether the queue is for normal usage or for transmitting messages to a remote queue manager.

If the value is set to `true`, the queue is a transmission queue for transmitting messages to a remote queue manager.

The `isTransmissionQueue` attribute must not normally be changed while messages are on the queue. The format of messages changes when they are put on a transmission queue.

## extended

The extended object can contain the following attributes that relate to extended queue properties:

### allowSharedInput

Boolean.

Specifies whether multiple instances of applications can open the queue for input.

If the value is set to `true`, multiple instances of applications can open the queue for input.

### backoutRequeueQueueName

String.

Specifies the name of the queue to which a message is transferred if it is backed out more times than the value of **backoutThreshold**.

The backout queue does not need to exist when the queue is created, but it must exist when the **backoutThreshold** value is exceeded.

### backoutThreshold

Integer.

Specifies the number of times that a message can be backed out before it is transferred to the backout queue that is specified by the **backoutRequeueQueueName** attribute.

If the **backoutThreshold** value is later reduced, messages that are already on the queue that were backed out at least as many times as the new value remain on the queue. Those messages are transferred if they are backed out again.

The value must be a value in the range 0 - 999,999,999.



### custom

String.

Specifies custom attributes for new features.

This attribute contains the values of attributes, as pairs of attribute name and value, which are separated by at least one space. The attribute name-value pairs have the form `NAME (VALUE)`. Single quotation marks, `'`, must be escaped with another single quotation mark.

### enableMediaImageOperations

  This attribute is available only on the IBM MQ Appliance, AIX, Linux, and Windows.

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used.

String.

The value must be one of the following values:

#### yes

Specifies that this queue object is recoverable.

#### no

The `rcdmqimg` and `rcrmqobj` commands are not permitted for these objects. If automatic media images are enabled, the media images are not written for these objects.

#### asQmgr

Specifies that the queue inherits the value from the queue manager `ImageRecoverQueue` attribute.

This is the default value for this attribute.

### hardenGetBackout

 This attribute is only available on z/OS.

Boolean.

Specifies whether the count of the number of times that a message was backed out is saved, to ensure that it is accurate across restarts of the queue manager.

If the value is set to `true`, the backout count is always accurate across restarts of the queue manager.

### supportDistributionLists

**MQ Appliance** **ALW** This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

Boolean.

Specifies whether distribution-list messages can be placed on the queue.

If the value is set to `true`, distribution lists can be placed on the queue.

### PATCH

Use the HTTP PATCH method with the queue resource to modify a queue on a specified queue manager.

#### Note:

- This resource URL is available only in version 1 of the REST API. To modify queues using version 2 of the REST API, use the [“/admin/action/qmgr/{qmgrName}/mqsc”](#) on page 2159 resource.
- **V9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

This REST API command is similar to the “Change, Copy, and Create Queue” on page 1064 PCF command, and the “ALTER queues (alter queue settings)” on page 382 MQSC commands.

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers”](#) on page 2333
- [Request body format](#)
- [“Security requirements”](#) on page 2334
- [Response status codes](#)
- [“Response headers”](#) on page 2335
- [Response body format](#)
- [Examples](#)

### Resource URL

`https://host:port/ibmmq/rest/v1/admin/qmgr/{qmgrName}/queue/{queueName}`

#### qmgrName

Specifies the name of the queue manager on which the queue to modify exists.

The queue manager name is case sensitive.

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A period (.) must be encoded as %2E.
- A percent sign (%) must be encoded as %25.

#### queueName

Specifies the name of the queue to modify.

You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

The queue manager name is case-sensitive.

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.
- A period (.) must be encoded as %2E.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Optional query parameters

### **commandScope=scope**

 This parameter is only available on z/OS.

Specifies how the command is run when the queue manager is a member of a queue sharing group. You cannot specify this parameter if the queue manager is not a member of a queue sharing group. *scope* can be one of the following values:

#### **The name of a queue manager**

Specifies that the command is run on the queue manager that is named. The queue manager must be active within the same queue sharing group as the queue manager that is specified in the resource URL.

You cannot specify the queue manager name that is the queue manager that is specified in the resource URL.

If the queue manager name includes a percent sign, %, this character must be URL encoded as %25.

\*

Specifies that the command is run on the local queue manager and also passed to every active queue manager in the queue sharing group.

If this option is used, an `ibm-mq-qmgrs` response header is returned with a comma-separated list of the queue managers that generated a response. For example, the header might look like the following header:

```
ibm-mq-qmgrs: MQ21, MQ22
```

### **force**

Specifies that the command is forced to complete, regardless of whether completing affects an open queue.

This parameter is not valid for model queues.

An open queue is affected in the following cases:

- The queue is an alias queue. The **targetName** is modified, and an application has the alias queue open.
- The queue is a local queue. The **allowedSharedInput** attribute is modified, and more than one application has the queue open for input.
- The queue is a local queue. The **isTransmissionQueue** attribute is modified, and messages are on the queue, or applications have the queue open.
- The queue is a remote queue. The **transmissionQueueName** attribute is modified, and an application has a remote queue open that would be affected by this change.
- The queue is remote queue. The **queueName**, **qmgrName**, or **transmissionQueueName** attributes are modified, and one or more applications has a queue open that resolved through this definition as a queue manager alias.

## Request headers

The following headers must be sent with the request:

### Content-Type

This header must be sent with a value of `application/json` optionally followed by `; charset=UTF-8`.

### ibm-mq-rest-csrf-token

This header must be set, but the value can be anything, including being blank.

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### ibm-mq-rest-gateway-qmgr

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## Request body format

The request body must be in JSON format in UTF-8 encoding. Within the request body attributes are specified, and named JSON objects are created to specify extra attributes to modify. Any attributes that are not specified are not changed.

For example, the following JSON contains the attribute **type**, and then the named JSON objects, **events** and **storage**. The named JSON objects define the additional attributes to modify the queue to disable queue depth high events, and change the maximum queue depth to 2000:

```
{
  "type": "local",
  "events" : {
    "serviceInterval" : {
      "highEnabled" : false,
      "okEnabled" : false
    }
  },
  "storage" : {
    "maximumDepth" : 2000
  }
}
```

For more examples, see [examples](#).

The following attributes can be included in the request body:

### type

String.

Specifies the type of queue.

The value can be one of the following values:

- `local`
- `alias`
- `model`
- `remote`

The default value is `local`.

The following objects can be included in the request body to specify extra attributes:

### remote

Contains attributes that are related to remote queues. The attributes in this object are supported only for remote queues.

**alias**

Contains attributes that are related to alias queues. The attributes in this object are supported only for alias queues.

**model**

Contains attributes that are related to model queues. The attributes in this object are supported only for model queues.

**cluster**

Contains attributes that are related to clusters.

**trigger**

Contains attributes that are related to triggering.

**events**

Contains two objects, one for queue depth and one for queue service interval events. Each object contains attributes that are related to the event type.

**applicationDefaults**

Contains attributes that are related to default behavior such as message persistence, message priority, shared input settings, and read ahead settings.

**queueSharingGroup**

Contains attributes that are related to queue sharing groups on z/OS.

**dataCollection**

Contains attributes that are related to data collection, monitoring, and statistics.

**storage**

Contains attributes that are related to message storage, such as the maximum depth of the queue, and the maximum length of messages that are allowed on the queue.

**general**

Contains attributes that are related to general queue properties, such as whether get or put operations are inhibited, the description of the queue, and transmission queue settings.

**extended**

Contains attributes that are related to extended queue properties, such as backout queue settings, and shared input settings.

For more information, see [“Request body attributes for queues” on page 2318](#).


## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

The security principal of the caller must be granted the ability to issue the following PCF commands for the specified queue manager:

- For the queue that is specified by the *{queueName}* portion of the resource URL, authority to issue the **MQCMD\_CHANGE\_Q** PCF command must be granted.

 On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the **setmqaut** command. For more information, see [setmqaut \(grant or revoke authority\)](#).

 On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 204

Queue modified successfully.

### 400

Invalid data provided.

For example, invalid queue data is specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. The `ibm-mq-rest-csrf-token` header must also be specified. For more information, see [“Security requirements” on page 2334](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources. For more information about the access that is required, see [“Security requirements” on page 2334](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Queue does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

The following headers are returned with the response:

### **z/OS** `ibm-mq-qmgrs`

On z/OS, if the optional query parameter `commandScope=*` is used, this header is returned with a comma-separated list of the queue managers that generated a response. For example, the header might look like the following header:

```
ibm-mq-qmgrs: MQ21, MQ22
```

If an error occurs before the command is issued to the queue managers, the response header does not contain the list of queue managers. For example, a request that generates a 200 or 201 status code has the header because the command was successful. A request that generates a 401 (not authenticated) status code does not have the header because the request was rejected. A request that generates a 403 (not authorized) status code has the header because individual queue managers decide whether the command is authorized.

### `ibm-mq-rest-gateway-qmgr`

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

## Response body format

The response body is empty if the queue is modified successfully. If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

## Examples

- The following example modifies an alias queue called `aliasQueue`. The following URL is used with the HTTP PATCH method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/aliasQueue
```

The following JSON payload is sent:

```
{
  "type": "alias",
  "alias": {
    "targetName": "aDifferentLocalQueue"
  }
}
```


## GET

Use the HTTP GET method with the queue resource to request information about queues.

### Note:

- This resource URL is available only in version 1 of the REST API. To request information about queues using version 2 of the REST API, use the [“/admin/action/qmgr/{qmgrName}/mqsc”](#) on page 2159 resource.
- **V 9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

The information that is returned is similar to the information returned by the [“MQCMD\\_INQUIRE\\_Q \(Inquire Queue\)”](#) on page 1313 and [“MQCMD\\_INQUIRE\\_Q\\_STATUS \(Inquire Queue Status\)”](#) on page 1385 PCF commands, and the [“DISPLAY QUEUE \(display queue attributes\)”](#) on page 801 and [“DISPLAY QSTATUS \(display queue status\)”](#) on page 789 MQSC commands.

**Note:**  On z/OS, the channel initiator must be running before you use the queue resource with the HTTP GET method in either of the following situations:

- The **type** optional query parameter is not specified.
- The **type** optional query parameter is specified as either `all` or `cluster`.
- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers”](#) on page 2342
- [Request body format](#)
- [“Security requirements”](#) on page 2342
- [Response status codes](#)
- [“Response headers”](#) on page 2343
- [Response body format](#)
- [Examples](#)

## Resource URL

```
https://host:port/ibmmq/rest/v1/admin/qmgr/{qmgrName}/queue/{queueName}
```

### qmgrName

Specifies the name of the queue manager on which to query the queues.



You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

The queue manager name is case-sensitive.

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.
- A period (.) must be encoded as %2E.

#### **queueName**

Optionally specifies the name of a queue that exists on the queue manager specified.

The queue name is case-sensitive.

If the queue name includes a forward slash or a percent sign, these characters must be URL encoded:

- A forward slash, /, must be encoded as %2F.
- A percent sign, %, must be encoded as %25.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## **Optional query parameters**

**attributes={*object*,...|\*|*object.attributeName*,...}**

#### **object,...**

Specifies a comma-separated list of JSON objects that contain related queue configuration attributes to return.

For example, to return all queue configuration attributes that are related to time stamps, specify `timestamps`. To return all queue configuration attributes that are related to storage and to data collection, specify `storage,dataCollection`.

The `status` and `applicationHandle` objects cannot be specified with this query parameter. Use the **status** and **applicationHandle** query parameters to return these attributes.

You cannot specify the same object more than once. If you request objects that are not valid for a particular queue, the attributes are not returned for that queue. However, if you specify a value for the **type** parameter that is not `all`, and request objects that are not valid for that queue type, an error is returned.

For a full list of objects and associated attributes, see [Attributes for queues](#).

#### **\***

Specifies all attributes.

#### **object.attributeName,...**

Specifies a comma-separated list of queue configuration attributes to return.

Each attribute must specify the JSON object that contains the attribute, in the form `object.attributeName`. For example, to return the `maximumDepth` attribute, which is contained in the `storage` object, specify `storage.maximumDepth`.

Attributes from the `status` and `applicationHandle` objects cannot be specified with this query parameter. Use the **status** and **applicationHandle** query parameters to return these attributes.

You cannot specify the same attribute more than once. If you request attributes that are not valid for a particular queue, the attributes are not returned for that queue. However, if you specify the **type** parameter and request attributes that are not valid for that queue type, an error is returned.

For a full list of attributes and associated objects, see [Attributes for queues](#).

**status={status|\*|status.attributeName,...}**

**status**

Specifies that all status attributes are returned.

**\***

Specifies all attributes. This parameter is equivalent to **status**.

**status.attributeName,...**

Specifies a comma-separated list of status attributes to return.

For example, to return the `currentDepth` attribute, specify `status.currentDepth`.

For a full list of status attributes, see [Status attributes for queues](#).

If you specify the **status** optional query parameter, you can specify the **type** parameter only with the `all` or `local` values. You cannot specify the **queueSharingGroupDisposition** parameter with the `group` value.

**applicationHandle={applicationHandle|\*|applicationHandle.attributeName,...}**

**applicationHandle**

Specifies that all application handle attributes are returned.

**\***

Specifies all attributes. This parameter is equivalent to **applicationHandle**.

**applicationHandle.attributeName,...**

Specifies a comma-separated list of application handle attributes to return.

For example, to return the `handleState` attribute, specify `applicationHandle.handleState`.

For a full list of application handle attributes, see [Application handle attributes for queues](#).

If you specify the **applicationHandle** optional query parameter, you can specify the **type** parameter only with the `all` or `local` values. You cannot specify the **queueSharingGroupDisposition** parameter with the `group` value.

**commandScope=scope**

 This parameter is only available on z/OS.

Specifies how the command is run when the queue manager is a member of a queue sharing group.

You cannot specify this parameter if the queue manager is not a member of a queue sharing group.

*scope* can be one of the following values:

**The name of a queue manager**

Specifies that the command is run on the queue manager that is named. The queue manager must be active within the same queue sharing group as the queue manager that is specified in the resource URL.

You cannot specify the queue manager name that is the queue manager that is specified in the resource URL.

If the queue manager name includes a percent sign, %, this character must be URL encoded as %25.

**\***

Specifies that the command is run on the local queue manager and also passed to every active queue manager in the queue sharing group.

If this option is used, an `ibm-mq-qmgrs` response header is returned with a comma-separated list of the queue managers that generated a response. For example, the header might look like the following header:

```
ibm-mq-qmgrs: MQ21, MQ22
```

**filter=filterValue**

Specifies a filter for the queue definitions that are returned.

If you specify a queue name in the resource URL, you can only filter on application handle attributes. If you filter on an application handle attribute, the only application handles returned are those that match the filter parameter.

You can specify only one filter. If you filter on an application handle attribute, you must specify the **applicationHandle** query parameter. If you filter on a status attribute, you must specify the **status** query parameter.


*filterValue* has the following format:

```
attribute:operator:value
```

where:

#### **attribute**


Specifies one of the applicable attributes. For a full list of attributes, see [Attributes for queues](#). The following attributes cannot be specified:

- name
- type
-  queueSharingGroup.disposition
- status.onQueueTime
- status.tpipeName
- applicationHandle.qmgrTransactionId
- applicationHandle.unitOfWorkId
- applicationHandle.openOptions

To filter on any attributes that are time stamps, the filter can specify any portion of the time stamp, with a trailing asterisk, \*. The format of a time stamp is, YYYY-MM-DDThh:mm:ss. For example, you can specify 2001-11-11\* to filter on dates in the range 2001-11-10 to 2001-11-19, or 2001-11-12T14:\* to filter any minute in the specified hour of the specified day.

Valid values for the YYYY section of the date are in the range 1900 - 9999.

The time stamp is a string. Therefore, only the equalTo and notEqualTo operators can be used with the time stamp.

**Note:**  If either the **filter** query parameter, or the **name** query parameter with a wildcard, are used with the **commandScope=\*** query parameter, and there are no matching queues on at least one of the active queue managers in the queue sharing group, then an error message is returned.

#### **operator**

Specifies one of the following operators:

##### **lessThan**

Use this operator only with integer attributes.

##### **greaterThan**

Use this operator only with integer attributes.

##### **equalTo**

Use this operator with any attribute.

##### **notEqualTo**

Use this operator with any attribute.

##### **lessThanOrEqualTo**

Use this operator only with integer attributes.

##### **greaterThanOrEqualTo**

Use this operator only with integer attributes.

## value

Specifies the constant value to test against the attribute.

The value type is determined by the attribute type.


For string and boolean attributes, you can omit the value field after the colon. For string attributes, omit the value to return queues with no value for the specified attribute. For boolean attributes, omit the value to return any queues that have the specified attribute set to false. For example, the following filter returns all queues where the description attribute is not specified:

```
filter=general.description:equalTo:
```

You can use a single asterisk, \*, at the end of the value as a wildcard. You cannot use only an asterisk.

If the value includes a space, a forward slash, a percent sign, or an asterisk that is not a wildcard, these characters must be URL encoded:

- A space must be encoded as %20
- A forward slash, /, must be encoded as %2F.
- A percent sign, %, must be encoded as %25.
- An asterisk, \*, must be encoded as %2A.

 If the filter query parameter is used with the **commandScope=\*** query parameter, and there are no matching values on at least one of the active queue managers in the queue sharing group, an error message is returned.

## name=name

This query parameter cannot be used if you specify a queue name in the resource URL.

Specifies a wildcard queue name to filter on.

The *name* specified must include an asterisk, \*, as a wildcard. You can specify one of the following combinations:

**\***

Specifies that all queues are returned.

**prefix\***


Specifies that all queues with the specified prefix in the queue name are returned.

**\*suffix**

Specifies that all queues with the specified suffix in the queue name are returned.

**prefix\*suffix**

Specifies that all queues with the specified prefix and the specified suffix in the queue name are returned.

 If the name query parameter is used with a wildcard, the **commandScope=\*** query parameter is specified, and there are no matching values on at least one of the active queue managers in the queue sharing group, an error message is returned.

## queueSharingGroupDisposition=disposition

 This parameter is only available on z/OS.

Specifies where the queue for which information is to be returned is defined and how it behaves. That is, it specifies the disposition of the queue for which information is to be returned.

You cannot specify the **queueSharingGroupDisposition** parameter if you specify **type=cluster** for the **type** parameter.

The value can be one of the following values:

### live

Specifies that the queue is defined as **qmgr** or **copy**.

In a shared queue manager environment, **live** also displays information for queues that are defined with **shared**.

If the **commandScope** optional query parameter is specified with the **live** option, then any queue definitions with a disposition of shared are returned only by the queue manager that received the REST request. Other queue managers in the group do not return these queue definitions.

If you specify **live** with the **attributes** parameter, and specify the **commandScope** parameter with a queue manager name, queue attributes are not returned for shared queues.

#### **all**

Specifies that the queue is defined as qmgr or copy.

In a shared queue manager environment, **all** also displays information for queues that are defined with group or shared.

If the **commandScope** optional query parameter is specified with **all**, then any queue definitions with a disposition of group or shared are returned only by the queue manager that received the REST request. Other queue managers in the group do not return these queue definitions.

If you specify **all** with the **attributes** parameter, and specify the **commandScope** parameter with a queue manager name, queue attributes are not returned for shared queues.

If you specify **all** and specify **type=all**, no cluster queues are returned.

#### **copy**

Specifies that the queue is defined as copy.

#### **group**

Specifies that the queue is defined as group.

If you specify **group**, you cannot specify the **commandScope** optional query parameter.

#### **private**

Specifies that the queue is defined as copy or qmgr.

#### **qmgr**

Specifies that the queue is defined as qmgr.

#### **shared**

Specifies that the queue is defined as shared.

You cannot specify the **commandScope** optional query parameter with this option, unless the **status** or **applicationHandle** optional query parameter is also specified.

You cannot specify this option with the **attributes** parameter if you also specify the **commandScope** parameter with a queue manager name.

If you specify **shared** and specify **type=all**, all shared queues are returned, including cluster queues with a disposition of shared.

The default value is `live`.


#### **type=type**

Specifies the type of queue to return information about.

The value can be one of the following values:

#### **all**

Specifies that information about all queues, including cluster queues, is returned.

 On z/OS, ensure that the channel initiator is running when you use this option.

#### **local**

Specifies that information about local queues is returned.

#### **alias**

Specifies that information about alias queues is returned.

#### **remote**

Specifies that information about remote queues is returned.

#### **cluster**

Specifies that information about cluster queues is returned.

**z/OS** You cannot specify **type=cluster** if you specify the **queueSharingGroupDisposition** parameter.

**z/OS** On z/OS, ensure that the channel initiator is running when you use this option.

### **model**

Specifies that information about model queues is returned.

The default value is all.

## **Request headers**

The following headers must be sent with the request:

### **Authorization**

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### **ibm-mq-rest-gateway-qmgr**

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## **Request body format**

None.

## **Security requirements**

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

The security principal of the caller must be granted the ability to issue the following PCF commands for the specified queue manager:

- If the **status** or **applicationHandle** query parameters are not specified:
  - For the queue that is specified by the *{queueName}* portion of the resource URL, or for queues that match the specified query parameters, authority to issue the **MQCMD\_INQUIRE\_Q** PCF command must be granted.
- If the **status** or **applicationHandle** query parameters are specified:
  - For the queue that is specified by the *{queueName}* portion of the resource URL, or for queues that match the specified query parameters, authority to issue the **MQCMD\_INQUIRE\_Q** PCF command must be granted.
  - For the queue that is specified by the *{queueName}* portion of the resource URL, or for queues that match the specified query parameters, authority to issue the **MQCMD\_INQUIRE\_QSTATUS** PCF command must be granted.

A principal has display authority if the principal can issue one or both of the **MQCMD\_INQUIRE\_Q** and **MQCMD\_INQUIRE\_QSTATUS** PCF commands. If the principal has display authority for only some of the queues that are specified by the resource URL and query parameters, then the array of queues that is returned from the REST request is limited to those queues that the principal has authority to display. No information is returned about queues that cannot be displayed. If the principal does not have display authority for any of the queues that are specified by the resource URL and query parameters, an HTTP status code of 403 is returned.

**ALW** On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the **setmqaut** command. For more information, see [setmqaut \(grant or revoke authority\)](#).

**z/OS** On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 200

Queue information retrieved successfully.

### 400

Invalid data provided.

For example, invalid queue attributes specified.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information, see [“Security requirements” on page 2342](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources. For more information about the access that is required, see [“Security requirements” on page 2342](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Queue does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

The following headers are returned with the response:

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

### **z/OS** **ibm-mq-qmgrs**

On z/OS, if the optional query parameter `commandScope=*` is used, this header is returned with a comma-separated list of the queue managers that generated a response. For example, the header might look like the following header:

```
ibm-mq-qmgrs: MQ21, MQ22
```

If an error occurs before the command is issued to the queue managers, the response header does not contain the list of queue managers. For example, a request that generates a 200 or 201 status code has the header because the command was successful. A request that generates a 401 (not authenticated) status code does not have the header because the request was rejected. A request that generates a 403 (not authorized) status code has the header because individual queue managers decide whether the command is authorized.

**ibm-mq-rest-gateway-qmgr**

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

**Response body format**

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `queue`. Each element in the array is a JSON object that represents information about a queue. Each of these JSON objects contains the following attributes:

**name**

String.

Specifies the name of the queue.

This attribute is always returned.

**type**

String.

Specifies the type of queue.

The value is one of the following values:

- `local`
- `alias`
- `remote`
- `cluster`
- `model`

This attribute is always returned.

The following objects can be included in the JSON object that represents information about a queue. Which objects and attributes are returned depends on the URL that was specified for the request:

**remote**

Contains attributes that are related to remote queues.

**alias**

Contains attributes that are related to alias queues.

**dynamic**

Contains attributes that are related to dynamic queues.

**model**

Contains attributes that are related to model queues.

**cluster**

Contains attributes that are related to clusters.

**trigger**

Contains attributes that are related to triggering.

**events**

Contains two objects, one for queue depth and one for queue service interval events. Each object contains attributes that are related to the event type.

**applicationDefaults**

Contains attributes that are related to default behavior such as message persistence, message priority, shared input settings, and read ahead settings.

**queueSharingGroup**

Contains attributes that are related to queue sharing groups on z/OS.

**dataCollection**

Contains attributes that are related to data collection, monitoring, and statistics.



### storage

Contains attributes that are related to message storage, such as the maximum depth of the queue, and the maximum length of messages that are allowed on the queue.

### general

Contains attributes that are related to general queue properties, such as whether get or put operations are inhibited, the description of the queue, and transmission queue settings.

### extended

Contains attributes that are related to extended queue properties, such as backout queue settings, and shared input settings.

### timestamps

Contains attributes that are related to date and time information, such as the time stamp of when a queue was created.

### status

Contains attributes that are related to queue status information.

### applicationHandle

Contains attributes that are related to application handle information.

If a queue has no application handles, but information about application handles is requested, an empty object is returned.

For more information, see [“Response body attributes for queues”](#) on page 2348.

If a damaged object is found, and the REST request did not specify a queue, an extra JSON array that is called `damaged` is returned. This JSON array contains a list of the objects that are damaged, specifying the object names. If the REST request specifies a queue name within the resource URL, but the object is damaged, an error is returned.

If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

## Examples

**Note:** Information about the `SYSTEM.*` queues is returned. It is expected that all queues are returned. However, for brevity, the results shown in the following examples do not include all the expected results.

- The following example lists all queues on the queue manager `QM1`. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue
```

The following JSON response is returned:

```
{
  "queue": [
    {
      "name": "localQueue",
      "type": "local"
    },
    {
      "name": "remoteQueue",
      "type": "remote",
      "remote": {
        "queueName": "queueOnQM1",
        "qmgrName": "QM1"
      }
    },
    {
      "name": "aliasQueue",
      "type": "alias",
      "alias": {
        "targetName": "localQueue"
      }
    },
    {
      "name": "modelQueue",
      "type": "model",
      "model": {
        "type": "permanentDynamic"
      }
    }
  ]
}
```

```

    }, {
      "name": "permanentDynamicQueue",
      "type": "local",
      "dynamic": {
        "type": "permanentDynamic"
      }
    }, {
      "name": "aliasQueue2",
      "type": "cluster",
      "cluster": {
        "name": "CLUSTER1",
        "qmgrName": "QM2",
        "queueType": "alias"
      }
    }
  ]
}

```

- The following example lists all local queues on the queue manager QM1, showing whether they are get or put enabled. The following URL is used with the HTTP GET method:

```

https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QMGR2/queue?
type=local&attributes=general.inhibitPut,general.inhibitGet

```

The following JSON response is returned:

```

{
  "queue":
  [
    {
      "name": "localQueue",
      "type": "local",
      "general": {
        "inhibitPut": true,
        "inhibitGet": false,
      }
    },
    {
      "name": "permanentDynamicQueue",
      "type": "local",
      "dynamic": {
        "type": "permanentDynamic"
      },
      "general": {
        "inhibitPut": false,
        "inhibitGet": false,
      }
    }
  ]
}

```

- The following example lists the status attributes for the queue Q1, on queue manager QM1. The following URL is used with the HTTP GET method:

```

https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/Q1?status=*

```

The following JSON response is returned:

```

{
  "queue":
  [
    {
      "name": "Q1",
      "status": {
        "currentDepth": 0,
        "lastGet": "2016-12-05T15:56:28.000Z",
        "lastPut": "2016-12-05T15:56:28.000Z",
        "mediaRecoveryLogExtent": "",
        "oldestMessageAge": 42,
        "onQueueTime": {
          "longSamplePeriod": 3275,
          "shortSamplePeriod": 3275
        },
        "openInputCount": 1,
        "openOutputCount": 1,
        "uncommittedMessages": 2
      },
      "type": "local"
    }
  ]
}

```

- The following example lists the application handle attributes for a queue Q1, on queue manager QM1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/Q1?applicationHandle=*
```

The following JSON response is returned:

```
{
  "queue":
  [{
    "applicationHandle":
    [{
      "asynchronousState": "none",
      "channelName": "",
      "connectionName": "",
      "description": "",
      "state": "inactive",
      "openOptions": [
        "MQOO_INPUT_SHARED",
        "MQOO_BROWSE",
        "MQOO_INQUIRE",
        "MQOO_SAVE_ALL_CONTEXT",
        "MQOO_FAIL_IF QUIESCING"
      ],
      "processID": 9388,
      "qmgrTransactionID": "AAAAAAhAAAA=",
      "recoveryID": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA====",
      "tag": "IBM\\Java70\\jre\\bin\\javaw.exe",
      "threadID": 0,
      "transactionType": "qmgr",
      "type": "userApplication",
      "userID": "myID"
    },
    {
      "asynchronousState": "none",
      "channelName": "",
      "connectionName": "",
      "description": "",
      "state": "inactive",
      "openOptions": [
        "MQOO_OUTPUT",
        "MQOO_FAIL_IF QUIESCING"
      ],
      "processID": 9388,
      "qmgrTransactionID": "AAAAAAhAAAA=",
      "recoveryID": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA====",
      "tag": "IBM\\Java70\\jre\\bin\\javaw.exe",
      "threadID": 0,
      "transactionType": "qmgr",
      "type": "userApplication",
      "userID": "myID"
    }
  ]
},
{
  "name": "Q1",
  "type": "local"
}
}]
}
```

- The following example shows how to get all information, including status and application handles, for the queue Q2 on queue manager QM1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/Q2?
attributes=*&status=*&applicationHandle=*
```

- The following example shows how to get all queue configuration and status information for queues with an **openInputCount** greater than three, for the queue manager QM1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue?
attributes=*&status=*&filter=status.openInputCount:greaterThan:3
```

### *Response body attributes for queues*

When you use the HTTP GET method with the queue object to request information about queues, the following attributes are returned within named JSON objects.

The following objects are available:

- [“remote” on page 2348](#)
- [“alias” on page 2348](#)
- [“dynamic” on page 2349](#)
- [“model” on page 2349](#)
- [“cluster” on page 2350](#)
- [“trigger” on page 2351](#)
- [“events” on page 2352](#)
- [“applicationDefaults” on page 2353](#)
- [“queueSharingGroup” on page 2354](#)
- [“dataCollection” on page 2355](#)
- [“storage” on page 2356](#)
- [“general” on page 2357](#)
- [“extended” on page 2358](#)
- [“timestamps” on page 2359](#)
- [“status” on page 2359](#)
- [“applicationHandle” on page 2361](#)

For more information about the PCF equivalents to the queue REST API parameters and attributes, see [“REST API and PCF equivalents for queues” on page 2380](#).

## **remote**

The `remote` object contains information about remote queues and is returned only for remote queues:

### **qmgrName**

String.

Specifies the name of the remote queue manager.

If this remote queue is used as a queue manager alias, this attribute is the name of the queue manager.

If this remote queue is used as a reply-to queue alias, this attribute is the name of the queue manager that is to be the reply-to queue manager.

This attribute is always returned.

### **queueName**

String.

Specifies the name of the queue as it is known on the remote queue manager.

This attribute is always returned.

### **transmissionQueueName**

String.

Specifies the name of the transmission queue that is used for messages that are destined for either a remote queue or for a queue manager alias definition.

## **alias**

The `alias` object contains information about alias queues and is returned only for alias queues:

**targetName**

String.

Specifies the name of the queue or topic that the alias resolves to.

This attribute is always returned.

**targetType**

String.

Specifies the type of object that the alias resolves to.

The value is one of the following values:

**queue**

Specifies that the object is a queue.

**topic**

Specifies that the object is a topic.

**dynamic**

The `dynamic` object contains information about dynamic queues and is returned only for local queues that are programmatically created from a model queue:

**type**

String.

Specifies the type of dynamic queue.


This attribute is always returned.

The value is one of the following values:

**permanentDynamic**

Specifies that the queue is a dynamically defined permanent queue.

**sharedDynamic**

 This attribute is only available on z/OS.

Specifies that the queue is a dynamically defined shared queue.

**temporaryDynamic**

Specifies that the queue is a dynamically defined temporary queue.

**model**

The `model` object contains information about model queues and is returned only for model queues:

**type**

String.

Specifies the model queue definition type.

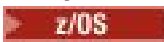
This attribute is always returned.

The value is one of the following values:

**permanentDynamic**

Specifies that the queue is a dynamically defined permanent queue.

**sharedDynamic**

 This attribute is only available on z/OS.

Specifies that the queue is a dynamically defined shared queue.

**temporaryDynamic**

Specifies that the queue is a dynamically defined temporary queue.

## cluster

The `cluster` object contains information about queues that are part of one or more clusters. The object is returned only for queues when `type=cluster` is specified, or if requested by the attributes query parameter:

### **name**

String.

Specifies the name of the cluster that the queue belongs to.

This attribute, or the **namelist** attribute, is always returned.

### **namelist**

String.

Specifies the namelist that lists the clusters that the queue belongs to.

This attribute, or the **name** attribute, is always returned.

### **qmgrId**

String.

Specifies the unique identifier of the queue manager.

This attribute is returned only when `type=cluster` is specified.

### **qmgrName**

String.

Specifies the name of the local queue manager.

This attribute is returned only when `type=cluster` is specified.

### **queueType**

String.

Specifies the type of queue.

This attribute is returned only when `type=cluster` is specified.

The value is one of the following values:

#### **local**

Specifies that the cluster queue represents a local queue.

#### **alias**

Specifies that the cluster queue represents an alias queue.

#### **remote**

Specifies that the cluster queue represents a remote queue.

#### **qmgrAlias**

Specifies that the cluster queue represents a queue manager alias.

### **transmissionQueueForChannelName**

String.

Specifies the generic name of the cluster-sender channels that use the queue as a transmission queue. The attribute specifies which cluster-sender channels send messages to a cluster-receiver channel from the cluster transmission queue.

### **workloadPriority**

Integer.

Specifies the priority of the queue in cluster workload management.

A value of 0 specifies the lowest priority and 9 specifies the highest priority.

### **workloadQueueUse**

String.

Specifies whether remote and local instances of the clustered queues are used in cluster workload distribution.

The value is one of the following values:

**asQmgr**

Use the value that is defined on the queue manager.

**any**

Use remote and local instances of the queues.

**local**

Use only local instances of the queues.

**workloadRank**

Integer.

Specifies the rank of the queue in cluster workload management.

A value of 0 specifies the lowest priority and 9 specifies the highest priority.

**trigger**

The `trigger` object contains information about triggering:

**enabled**

Boolean.

Specifies whether trigger messages are written to the initiation queue.

**data**

String.

Specifies the user data that is included in the trigger message.

**depth**

Integer.

Specifies the number of messages that initiates a trigger message to the initiation queue.

**initiationQueueName**

String.

Specifies the local queue for trigger messages that relate to the queue.

**messagePriority**

Integer.

Specifies the minimum priority that a message must have before it can cause, or be counted for, a trigger event.

**processName**

String.

Specifies the local name of the IBM MQ process that identifies the application to be started when a trigger event occurs.

If the queue is a transmission queue, the process definition contains the name of the channel to be started.

**type**

String.

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

The value is one of the following values:

**none**

Send no trigger messages.

**every**

Send a trigger message for every message that arrives on the queue.

**first**

Send a trigger message when the queue depth goes from 0 to 1.

**depth**

Send a trigger message when the queue depth exceeds the value of the **depth** attribute.

## events

The events object contains two objects, one for queue depth and one for queue service interval events. Each object contains attributes that are related to the event type:

### depth

JSON object.

A JSON object that can contain the following attributes that related to queue depth events:

#### **highEnabled**

Boolean.

Specifies whether queue depth high events are generated.

A queue depth high event indicates that the number of messages on the queue is greater than or equal to the queue depth high limit, **highPercentage**.

#### **highPercentage**

Integer.

Specifies the threshold against which the queue depth is compared to generate a queue depth high event.

This value is expressed as a percentage of the maximum queue depth.

#### **lowEnabled**

Boolean.

Specifies whether queue depth low events are generated.

A queue depth low event indicates that the number of messages on the queue is less than or equal to the queue depth low limit, **lowPercentage**.

#### **lowPercentage**

Integer.

Specifies the threshold against which the queue depth is compared to generate a queue depth low event.

This value is expressed as a percentage of the maximum queue depth.

#### **fullEnabled**

Boolean.

Specifies whether queue full events are generated.

A queue full event indicates that no more messages can be put on a queue because the queue is full. That is, the queue depth reached the maximum queue depth.

### serviceInterval

JSON object.

A JSON object that can contain the following attributes that are related to queue service interval events:

#### **highEnabled**

Boolean.

Specifies whether queue service interval high events are generated.

A queue service interval high event is generated when no messages were put to, or retrieved from, the queue for at least the amount of time specified by the **duration** attribute.

#### **okEnabled**

Boolean.

Specifies whether queue service interval OK events are generated.

A queue service interval OK event is generated when a message was retrieved from the queue within the amount of time that is specified by the **duration** attribute.

#### **duration**

Integer.



Specifies the service interval duration, in milliseconds, that is used to generate queue service interval high and queue service interval OK events.

## **applicationDefaults**

The `applicationDefaults` object contains attributes that are related to default behavior such as message persistence, message priority, shared input settings, and read ahead settings:

### **clusterBind**

String.

Specifies the binding to be used when `MQOO_BIND_AS_Q_DEF` is specified on the `MQOPEN` call.

The value is one of the following values:

#### **onOpen**

Specifies that the binding is fixed by the `MQOPEN` call.

#### **notFixed**

Specifies that the binding is not fixed.

#### **onGroup**

Specifies that the application can request that a group of messages is allocated to the same destination instance.

### **messagePropertyControl**

String.

Specifies how message properties are handled when messages are retrieved from queues when `MQGMO_PROPERTIES_AS_Q_DEF` is specified on the `MQGET` call.

This attribute is applicable to local, alias, and model queues.

The value is one of the following values:

#### **all**

Specifies that all properties of the message are included when the message is sent to the remote queue manager. The properties, except those properties in the message descriptor or extension, are placed in one or more `MQRFH2` headers in the message data.

#### **compatible**

Specifies that if the message contains a property with the prefix `mcd.`, `jms.`, `usr.`, or `mqext.`, all message properties are delivered to the application in an `MQRFH2` header. Otherwise, all properties, except those properties in the message descriptor or extension, are discarded and are no longer accessible.

#### **force**

Specifies that properties are always returned in the message data in an `MQRFH2` header regardless of whether the application specifies a message handle. A valid message handle that is included in the `MsgHandle` field of the `MQGMO` structure on the `MQGET` call is ignored. Properties of the message are not accessible by using the message handle.

#### **none**

Specifies that all properties of the message are removed from the message before the message is sent to the remote queue manager. Properties in the message descriptor, or extension, are not removed.

#### **version6Compatible**

Any application `MQRFH2` header is received as it was sent. Any properties set by using `MQSETMP` must be retrieved by using `MQINQMP`. They are not added to the `MQRFH2` created by the application. Properties that were set in the `MQRFH2` header by the sending application cannot be retrieved by using `MQINQMP`.

### **messagePersistence**

String.

Specifies the default for message persistence on the queue. Message persistence determines whether messages are preserved across restarts of the queue manager.

The value is one of the following values:

**persistent**

Specifies that the messages on the queue are persistent, and are preserved when the queue manager restarts.

**nonPersistent**

Specifies that the messages on the queue are not persistent, and are lost when the queue manager restarts.

**messagePriority**

Integer.

Specifies the default priority of messages that are put on the queue.

**putResponse**

String.

Specifies the type of response that is used for put operations to the queue when an application specifies MQPMO\_RESPONSE\_AS\_Q\_DEF.

The value is one of the following values:

**synchronous**

The put operation is run synchronously, returning a response.

**asynchronous**

The put operation is run asynchronously, returning a subset of MQMD fields.

**readAhead**

String.

Specifies the default read-ahead behavior for non-persistent messages that are delivered to the client.

The value is one of the following values:

**no**

Specifies that non-persistent messages are not read ahead unless the client application is configured to request read ahead.

**yes**

Specifies that non-persistent messages are sent ahead to the client before an application requests them. Non-persistent messages can be lost if the client ends abnormally or if the client does not consume all the messages that it is sent.

**disabled**

Specifies that non-persistent messages are not read ahead, regardless of whether read ahead is requested by the client application.

**sharedInput**

Boolean.

Specifies the default share option for applications that open this queue for input.

If the value is set to true, queues are enabled to get messages with shared access.

## queueSharingGroup

The queueSharingGroup object contains attributes that are related to queue sharing groups on z/OS:

**disposition**

String.

 This attribute is only available on z/OS.

Specifies where the queue is defined and how it behaves. That is, it specifies the disposition of the queue.

This value is always returned if the queue manager is a member of the queue sharing group.

The value is one of the following values:

**copy**

Specifies that the queue definition exists on the page set of the queue manager that runs the command. For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.

**group**

Specifies that the queue definition exists in the shared repository.

**qmgr**

Specifies that the queue definition exists on the page set of the queue manager that runs the command. For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.


**shared**

This value is only valid for local queues.

Specifies that the queue exists in the shared repository. Messages are stored in the coupling facility and are available to any queue manager in the queue sharing group.

**qmgrName**

String.


 This attribute is only available on z/OS.

Specifies the name of the queue manager that generates the response to the REST request.

This attribute is only returned if the queue manager to which the REST request is made is part of a queue sharing group, and the **commandScope** optional query parameter is specified.

**structureName**

String.

 This attribute is only available on z/OS.

Specifies the name of the coupling facility structure where messages are stored when you used shared queues.

**dataCollection**

The `dataCollection` object contains attributes that are related to data collection, monitoring, and statistics:

**accounting**

String.

Specifies whether accounting data is collected for the queue.

The value is one of the following values:

**asQmgr**

Specifies that the queue inherits the value from the queue manager MQSC parameter ACCTQ.

**off**

Specifies that accounting data is not collected for the queue.

**on**

Specifies that accounting data is collected for the queue if the ACCTQ MQSC parameter on the queue manager is not set to none.

**monitoring**

String.

Specifies whether online monitoring data is collected, and if so, the rate at which the data is collected.

The value is one of the following values:

**off**

Specifies that online monitoring data is not collected for the queue.

**asQmgr**

Specifies that the queue inherits the value from the queue manager MONQ MQSC parameter.

**low**

Specifies that online monitoring data is collected for the queue if the MONQ MQSC parameter on the queue manager is not set to none. The rate of data collection is low.



**medium**

Specifies that online monitoring data is collected for the queue if the MONQ MQSC parameter on the queue manager is not set to none. The rate of data collection is moderate.

**high**

Specifies that online monitoring data is collected for the queue if the MONQ MQSC parameter on the queue manager is not set to none. The rate of data collection is high.

**statistics**

  This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

String.

Specifies whether statistics data is collected for the queue.

The value is one of the following values:

**asQmgr**

Specifies that the queue inherits the value from the queue manager STATQ MQSC parameter.

**off**

Specifies that statistics data is not collected for the queue.


**on**

Specifies that statistics data is collected for the queue if the STATQ MQSC parameter on the queue manager is not set to none.

**storage**

The `storage` object contains attributes that are related to message storage, such as the maximum depth of the queue, and the maximum length of messages that are allowed on the queue:

**indexType**

 This attribute is only available on z/OS.

String.

Specifies the type of index that is maintained by the queue manager to expedite MQGET operations on the queue. For shared queues, the type of index determines what type of MQGET calls can be used.

The value is one of the following values:

**none**

Specifies that there is no index. Messages are retrieved sequentially.

**correlationId**

Specifies that the queue is indexed by using correlation identifiers.

**groupId**

Specifies that the queue is indexed by using group identifiers.

**messageId**

Specifies that the queue is indexed by using message identifiers.

**messageToken**

Specifies that the queue is indexed by using message tokens.

**maximumMessageLength**

Integer.

Specifies the maximum message length that is allowed, in bytes, for messages on the queue.

**maximumDepth**

Integer.

Specifies the maximum number of messages that are allowed on the queue.

### **messageDeliverySequence**

String.

Specifies whether messages are delivered in priority order or by sequence.

The value is one of the following values:



#### **priority**

Specifies that messages are returned in priority order.

#### **fifo**

Specifies that messages are returned in first in, first out order.

### **nonPersistentMessageClass**

  This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

String.

This attribute is valid only on local and model queues.

Specifies the level of reliability that is assigned to non-persistent messages that are put to the queue.

The value is one of the following values:


#### **normal**

Specifies that non-persistent messages persist for the lifetime of the queue manager session. They are discarded if the queue manager restarts.

#### **high**

Specifies that the queue manager attempts to retain non-persistent messages for the lifetime of the queue. Non-persistent messages might still be lost if a failure occurs.


### **pageSet**

 This attribute is only available on z/OS.

Integer.

Specifies the ID of the page set.

### **storageClass**

 This attribute is only available on z/OS.

String.

Specifies the name of the storage class.

## **general**

The `general` object contains attributes that are related to general queue properties, such as whether get or put operations are inhibited, the description of the queue, and transmission queue settings:

### **description**

String.

Specifies the description of the queue.

### **inhibitGet**

Boolean.

Specifies whether get operations are allowed on the queue.

If the value is set to `true`, get operations are not allowed on the queue.

### **inhibitPut**

Boolean.

Specifies whether put operations are allowed on the queue.

If the value is set to `true`, put operations are not allowed on the queue.

### **isTransmissionQueue**

String.

Specifies whether the queue is for normal usage or for transmitting messages to a remote queue manager.

If the value is set to `true`, the queue is a transmission queue for transmitting messages to a remote queue manager.

## extended

The extended object contains attributes that are related to extended queue properties, such as backout queue settings, and shared input settings:

### allowSharedInput

Boolean.

Specifies whether multiple instances of applications can open the queue for input.

If the value is set to `true`, multiple instances of applications can open the queue for input.

### backoutRequeueQueueName

String.

Specifies the name of the queue to which a message is transferred if it is backed out more times than the value of **backoutThreshold**.

### backoutThreshold

Integer.



Specifies the number of times that a message can be backed out before it is transferred to the backout queue that is specified by the **backoutRequeueQueueName** attribute.

### custom

String.

Specifies custom attributes for new features.

### enableMediaImageOperations

  This attribute is available only on the IBM MQ Appliance, AIX, Linux, and Windows.

Specifies whether a local or permanent dynamic queue object is recoverable from a media image, if linear logging is being used.

String.

The value is one of the following values:

#### yes

Specifies that this queue object is recoverable.

#### no

The `rcdmqimg` and `rcrmqobj` commands are not permitted for these objects. If automatic media images are enabled, the media images are not written for these objects.

#### asQmgr

Specifies that the queue inherits the value from the queue manager `ImageRecoverQueue` attribute.

This is the default value for this attribute.

### hardenGetBackout

 This attribute is only available on z/OS.

Boolean.

Specifies whether the count of the number of times that a message was backed out is saved, to ensure that it is accurate across restarts of the queue manager.

If the value is set to `true`, the backout count is always accurate across restarts of the queue manager.

## supportDistributionLists

MQ Appliance ALW

This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

Boolean.

Specifies whether distribution-list messages can be placed on the queue.

If the value is set to `true`, distribution lists can be placed on the queue.

## timestamps

The `timestamps` object contains attributes that are related to date and time information.

### altered

String.

Specifies the date and time at which the queue was last altered.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

### clustered

String.

Specifies the date and time at which the information became available to the local queue manager.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

### created

String.

Specifies the date and time at which the queue was created.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

## status

The `status` object contains attributes that are related to queue status information:

### currentDepth

Integer.

Specifies the current queue depth.

### lastGet

String.

Specifies the date and time at which the last message was destructively read from the queue.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

### lastPut

String.

Specifies the date and time at which the last message was successfully put to the queue.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

## mediaRecoveryLogExtent

MQ Appliance ALW

This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

String.

Specifies the name of the oldest log extent that is required to perform media recovery of the queue.

The name that is returned is of the form `Snnnnnnn.LOG` and is not a fully qualified path name.

**oldestMessageAge**

Integer.

Specifies the age, in seconds, of the oldest message on the queue.

If the queue is empty, 0 is returned. If the value is greater than 999 999 999, it is returned as 999 999 999. If no data is available, -1 is returned.

**onQueueTime**

JSON object.

A JSON object that can contain the following attributes that related to the amount of time that a message remains on the queue:

**longSamplePeriod**

Integer.

Specifies an indication of the time, in microseconds, that a message remains on the queue based on activity over a long period.

**shortSamplePeriod**

Integer.

Specifies an indication of the time, in microseconds, that a message remains on the queue based on activity over a short period.

This attribute cannot be used to filter results.

**openInputCount**

Integer.

Specifies the number of handles that are currently valid for removing messages from the queue by using the MQGET call.

**openOutputCount**

Integer.

Specifies the number of handles that are currently valid for putting messages to the queue by using the MQPUT call.

**monitoringRate**

String.

Specifies the rate at which monitoring data is collected for the queue.

The value is one of the following values:

**off**

Specifies that no data is collected.

**low**

Specifies a low rate of data collection.


**medium**

Specifies a medium rate of data collection.

**high**

Specifies a high rate of data collection.

**tpipeName**

 This attribute is only available on z/OS.

Array.

Specifies the TPIPE names that are used for communication with OTMA by using the IBM MQ IMS bridge, if the bridge is active.

This attribute cannot be used to filter results.

**uncommittedMessages**

Integer.

Specifies the number of uncommitted changes that are pending for the queue.



On z/OS, the value can be only either 0 or 1. A value of 1 indicates that there is at least one uncommitted message on the queue.

## **applicationHandle**


The `applicationHandle` object contains attributes that are related to application handle information:

### **description**

String.

Specifies a description for the application.

### **tag**

 This attribute is only available on z/OS.

String.

Specifies the tag of the open application.

### **type**

String.

Specifies the type of application.

This value is one of the following values:

#### **queueManagerProcess**

Specifies that the open application is a queue manager process.


#### **channelInitiator**

Specifies that the open application is a channel initiator.

#### **userApplication**


Specifies that the open application is a user application.

#### **batchConnection**

 This attribute is only available on z/OS.


Specifies that the open application is using a batch connection.

#### **rrsBatchConnection**

 This attribute is only available on z/OS.


Specifies that the open application is an RRS-coordinated application that uses a batch connection.

#### **cicsTransaction**

 This attribute is only available on z/OS.

Specifies that the open application is a CICS transaction.

#### **imsTransaction**

 This attribute is only available on z/OS.

Specifies that the open application is an IMS transaction.

#### **systemExtension**

Specifies that the open application is an application that performs an extension of function that is provided by the queue manager.

### **asynchronousConsumerState**

String.

Specifies the state of the asynchronous consumer on the queue.

The value is one of the following values:

#### **active**

Specifies that an MQCB call set up a function to call back to process messages asynchronously, and the connection handle has started so that asynchronous message consumption can proceed.

**inactive**

Specifies that an MQCB call set up a function to call back to process messages asynchronously, but the connection handle is not started, or is stopped or suspended.

**suspended**

Specifies that the asynchronous consumption callback is suspended so that asynchronous message consumption cannot proceed on the handle.

This situation can be either because an MQCB or MQCTL call with *Operation* MQOP\_SUSPEND was issued against this object handle by the application, or because it was suspended by the system. If it was suspended by the system, as part of the process of suspending asynchronous message consumption the callback function is called with the reason code that describes the problem that resulted in suspension. This situation is reported in the reason field in the MQCBC structure passed to the callback. In order for asynchronous message consumption to proceed, the application must issue an MQCB or MQCTL call with *Operation* MQOP\_RESUME.

**suspendedTemporarily**


Specifies that the asynchronous consumption callback is temporarily suspended by the system so that asynchronous message consumption cannot proceed on this handle.

As part of the process of suspending asynchronous message consumption the callback function is called with the reason code that describes the problem that resulted in suspension. This situation is reported in the reason field in the MQCBC structure passed to the callback. The callback function is called again when asynchronous message consumption is resumed by the system after the temporary condition is resolved.

**none**

Specifies that an MQCB call was not issued against this handle, so asynchronous message consumption is not configured on the handle.

**addressSpaceId**

 This attribute is only available on z/OS.

String.

Specifies a four character address space identifier for the application.

**channelName**

String.

Specifies the channel name.

**connectionName**

String.

Specifies the connection name.

**state**

String.

Specifies the state of the handle.

This value is one of the following values:

**active**

Specifies that an API call from a connection is in progress for the queue. This state can occur when an MQGET WAIT call is in progress.

**inactive**

Specifies that no API call from a connection is in progress for the queue. This state can occur when no MQGET WAIT call is in progress.

**openOptions**

JSON array.

Specifies the open options that are in force for the queue.

Any of the valid MQOO options can be present in the array. For more information about the MQOO\_\* options, see [MQOO\\_\\* \(Open Options\)](#).

### **processId**

MQ Appliance ALW

This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

Integer.

Specifies the process ID of the open application.

### **processSpecificationBlockName**

z/OS

This attribute is only available on z/OS.

String.

Specifies the eight character name of the program specification block that is associated with the running IMS transaction.

### **processSpecificationTableId**

z/OS

This attribute is only available on z/OS.

String.

Specifies the four character identifier of the program specification table region identifier for the connected IMS region.

### **qmgrTransactionId**

String.

Specifies the unit of recovery that is assigned by the queue manager.

This identifier is represented as 2 hexadecimal digits for each byte of the recovery identifier.

This attribute cannot be used to filter results.

### **cicsTaskNumber**

z/OS

This attribute is only available on z/OS.

Integer.

Specifies a seven digit CICS task number.

### **threadId**

MQ Appliance ALW

This attribute is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

Integer.

Specifies the thread ID of the open application.

A value of 0 indicates that the handle was opened by a shared connection. A handle that is created by a shared connection is logically open to all threads.

### **cicsTransactionId**

z/OS

This attribute is only available on z/OS.

String.

Specifies a four character CICS transaction ID.

### **unitOfWorkId**

String.

Specifies the recovery identifier for the unit of recovery. The format of this value is determined by the value of **unitOfWorkType**.

This identifier is represented as 2 hexadecimal digits for each byte of the recovery identifier.

This attribute cannot be used to filter results.

### **unitOfWorkType**


String.

Specifies the type of external unit of recovery identifier as perceived by the queue manager.


The value is one of the following values:

**qmgr**


**cics**

 This value is only available on z/OS.

**ims**

 This value is only available on z/OS.

**rrs**

 This value is only available on z/OS.

**xa**

**userId**


String.

Specifies the user identifier of the open application.

## **DELETE**

Use the HTTP DELETE method with the queue resource to delete a specified queue on a specified queue manager.

### **Note:**

- This resource URL is available only in version 1 of the REST API. To delete queues using version 2 of the REST API, use the [“/admin/action/qmgr/{qmgrName}/mqsc”](#) on page 2159 resource.
-  This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

This REST API command is similar to the [“MQCMD\\_DELETE\\_Q \(Delete Queue\)”](#) on page 1147 PCF command, and the [“DELETE queues”](#) on page 620 MQSC commands.

- [Resource URL](#)
- [Optional query parameters](#)
- [“Request headers”](#) on page 2366
- [Request body format](#)
- [“Security requirements”](#) on page 2367
- [Response status codes](#)
- [“Response headers”](#) on page 2367
- [Response body format](#)
- [Examples](#)

## **Resource URL**

`https://host:port/ibmmq/rest/v1/admin/qmgr/{qmgrName}/queue/{queueName}`

### **qmgrName**

Specifies the name of the queue manager on which the queue to delete exists.

You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

The queue manager name is case-sensitive.

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.

- A period (.) must be encoded as %2E.

### queueName

Specifies the name of the queue to delete.

The queue name is case-sensitive.


If the queue name includes a forward slash or a percent sign, these characters must be URL encoded:

- A forward slash, /, must be encoded as %2F.
- A percent sign, %, must be encoded as %25.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Optional query parameters

### keepAuthorityRecords

 This parameter is only available on the IBM MQ Appliance, AIX, Linux, and Windows.

Specifies that the associated authority records are not deleted.

### commandScope=scope

 This parameter is only available on z/OS.

Specifies how the command is run when the queue manager is a member of a queue sharing group.

You cannot specify this parameter if the queue manager is not a member of a queue sharing group.

scope can be one of the following values:

#### The name of a queue manager

Specifies that the command is run on the queue manager that is named. The queue manager must be active within the same queue sharing group as the queue manager that is specified in the resource URL.

You cannot specify the queue manager name that is the queue manager that is specified in the resource URL.

If the queue manager name includes a percent sign, %, this character must be URL encoded as %25.

\*

Specifies that the command is run on the local queue manager and also passed to every active queue manager in the queue sharing group.

If this option is used, an `ibm-mq-qmgrs` response header is returned with a comma-separated list of the queue managers that generated a response. For example, the header might look like the following header:

```
ibm-mq-qmgrs: MQ21, MQ22
```

### purge

Specifies that all messages are purged from the queue.

If messages are on the queue, you must specify **purge**, or the queue cannot be deleted.

### queueSharingGroupDisposition=disposition

 This parameter is only available on z/OS.

Specifies where the queue is defined and how it behaves. That is, it specifies the disposition of the queue.

disposition can be one of the following values:

### copy

Specifies that the queue definition exists on the page set of the queue manager that runs the command. The queue was defined by a command that used the **MQQSGD\_COPY** PCF parameter, or the **copy** REST API parameter.

Any queue in the shared repository, or any queue that is defined by using the **MQQSGD\_Q\_MGR** PCF parameter, or **qmgr** REST API parameter, is not affected by this command.

### group

Specifies that the queue definition exists in the shared repository. The queue was defined by a command that used the **MQQSGD\_GROUP** PCF parameter, or the **group** REST API parameter.

Any queue that exists on the page set of the queue manager that runs the command, except for a local copy of the queue, is not affected by this command.

If the deletion is successful, the following MQSC command is generated and sent to all active queue managers in the queue sharing group to delete local copies on page set zero:

```
DELETE queue(q-name) QSGDISP(COPY)
```

or for a local queue only:

```
DELETE QLOCAL(q-name) NOPURGE QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with QSGDISP(COPY) fails.

**Note:** You always get the NOPURGE option even if you specify the **purge** flag. To delete messages on local copies of the queues you must explicitly run, for each copy, a command to delete the queue with the **purge** flag, and a **queueSharingGroupDisposition** value of copy.

### qmgr

Specifies that the queue definition exists on the page set of the queue manager that runs the command. The object was defined by a command that used the **MQQSGD\_Q\_MGR** PCF parameter or the **qmgr** REST API parameter.

Any queue that exists in the shared repository, or any local copy of such a queue, is not affected by this command.

### shared

This value is only valid for local queues.

Specifies that the queue exists in the shared repository. The object was defined by a command that used the **MQQSGD\_SHARED** PCF parameter or the **shared** REST API parameter.

Any queue that exists on the page set of the queue manager that runs the command, or any queue that is defined by a command that uses the parameter **MQQSGD\_GROUP** is not affected by this command.

The default value is **qmgr**.

## Request headers

The following headers must be sent with the request:

### ibm-mq-rest-csrf-token

This header must be set, but the value can be anything, including being blank.

### Authorization

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### ibm-mq-rest-gateway-qmgr

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## Request body format

None.

## Security requirements

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

The security principal of the caller must be granted the ability to issue the following PCF commands for the specified queue manager:

- For the queue that is specified by the *{queueName}* portion of the resource URL, authority to issue the **MQCMD\_DELETE\_Q** PCF command must be granted.

**ALW** On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the **setmqaut** command. For more information, see [setmqaut \(grant or revoke authority\)](#).

**z/OS** On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 204

Queue deleted successfully.

### 400

Invalid data provided.

For example, invalid queue data is specified, or the queue is not empty.

### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. The `ibm-mq-rest-csrf-token` header must also be specified. For more information, see [“Security requirements” on page 2367](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources. For more information about the access that is required, see [“Security requirements” on page 2367](#).
- **V9.3.5** Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Queue does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

The following headers are returned with the response:

On z/OS, if the optional query parameter `commandScope=*` is used, this header is returned with a comma-separated list of the queue managers that generated a response. For example, the header might look like the following header:

```
ibm-mq-qmgrs: MQ21, MQ22
```

If an error occurs before the command is issued to the queue managers, the response header does not contain the list of queue managers. For example, a request that generates a 200 or 201 status code has the header because the command was successful. A request that generates a 401 (not authenticated) status code does not have the header because the request was rejected. A request that generates a 403 (not authorized) status code has the header because individual queue managers decide whether the command is authorized.

### ibm-mq-rest-gateway-qmgr

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

## Response body format

The response body is empty if the queue is deleted successfully. If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

## Examples

The following example deletes the queue Q1 from the queue manager QM1, and purges all messages from the queue when used with the HTTP DELETE method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue/Q1?purge
```

## /admin/qmgr/{qmgrName}/subscription

You can use the HTTP GET method with the subscription resource to request information about subscriptions.

### Note:

- This resource URL is available only in version 1 of the REST API. To query subscriptions using version 2 of the REST API, use the ["/admin/action/qmgr/{qmgrName}/mqsc"](#) on page 2159 resource.
- **V 9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).

You can use the administrative REST API gateway with this resource URL.

For more information about the PCF equivalents to the subscription REST API parameters and attributes, see ["REST API and PCF equivalents for subscriptions"](#) on page 2387.

## GET

Use the HTTP GET method with the subscription resource to request information about subscriptions.

### Note:

- This resource URL is available only in version 1 of the REST API. To query subscriptions using version 2 of the REST API, use the ["/admin/action/qmgr/{qmgrName}/mqsc"](#) on page 2159 resource.
- **V 9.3.5** This resource is not available in a stand-alone IBM MQ Web Server installation. For more information about the installation options for the IBM MQ component that runs the administrative REST API, see [The IBM MQ Console and REST API](#).



The information that is returned is similar to the information returned by the “MQCMD\_INQUIRE\_SUBSCRIPTION (Inquire Subscription)” on page 1413 PCF command, and the “DISPLAY SUB (display subscription information)” on page 835 MQSC command.

- [“Resource URL” on page 2369](#)
- [“Optional query parameters” on page 2369](#)
- [“Request headers” on page 2371](#)
- [“Request body format” on page 2371](#)
- [“Security requirements” on page 2372](#)
- [“Response status codes” on page 2372](#)
- [“Response headers” on page 2373](#)
- [“Response body format” on page 2373](#)
- [“Examples” on page 2374](#)

## Resource URL

`https://host:port/ibmmq/rest/v1/admin/qmgr/{qmgrName}/subscription/{subscriptionName}`

### qmgrName

Specifies the name of the queue manager on which to query the subscriptions.

You can specify a remote queue manager as the **qmgrName**. If you specify a remote queue manager, you must configure a gateway queue manager. For more information, see [Remote administration using the REST API](#).

The queue manager name is case-sensitive.

If the queue manager name includes a forward slash, a period, or a percent sign, these characters must be URL encoded:

- A forward slash (/) must be encoded as %2F.
- A percent sign (%) must be encoded as %25.
- A period (.) must be encoded as %2E.

### subscriptionName

Optionally specifies the name of a subscription that exists on the specified queue manager.

The subscription name is case-sensitive.

If the subscription name includes any non-alphanumeric characters, they must be URL encoded.

You can use HTTP instead of HTTPS if you enable HTTP connections. For more information about enabling HTTP, see [Configuring HTTP and HTTPS ports](#).

## Optional query parameters

**attributes={object,...[\*|object.attributeName,...]}**

### object,...

Specifies a comma-separated list of JSON objects that contain related subscription attributes to return.

For example, to return all subscription attributes that are related to time stamps, specify `timestamps`. To return all subscription attributes that are related to the destination and user, specify `destination,user`.

You cannot specify the same object more than once.

For a full list of objects and associated attributes, see [Attributes for subscriptions](#).

### \*

Specifies all attributes.

**object.attributeName,...**

Specifies a comma-separated list of queue configuration attributes to return.

Each attribute must specify the JSON object that contains the attribute, in the form `object.attributeName`. For example, to return the `correlationId` attribute, which is contained in the destination object, specify `destination.correlationId`.

You cannot specify the same attribute more than once.

For a full list of attributes and associated objects, see [Attributes for subscriptions](#).

**filter=filterValue**

Specifies a filter for the subscription definitions that are returned.

This query parameter cannot be used if you specify a subscription name in the resource URL or if you use the ID query parameter.

You can specify only one filter.

*filterValue* has the following format:

```
attribute:operator:value
```

where:

**attribute**

Specifies one of the applicable attributes. For a full list of attributes, see [Attributes for subscriptions](#). The following attributes cannot be specified:

- name
- id

To filter on any attributes that are time stamps, the filter can specify any portion of the time stamp, with a trailing asterisk, \*. The format of a time stamp is, YYYY-MM-DDThh:mm:ss. For example, you can specify 2001-11-1\* to filter on dates in the range 2001-11-10 to 2001-11-19, or 2001-11-12T14:\* to filter any minute in the specified hour of the specified day.

Valid values for the YYYY section of the date are in the range 1900 - 9999.

The time stamp is a string. Therefore, only the `equalTo` and `notEqualTo` operators can be used with the time stamp.

**operator**

Specifies one of the following operators:

**lessThan**

Use this operator only with integer attributes.

**greaterThan**

Use this operator only with integer attributes.

**equalTo**

Use this operator with any attribute.

**notEqualTo**

Use this operator with any attribute.

**lessThanOrEqualTo**

Use this operator only with integer attributes.

**greaterThanOrEqualTo**

Use this operator only with integer attributes.

**value**

Specifies the constant value to test against the attribute.

The value type is determined by the attribute type.

For string and boolean attributes, you can omit the value field after the colon. For string attributes, omit the value to return subscriptions with no value for the specified attribute. For boolean attributes, omit the value to return any subscriptions that have the specified attribute set to false.

For example, the following filter returns all subscriptions where the topic name attribute is not specified:

```
filter=topic.name:equalTo:
```

A single asterisk, \*, can be used for string attributes specified at the end of the value as a wildcard. If the value includes non-alphanumeric characters, then they must be URL encoded. If the value contains a percent character or any asterisk that is not intended to be a wildcard, then the value must be URL encoded a second time. That is, a percent character must be encoded as %2525. An asterisk must be encoded as %252A.

#### **id=id**

Specifies the ID of a subscription that exists on the queue manager specified.

This query parameter cannot be used if you specify a subscription name in the resource URL or the name query parameter.

The ID is a string that contains a hexadecimal number. It can be comprised of a mixture of uppercase and lowercase characters.

#### **name=name**

Specifies a wildcard subscription name to filter on.

This query parameter cannot be used if you specify a subscription name in the resource URL or the id query parameter.

The *name* specified must either be blank or include an asterisk, \*, as a wildcard. You can specify one of the following combinations:

Specifies that subscriptions that do have a blank name attribute are returned.

**\***

Specifies that all subscriptions are returned.

#### **prefix\***

Specifies that all subscriptions with the specified prefix in the subscription name are returned.

#### **\*suffix**

Specifies that all subscriptions with the specified suffix in the subscription name are returned.

#### **prefix\*suffix**

Specifies that all subscriptions with the specified prefix and the specified suffix in the subscription name are returned.

## **Request headers**

The following headers must be sent with the request:

### **Authorization**

This header must be sent if you are using basic authentication. For more information, see [Using HTTP basic authentication with the REST API](#).

The following headers can optionally be sent with the request:

### **ibm-mq-rest-gateway-qmgr**

This header specifies the queue manager that is to be used as the gateway queue manager. The gateway queue manager is used to connect to a remote queue manager. For more information, see [Remote administration using the REST API](#).

## **Request body format**

None.

## Security requirements


The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information about security for the administrative REST API, see [IBM MQ Console and REST API security](#).

If token based security is used, the LTPA token that is used to authenticate the user must be provided with the request as a cookie. For more information about token-based authentication, see [Using token-based authentication with the REST API](#).

The security principal of the caller must be granted the ability to issue the following PCF commands for the specified queue manager:

- For the subscription that is specified by the *{subscriptionName}* portion of the resource URL, the *id* query parameter, or for subscriptions that match the specified query parameters, authority to issue the **MQCMD\_INQUIRE\_SUBSCRIPTION** PCF command must be granted.

A principal has display authority if the principal can issue the **MQCMD\_INQUIRE\_SUBSCRIPTION** PCF command. If the principal has display authority for only some of the subscriptions that are specified by the resource URL and query parameters, then the array of subscriptions that is returned from the REST request is limited to those subscriptions that the principal has authority to display. No information is returned about subscriptions that cannot be displayed. If the principal does not have display authority for any of the subscriptions that are specified by the resource URL and query parameters, an HTTP status code of 403 is returned.

 On AIX, Linux, and Windows, you can grant authority to security principals to use IBM MQ resources by using the **setmqaut** command. For more information, see [setmqaut \(grant or revoke authority\)](#).

 On z/OS, see [Setting up security on z/OS](#).

## Response status codes

### 200

Subscriptions retrieved successfully.

### 400

Invalid data provided.

For example, invalid subscription attributes specified.


### 401

Not authenticated.

The caller must be authenticated to the mqweb server and must be a member of one or more of the MQWebAdmin, MQWebAdminRO, or MQWebUser roles. For more information, see [“Security requirements” on page 2372](#).

### 403

Access prohibited for one of the following reasons:

- Not authorized. The caller is authenticated to the mqweb server and is associated with a valid principal. However, the principal does not have access to the required IBM MQ resources. For more information about the access that is required, see [“Security requirements” on page 2372](#).
-  Access prohibited in the current server environment. The administrative REST API is not available in a stand-alone IBM MQ Web Server installation.

### 404

Subscription does not exist.

### 500

Server issue or error code from IBM MQ.

### 503

Queue manager not running.

## Response headers

The following headers are returned with the response:

### Content-Type

This header is returned with a value of `application/json; charset=utf-8`.

### ibm-mq-rest-gateway-qmgr

This header is returned if a remote queue manager is specified in the resource URL. The value of this header is the name of the queue manager that is used as the gateway queue manager.

## Response body format

The response is in JSON format in UTF-8 encoding. The response contains an outer JSON object that contains a single JSON array called `subscription`. Each element in the array is a JSON object that represents information about a subscription. Each of these JSON objects contains the following attributes:

### id

Hexadecimal string

Specifies the unique key that identifies the subscription.

This attribute is always returned.

### name

String

Specifies the name of the subscription.

This attribute is always returned.

### resolvedTopicString

String

Specifies the fully resolved topic string using the combined values from the topic name and defined string when the subscription was created.

This attribute is always returned.

The following objects can be included in the JSON object that represents information about a subscription. Which objects and attributes are returned depends on the URL that was specified for the request:

### topic

Contains attributes that are related to a defined topic.

### selector

Contains attributes that are related to the message selector.

### destination

Contains attributes that are related to the destination queue / queue manager.

### user

Contains attributes that are related to user, such as the accounting token, the user ID that owns the subscription and the user data.

### general

Contains attributes that are related to general subscription properties, such whether the subscription is durable, how the subscription was created and whether wildcards should be interpreted in the topic string.

### extended

Contains attributes that are related to extended subscription properties, such as the expiry time, the message priority, and the network scope.

### timestamps

Contains attributes that are related to date and time information, such as the time stamp of when the subscription was created.

For more information, see [“Response body attributes for subscriptions” on page 2375](#).

If an error occurs, the response body contains an error message. For more information, see [REST API error handling](#).

## Examples

- The following example lists all subscriptions on the queue manager QM1. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/subscription
```

The following JSON response is returned:

```
{
  "subscription":
  [
    {
      "id": "414D5120514D332020202020202020A878195911AFD206",
      "name": "SYSTEM.DEFAULT.SUB",
      "resolvedTopicString": ""
    },
    {
      "id": "414D5120514D332020202020202020C0740592162214A",
      "name": "MySubscription",
      "resolvedTopicString": "sports/golf"
    },
    {
      "id": "414D5120514D332020202020202020C07405921621307",
      "name": "QM1 SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS
414D51590101000000000000000000000000000000000000 SYSTEM.BROKER.ADMIN.STREAM MQ/QM1 /
StreamSupport",
      "resolvedTopicString": "SYSTEM.BROKER.ADMIN.STREAM/MQ/QM1 /StreamSupport"
    }
  ]
}
```

- The following example lists all subscriptions on the queue manager QM1, showing their topic properties. The following URL is used with the HTTP GET method:

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/subscription?attributes=topic
```

The following JSON response is returned:

```
{
  "subscription":
  [
    {
      "id": "414D5120514D332020202020202020A878195911AFD206",
      "name": "SYSTEM.DEFAULT.SUB",
      "resolvedTopicString": "",
      "topic": {
        "definedString": "",
        "name": ""
      }
    },
    {
      "id": "414D5120514D332020202020202020C0740592162214A",
      "name": "MySubscription",
      "resolvedTopicString": "sports/snooker",
      "topic": {
        "definedString": "sports/snooker",
        "name": ""
      }
    },
    {
      "id": "414D5120514D332020202020202020C07405921621307",
      "name": "QM1 SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS
414D51590101000000000000000000000000000000000000 SYSTEM.BROKER.ADMIN.STREAM MQ/QM1 /
StreamSupport",
      "resolvedTopicString": "SYSTEM.BROKER.ADMIN.STREAM/MQ/QM1 /StreamSupport",
      "topic": {
        "definedString": "MQ/QM1 /StreamSupport",
        "name": "SYSTEM.BROKER.ADMIN.STREAM"
      }
    }
  ]
}
```

### *Response body attributes for subscriptions*

When you use the HTTP GET method with the `subscription` object to request information about subscriptions, the following attributes are returned within named JSON objects.

The following objects are available:

- [“topic” on page 2375](#)
- [“selector” on page 2375](#)
- [“destination” on page 2376](#)
- [“user” on page 2376](#)
- [“general” on page 2376](#)
- [“extended” on page 2377](#)
- [“timestamps” on page 2378](#)

For more information about the PCF equivalents to the subscription REST API parameters and attributes, see [“REST API and PCF equivalents for subscriptions” on page 2387](#).

## **topic**

The `topic` object contains attributes that are related to a defined topic.

### **name**

String.

Specifies the name of a previously defined topic object from which the topic string prefix is obtained for the subscription.

### **definedString**

String.

Specifies the topic string that contains the application part of the topic string only.

## **selector**

The `selector` object contains attributes that are related to the message selector.

### **value**

String.

Specifies the selector applied to messages published to the topic.

Only those messages that satisfy the selection criteria are put to the destination specified by this subscription.

### **type**

String.

Specifies type of selector.

The value is one of the following values:

#### **none**

Specifies that no selector is present.

#### **standard**

Specifies that the selector references only the properties of the message, not its content, using the standard IBM MQ selector syntax. Selectors of this type are to be handled internally by the queue manager.

#### **extended**

Specifies that the selector uses extended selector syntax, typically referencing the content of the message. Selectors of this type cannot be handled internally by the queue manager; extended selectors can be handled only by another program such as IBM Integration Bus.

## destination

The `destination` object contains attributes that are related to the destination queue / queue manager.

### **isManaged**

Boolean.

Specifies whether the destination is managed.

### **qmgrName**

String.

Specifies the name of the destination queue manager, either local or remote, to which messages for the subscription are forwarded.

### **name**

String.

Specifies the name of the alias, local, remote, or cluster queue to which messages for this subscription are put.

### **correlationId**

Hexadecimal.

Specifies the correlation identifier that is placed in the `CorrelId` field of the message descriptor for all the messages sent to this subscription.

## user

The `user` object contains attributes that are related to user that created the subscription, such as the accounting token, the user ID that owns the subscription and the user data.

### **accountingToken**

Hexadecimal.

Specifies the accounting token used in the `AccountingToken` field of the message descriptor.

### **applicationIdentityData**

String.

Specifies the application identity data used in the `ApplIdentityData` field of the message descriptor.

### **data**

String.

Specifies the user data associated with the subscription.

### **name**

String.

Specifies the `userid` that 'owns' this subscription. This parameter is either the `userid` associated with the creator of the subscription, or, if subscription takeover is permitted, the `userid` which last took over the subscription.

### **isVariable**

Boolean.

Specifies whether any user other than the one who created the subscription can take over ownership.

## general

The `general` object contains attributes that are related to general subscription properties, such as whether the subscription is durable, how the subscription was created and whether wildcards should be interpreted in the topic string.

### **isDurable**

Boolean.

Specifies whether this subscription is a durable subscription.



If the subscription is durable, the subscription persists, even if the creating application disconnects from the queue manager or issues an MQCLOSE call for the subscription. The queue manager reinstates the subscription during restart.

If the subscription is non-durable, the queue manager removes the subscription when the creating application disconnects from the queue manager or issues an MQCLOSE call for the subscription. If the subscription has a **destination.class** of managed, the queue manager removes any messages not yet consumed when it closes the subscription.

### **type**

String.

Specifies how the subscription was created.

The value is one of the following values:

#### **administrative**

Created using DEF SUB MQSC, REST or PCF command. It also indicates that a subscription has been modified using an administrative command.

#### **api**

Created using an MQSUB API request.

#### **proxy**

Created internally and used for routing publications through a queue manager.

### **usesCharacterWildcard**

Boolean.

Specifies the schema to be used when any wildcard characters that are contained in the topic string are interpreted.

If the value is set to `true`, wildcard characters represent portions of strings; this is for compatibility with IBM MQ V6.0 brokers.

If the value is set to `false`, wildcard characters represent portions of the topic hierarchy; this value is for compatibility with IBM Integration Bus brokers.

### **extended**

The extended object contains attributes that are related to extended subscription properties, such as the expiry time, the message priority and the network scope.

#### **expiry**

Integer.

Specifies the time, in tenths of seconds, at which a subscription expires after its creation date.

A value of -1 can be used to represent unlimited.

#### **level**

Integer.

Specifies the level within the subscription interception hierarchy at which this subscription is made.

#### **messagePriority**

String.

Specifies the priority of messages sent to this subscription. It has the range 0-9.

Additionally, the value can be one of the following values:

##### **asPublished**

The priority of messages sent to this subscription is taken from that priority supplied to the published message.

##### **asQueue**

The priority of messages sent to this subscription is determined by the default priority of the queue defined as a destination.

#### **messagePropertyControl**

String.

Specifies how publish/subscribe related message properties are added to messages sent to this subscription.

The value is one of the following values:

**none**

Specifies that publish/subscribe properties are not added to the messages.

**compatible**

Specifies that if the original publication is a PCF message, then the publish/subscribe properties are added as PCF attributes. Otherwise, publish/subscribe properties are added within an MQRFH version 1 header. This method is compatible with applications coded for use with previous versions of IBM MQ.

**pcf**

Specifies that publish/subscribe properties are added as PCF attributes.

**rfh2**

Specifies that publish/subscribe properties are added within an MQRFH version 2 header. This method is compatible with applications coded for use with IBM Integration Bus brokers.

**deliverOnRequest**

Boolean.

Specifies whether the subscriber polls for updates using the MQSUBRQ API call, or whether all publications are delivered to this subscription.

If the value is set to `true`, publications are only delivered to this subscription in response to an MQSUBRQ API call.

If the value is set to `false`, all publications on the topic are delivered to this subscription.

**networkScope**

String.

Specifies whether this subscription is passed to other queue managers in the network.

The value is one of the following values:

**all**

Specifies that the subscription is forwarded to all queue managers directly connected through a publish/subscribe collective or hierarchy.

**qmgr**

Specifies that the subscription forwards only messages that are published on the topic within this queue manager.

**timestamps**

The `timestamps` object contains attributes that are related to date and time information.

**altered**

String.

Specifies the date and time at which the subscription was last altered.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

**created**

String.

Specifies the date and time at which the subscription was created.

For more information about the time stamp format that is used to return the date and time, see [REST API time stamps](#).

## REST API and PCF equivalents

For most REST API optional query parameters and attributes, an equivalent PCF parameter or attribute exists. Use these topics to understand these equivalents.

### REST API and PCF equivalents for queue managers

For most REST API optional query parameters and attributes for queue managers, an equivalent PCF parameter or attribute exists. Use the tables that are provided to understand these equivalents.

- [“Queue manager attribute equivalents” on page 2379](#)
- [“Unsupported PCF attributes” on page 2379](#)

### Queue manager attribute equivalents

<i>Table 351. Queue manager attributes for the REST API and equivalent PCF attributes.</i>			
REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
name	<b>MQCA_Q_MGR_NAME</b>		
state	<b>MQIACF_Q_MGR_STATU S</b>		
status.started	<b>MQCACF_Q_MGR_START _DATE</b> <b>MQCACF_Q_MGR_START _TIME</b>		
status.channelInit iatorState	<b>MQIACF_CHINIT_STAT US</b>	MQSVC_STATUS_STOP PED MQSVC_STATUS_START ING MQSVC_STATUS_RUNNI NG MQSVC_STATUS_STOP PING	stopped starting running stopping
status.ldapConnect ionState	<b>MQIACF_LDAP_CONNEC TION_STATUS</b>	MQLDAPC_CONNECTED MQLDAPC_ERROR MQLDAPC_INACTIVE	connected error disconnected
status.connectionC ount	<b>MQIACF_CONNECTION_ COUNT</b>		

### Unsupported PCF attributes

The following queue manager PCF attributes are not supported by the administrative REST API qmgr resource:

- **MQCA\_INSTALLATION\_DESC**
- **MQCA\_INSTALLATION\_NAME**
- **MQCA\_INSTALLATION\_PATH**
- **MQCACF\_CURRENT\_LOG\_EXTENT\_NAME**
- **MQCACF\_LOG\_PATH**
- **MQCACF\_MEDIA\_LOG\_EXTENT\_NAME**

- MQCACF\_RESTART\_LOG\_EXTENT\_NAME

## REST API and PCF equivalents for queues

For most REST API optional query parameters and attributes for queues, an equivalent PCF parameter or attribute exists. Use the tables that are provided to understand these equivalents.

- [“Optional query parameter equivalents” on page 2380](#)
- [“Queue attribute equivalents” on page 2381](#)
- [“Unsupported PCF attributes” on page 2387](#)

## Optional query parameter equivalents

<i>Table 352. Queue optional query parameters for the REST API and equivalent PCF parameters.</i>			
<b>REST API optional query parameter</b>	<b>PCF parameter</b>	<b>Related values (REST API)</b>	<b>Related values (PCF)</b>
commandScope= <i>scope</i>	<b>MQCACF_COMMAND_SCOPE</b>	None.	None.
filter= <i>filterValue</i>	<b>MQCFT_INTEGER_FILTER</b> <b>MQCFT_STRING_FILTER</b>	lessThan greaterThan lessThanOrEqualTo greaterThanOrEqualTo equalTo notEqualTo	MQCFOP_LESS MQCFOP_GREATER MQCFOP_NOT_GREATER MQCFOP_NOT_LESS MQCFOP_EQUAL MQCFOP_LIKE MQCFOP_NOT_EQUAL MQCFOP_NOT_LIKE
force	<b>MQIACF_FORCE</b>		
keepAuthorityRecords	<b>MQIACF_REMOVE_AUTHREC</b>		
like= <i>queueName</i>	<b>MQCACF_FROM_Q_NAME</b>		
noReplace	<b>MQIACF_REPLACE</b>		
purge	<b>MQIACF_PURGE</b>		
queueSharingGroupDisposition= <i>disposition</i>	<b>MQIA_QSG_DISP</b>	live all copy  group private qmgr shared	MQQSGD_LIVE MQQSGD_ALL MQQSGD_COPY MQQSGD_GROUP MQQSGD_PRIVATE MQQSGD_Q_MGR MQQSGD_SHARED
type= <i>type</i>	<b>MQIA_Q_TYPE</b>	all local alias remote cluster model	None. MQQT_LOCAL MQQT_ALIAS MQQT_REMOTE MQQT_CLUSTER MQQT_MODEL

## Queue attribute equivalents

<i>Table 353. Queue attributes for the REST API and equivalent PCF attributes.</i>			
<b>REST API attribute</b>	<b>PCF attribute</b>	<b>Related values (REST API)</b>	<b>Related values (PCF)</b>
name	<b>MQCA_Q_NAME</b>		
type	<b>MQIA_Q_TYPE</b>	local alias remote cluster model	MQQT_LOCAL MQQT_ALIAS MQQT_REMOTE MQQT_CLUSTER MQQT_MODEL
remote.qmgrName	<b>MQCA_REMOTE_Q_MGR_NAME</b>		
remote.queueName	<b>MQCA_REMOTE_Q_NAME</b>		
remote.transmissionQueueName	<b>MQCA_XMIT_Q_NAME</b>		
alias.targetName	<b>MQCA_BASE_OBJECT_NAME</b>		
alias.targetType	<b>MQIA_BASE_TYPE</b>	queue topic	MQOT_Q MQOT_TOPIC
dynamic.type	<b>MQIA_DEFINITION_TYPE</b>	permanentDynamic sharedDynamic temporaryDynamic	MQQDT_PERMANENT_DYNAMIC MQQDT_SHARED_DYNAMIC MQQDT_TEMPORARY_DYNAMIC
model.type	<b>MQIA_DEFINITION_TYPE</b>	permanentDynamic sharedDynamic temporaryDynamic	MQQDT_PERMANENT_DYNAMIC MQQDT_SHARED_DYNAMIC MQQDT_TEMPORARY_DYNAMIC
cluster.name	<b>MQCA_CLUSTER_NAME</b>		
cluster.namelist	<b>MQCA_CLUSTER_NAMELIST</b>		
cluster.qmgrId	<b>QMgrIdentifier</b>		
cluster.qmgrName	<b>QMgrName</b>		
cluster.queueType	<b>ClusterQType</b>	local alias remote qmgrAlias	MQCQT_LOCAL_Q MQCQT_ALIAS_Q MQCQT_REMOTE_Q MQCQT_Q_MGR_ALIAS
cluster.transmissionQueueForChannelName	<b>ClusterChannelName</b>		

Table 353. Queue attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
cluster.workloadPriority	MQIA_CLWL_Q_PRIORITY		
cluster.workloadQueueUse	MQIA_CLWL_USEQ	true false	MQTC_ON MQTC_OFF
cluster.workloadRank	MQIA_CLWL_Q_RANK		
trigger.enabled	MQIA_TRIGGER_CONTROL	true false	MQTC_ON MQTC_OFF
trigger.data	MQCA_TRIGGER_DATA		
trigger.depth	MQIA_TRIGGER_DEPTH		
trigger.initiationQueueName	MQCA_INITIATION_Q_NAME		
trigger.messagePriority	MQIA_TRIGGER_MSG_PRIORITY		
trigger.processName	MQCA_PROCESS_NAME		
trigger.type	MQIA_TRIGGER_TYPE	none every first depth	MQTT_NONE MQTT_EVERY MQTT_FIRST MQTT_DEPTH
events.depth.highEnabled	MQIA_Q_DEPTH_HIGH_EVENT	true false	MQEVR_ENABLED MQEVR_DISABLED
events.depth.highPercentage	MQIA_Q_DEPTH_HIGH_LIMIT		
events.depth.lowEnabled	MQIA_Q_DEPTH_LOW_EVENT	true false	MQEVR_ENABLED MQEVR_DISABLED
events.depth.lowPercentage	MQIA_Q_DEPTH_LOW_LIMIT		
events.depth.fullEnabled	MQIA_Q_DEPTH_MAX_EVENT	true false	MQEVR_ENABLED MQEVR_DISABLED
events.serviceInterval.highEnabled	MQIA_Q_SERVICE_INTERVAL_EVENT	true false	MQSIE_HIGH MQSIE_NONE (Equivalent only when okEnabled is also false)

Table 353. Queue attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
events.serviceInterval.okEnabled	<b>MQIA_Q_SERVICE_INTERVAL_EVENT</b>	true false	MQSIE_OK MQSIE_NONE (Equivalent only when highEnabled is also false)
events.serviceInterval.duration	<b>MQIA_Q_SERVICE_INTERVAL</b>		
applicationDefaults.clusterBind	<b>MQIA_DEF_BIND</b>	onOpen notFixed onGroup	MQBND_BIND_ON_OPEN MQBND_BIND_NOT_FIXED MQBND_BIND_ON_GROUP
applicationDefaults.messagePropertyControl	<b>MQIA_PROPERTY_CONTROL</b>	all compatible force none version6Compatible	MQPROP_ALL MQPROP_COMPATIBILITY MQPROP_FORCE_MQRFH2 MQPROP_NONE MQPROP_V6COMPAT
applicationDefaults.messagePersistence	<b>MQIA_DEF_PERSISTENCE</b>	persistent nonPersistent	MQPER_PERSISTENT MQPER_NOT_PERSISTENT
applicationDefaults.messagePriority	<b>MQIA_DEF_PRIORITY</b>		
applicationDefaults.putResponse	<b>MQIA_DEF_PUT_RESPONSE_TYPE</b>	synchronous asynchronous	MQPRT_SYNC_RESPONSE MQPRT_ASYNC_RESPONSE
applicationDefaults.readAhead	<b>MQIA_DEF_READ_AHEAD</b>	no yes disabled	MQREADA_NO MQREADA_YES MQREADA_DISABLED
applicationDefaults.sharedInput	<b>MQIA_DEF_INPUT_OPTION</b>	true false	MQOO_INPUT_SHARED MQOO_INPUT_EXCLUSIVE
queueSharingGroup.disposition	<b>MQIA_QSG_DISP</b>	copy group qmgr shared	MQQSGD_COPY MQQSGD_GROUP MQQSGD_Q_MGR MQQSGD_SHARED
queueSharingGroup.qmgrName	No equivalent.		

Table 353. Queue attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
queueSharingGroup.structureName	<b>MQCA_CF_STRUC_NAME</b>		
dataCollection.accounting	<b>MQIA_ACCOUNTING_Q</b>	asQmgr off on	MQMON_Q_MGR MQMON_OFF MQMON_ON
dataCollection.monitoring	<b>MQIA_MONITORING_Q</b>	off asQmgr low medium high	MQMON_OFF MQMON_Q_MGR MQMON_LOW MQMON_MEDIUM MQMON_HIGH
dataCollection.statistics	<b>MQIA_STATISTICS_Q</b>	asQmgr off on	MQMON_Q_MGR MQMON_OFF MQMON_ON
storage.indexType	<b>MQIA_INDEX_TYPE</b>	none correlationId groupId messageId messageToken	MQIT_NONE MQIT_CORREL_ID MQIT_GROUP_ID MQIT_MSG_ID MQIT_MSG_TOKEN
storage.maximumMessageLength	<b>MQIA_MAX_MSG_LENGTH</b>		
storage.maximumDepth	<b>MQIA_MAX_Q_DEPTH</b>		
storage.messageDeliverySequence	<b>MQIA_MSG_DELIVERY_SEQUENCE</b>	priority fifo	MQMDS_PRIORITY MQMDS_FIFO
storage.nonPersistentMessageClass	<b>MQIA_NPM_CLASS</b>	normal high	MQNPM_CLASS_NORMAL MQNPM_CLASS_HIGH
storage.pageSet	<b>PageSetID</b>		
storage.storageClass	<b>MQCA_STORAGE_CLASS</b>		
general.description	<b>MQCA_Q_DESC</b>		
general.inhibitGet	<b>MQIA_INHIBIT_GET</b>	true false	MQQA_GET_INHIBITED MQQA_GET_ALLOWED
general.inhibitPut	<b>MQIA_INHIBIT_PUT</b>	true false	MQQA_PUT_INHIBITED MQQA_PUT_ALLOWED
general.isTransmissionQueue	<b>MQIA_USAGE</b>	true false	MQUS_TRANSMISSION MQUS_NORMAL



Table 353. Queue attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
extended.allowSharedInput	<b>MQIA_SHAREABILITY</b>	true false	MQQA_SHAREABLE MQQA_NOT_SHAREABLE
extended.backoutRequestQueueName	<b>MQCA_BACKOUT_REQ_Q_NAME</b>		
extended.backoutThreshold	<b>MQIA_BACKOUT_THRES HOLD</b>		
extended.custom	<b>MQCA_CUSTOM</b>		
extended.supportDistributionLists	<b>MQIA_DIST_LISTS</b>	true false	MQDL_SUPPORTED MQDL_NOT_SUPPORTED
extended.hardenGetBackout	<b>MQIA_HARDEN_GET_BACKOUT</b>	true false	MQQA_BACKOUT_HARDENED MQQA_BACKOUT_NOT_HARDENED
extended.enableMediaImageOperations	<b>ImageRecoverQueue</b>	yes no asQmgr	MQIMGRCOV_YES MQIMGRCOV_NO MQIMGRCOV_AS_QMGR
timestamps.altered	<b>MQCA_ALTERATION_DATE</b> <b>MQCA_ALTERATION_TIME</b>		
timestamps.clustered	<b>MQCA_CLUSTER_DATE</b> <b>MQCA_CLUSTER_TIME</b>		
timestamps.created	<b>MQCA_CREATION_DATE</b> <b>MQCA_CREATION_TIME</b>		
status.currentDepth	<b>MQIA_CURRENT_Q_DEPTH</b>		
status.lastGet	<b>MQCACF_LAST_GET_DATE</b> <b>MQCACF_LAST_GET_TIME</b>		
status.lastPut	<b>MQCACF_LAST_PUT_DATE</b> <b>MQCACF_LAST_PUT_TIME</b>		
status.mediaRecoveryLogExtent	<b>MQCACF_MEDIA_LOG_EXTENT_NAME</b>		

Table 353. Queue attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
status.oldestMessageAge	MQIACF_OLDEST_MESSAGE		
status.onQueueTime.longSamplePeriod	MQIACF_Q_TIME_INDICATOR		
status.onQueueTime.shortSamplePeriod	MQIACF_Q_TIME_INDICATOR		
status.openInputCount	MQIA_OPEN_INPUT_COUNT		
status.openOutputCount	MQIA_OPEN_OUTPUT_COUNT		
status.monitoringRate	MQIA_MONITORING_Q	off low medium high	MQMON_OFF MQMON_LOW MQMON_MEDIUM MQMON_HIGH
status.tPipeName	MQCA_TPIPE_NAME		
status.uncommittedMessages	MQIACF_UNCOMMITTED_MSGS		
applicationHandle.description	MQCACF_APPL_DESC		
applicationHandle.tag	MQCACF_APPL_TAG		
applicationHandle.type	MQIA_APPL_TYPE	queueManagerProcess channelInitiator userApplication batchConnection rrsBatchConnection cicsTransaction imsTransaction SystemExtension	MQAT_QMGR MQAT_CHANNEL_INITIATOR MQAT_USER MQAT_BATCH MQAT_RRS_BATCH MQAT_CICS MQAT_IMS MQAT_SYSTEM_EXTENSION
applicationHandle.asynchronousConsumerState	MQIACF_ASYNC_STATE	active inactive suspended suspendedTemporarily none	MQAS_ACTIVE MQAS_INACTIVE MQAS_SUSPENDED MQAS_SUSPENDED_TEMPORARY MQAS_NONE
applicationHandle.addressSpaceId	MQCACF_ASID		
applicationHandle.channelName	MQCACH_CHANNEL_NAME		

Table 353. Queue attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
applicationHandle.connectionName	<b>MQCACH_CONNECTION_NAME</b>		
applicationHandle.state	<b>MQIACF_HANDLE_STATE</b>	active inactive	MQHSTATE_ACTIVE MQHSTATE_INACTIVE
applicationHandle.openOptions	<b>MQIACF_OPEN_OPTIONS</b>		
applicationHandle.processId	<b>MQIACF_PROCESS_ID</b>		
applicationHandle.processSpecificationBlockName	<b>MQCACF_PSB_NAME</b>		
applicationHandle.processSpecificationTableId	<b>MQCACF_PST_ID</b>		
applicationHandle.qmgrTransactionId	<b>MQBACF_Q_MGR_UOW_ID</b>		
applicationHandle.cicsTaskNumber	<b>MQCACF_TASK_NUMBER</b>		
applicationHandle.threadId	<b>MQIACF_THREAD_ID</b>		
applicationHandle.cicsTransactionId	<b>MQCACF_TRANSACTION_ID</b>		
applicationHandle.unitOfWorkId	<b>MQBACF_EXTERNAL_UOW_ID</b>		
applicationHandle.unitOfWorkType	<b>MQIACF_UOW_TYPE</b>	qmgr cics ims rrs xa	MQUOWT_Q_MGR MQUOWT_CICS MQUOWT_IMS MQUOWT_RRS MQUOWT_XA
applicationHandle.UserId	<b>MQCACF_USER_IDENTIFIER</b>		

### Unsupported PCF attributes

The following queue PCF attributes are not supported by the administrative REST API:

- **MQIA\_SCOPE**
- **MQIA\_RETENTION\_INTERVAL**

### REST API and PCF equivalents for subscriptions

For most REST API optional query parameters and attributes for subscriptions, an equivalent PCF parameter or attribute exists. Use the tables that are provided to understand these equivalents.

- [“Optional query parameter equivalents” on page 2388](#)

- [“Subscription attribute equivalents” on page 2388](#)
- [“Unsupported PCF parameters” on page 2390](#)

## Optional query parameter equivalents

<i>Table 354. Subscription optional query parameters for the REST API and equivalent PCF parameters.</i>			
REST API optional query parameter	PCF parameter	Related values (REST API)	Related values (PCF)
<code>filter=filterValue</code>	<b>MQCFT_INTEGER_FILTER</b> <b>MQCFT_STRING_FILTER</b>	lessThan greaterThan lessThanOrEqualTo greaterThanOrEqualTo equalTo  notEqualTo	MQCFOP_LESS MQCFOP_GREATER MQCFOP_NOT_GREATER MQCFOP_NOT_LESS MQCFOP_EQUAL MQCFOP_LIKE MQCFOP_NOT_EQUAL MQCFOP_NOT_LIKE

## Subscription attribute equivalents

<i>Table 355. Subscription attributes for the REST API and equivalent PCF attributes.</i>			
REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
<code>name</code>	<b>MQCACF_SUB_NAME</b>		
<code>id</code>	<b>MQBACF_SUB_ID</b>		
<code>resolvedTopicString</code>	<b>MQCA_TOPIC_STRING</b>		
<code>topic.name</code>	<b>MQCA_TOPIC_NAME</b>		
<code>topic.definedString</code>	<b>MQCA_TOPIC_STRING</b>		
<code>selector.value</code>	<b>MQCACF_SUB_SELECTOR</b>		
<code>selector.type</code>	<b>MQIACF_SELECTOR_TYPE</b>	none standard extended	MQSELTYPE_NONE MQSELTYPE_STANDARD MQSELTYPE_EXTENDED
<code>destination.isManaged</code>	<b>MQIACF_DESTINATION_CLASS</b>	true false	MQDC_MANAGED MQDC_PROVIDED
<code>destination.qmgrName</code>	<b>MQCACF_DESTINATION_Q_MGR</b>		
<code>destination.name</code>	<b>MQCACF_DESTINATION</b>		
<code>destination.correlationId</code>	<b>MQBACF_DESTINATION_CORREL_ID</b>		
<code>user.accountingToken</code>	<b>MQBACF_ACCOUNTING_TOKEN</b>		

Table 355. Subscription attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
user.applicationId entityData	MQCACF_APPL_IDENTITY_DATA		
user.data	MQCACF_SUB_USER_DATA		
user.name	MQCACF_SUB_USER_ID		
user.isVariable	MQIACF_VARIABLE_USER_ID	true false	MQVU_ANY_USER MQVU_FIXED_USER
general.isDurable	MQIACF_DURABLE_SUBSCRIPTION	true false	MQSUB_DURABLE_YES MQSUB_DURABLE_NO
general.type	MQIACF_SUB_TYPE	administrative api proxy	MQSUBTYPE_ADMIN MQSUBTYPE_API MQSUBTYPE_PROXY
general.usesCharacterWildcard	MQIACF_WILDCARD_SCHEMA	true false	MQWS_CHAR MQWS_TOPIC
extended.expiry	MQIACF_EXPIRY		
extended.level	MQIACF_SUB_LEVEL		
extended.messagePriority	MQIACF_PUB_PRIORITY	asPublished asQueue	MQPRI_PRIORITY_AS_PUBLISHED MQPR_PRIORITY_AS_QUEUE
extended.messagePropertyControl	MQIACF_PUBSUB_PROPERTIES	none compatible pcf rfh2	MQPSPROP_NONE MQPSPROP_COMPAT MQPSPROP_MSGPROP MQPSPROP_RFH2
extended.deliverOnRequest	MQIACF_REQUEST_ONLY	true false	MQRU_PUBLISH_ON_REQUEST MQRU_PUBLISH_ALL
extended.networkScope	MQIACF_SUBSCRIPTION_SCOPE	all qmgr	MQTSCOPE_ALL MQTSCOPE_QMGR
timestamps.altered	MQCA_ALTERATION_DATE MQCA_ALTERATION_TIME		
timestamps.created	MQCA_CREATION_DATE MQCA_CREATION_TIME		

## Unsupported PCF parameters

The following subscription PCF inquire parameters are not supported by the administrative REST API:

- MQIA\_DISPLAY\_TYPE
- MQIACF\_SUB\_TYPE
- MQIACF\_SUB\_ATTRS

## REST API and PCF equivalents for channels

For most REST API optional query parameters and attributes for channels, an equivalent PCF parameter or attribute exists. Use the tables that are provided to understand these equivalents.

- [“Optional query parameter equivalents” on page 2390](#)
- [“Channel attribute equivalents” on page 2390](#)
- [“Unsupported PCF parameters” on page 2401](#)

## Optional query parameter equivalents

REST API optional query parameter	PCF parameter	Related values (REST API)	Related values (PCF)
filter= <i>filterValue</i>	MQCFT_INTEGER_FILTER MQCFT_STRING_FILTER	lessThan greaterThan lessThanOrEqualTo greaterThanOrEqualTo equalTo notEqualTo	MQCFOP_LESS MQCFOP_GREATER MQCFOP_NOT_GREATER MQCFOP_NOT_LESS MQCFOP_EQUAL MQCFOP_LIKE MQCFOP_NOT_EQUAL MQCFOP_NOT_LIKE
type= <i>type</i>	MQIACH_CHANNEL_TYPE	all sender receiver server requester clusterSender clusterReceiver	None. MQCHT_SENDER MQCHT_RECEIVER MQCHT_SERVER MQCHT_REQUESTER MQCHT_CLUSSDR MQCHT_CLUSRCVR
queueSharingGroupDisposition= <i>disposition</i>	MQIA_QSG_DISP	live all copy group private qmgr	MQQSGD_LIVE MQQSGD_ALL MQQSGD_COPY MQQSGD_GROUP MQQSGD_PRIVATE MQQSGD_Q_MGR

## Channel attribute equivalents

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
name	MQIACH_CHANNEL_NAME		

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
type	<b>MQIACH_CHANNEL_TYPE</b>		
clusterRouting.workloadPriority	<b>MQIACH_CLWL_CHANNEL_PRIORITY</b>		
clusterRouting.workloadRank	<b>MQIACH_CLWL_CHANNEL_RANK</b>		
clusterRouting.workloadWeight	<b>MQIACH_CLWL_CHANNEL_WEIGHT</b>		
clusterRouting.networkPriority	<b>MQIACH_NETWORK_PRIORITY</b>		
[type].connection.host [type].connection.port sender.connection.host sender.connection.port server.connection.host server.connection.port requester.connection.host requester.connection.port clusterSender.connection.host clusterSender.connection.port clusterReceiver.connection.host clusterReceiver.connection.port	<b>MQCACH_CONNECTION_NAME</b>		
[type].transmissionQueueName sender.transmissionQueueName server.transmissionQueueName	<b>MQCACH_XMIT_Q_NAME</b>		
clusterSender.clusterName clusterReceiver.clusterName	<b>MQCA_CLUSTER_NAME</b>		

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
clusterSender.clusterNameList clusterReceiver.clusterNameList	<b>MQCA_CLUSTER_NAMELIST</b>		
connectionManagement.heartbeatInterval	<b>MQIACH_HB_INTERVAL</b>		
connectionManagement.disconnectInterval	<b>MQIACH_DISC_INTERVAL</b>		
connectionManagement.keepAliveInterval	<b>MQIACH_KEEP_ALIVE_INTERVAL</b>		
connectionManagement.localAddress.host connectionManagement.localAddress.port connectionManagement.localAddress.portRange	<b>MQCACH_LOCAL_ADDRESS</b>		
connectionManagement.longRetry.count	<b>MQIACH_LONG_RETRY</b>		
connectionManagement.longRetry.interval	<b>MQIACH_LONG_TIMER</b>		
connectionManagement.shortRetry.count	<b>MQIACH_SHORT_RETRY</b>		
connectionManagement.shortRetry.interval	<b>MQIACH_SHORT_TIMER</b>		
compression.header	<b>MQIACH_HDR_COMPRESSION</b>	none system	MQCOMPRESS_NONE MQCOMPRESS_SYSTEM
compression.message	<b>MQIACH_MSG_COMPRESSION</b>	none runLengthEncoding zlibFast zlibHigh  any	MQCOMPRESS_NONE MQCOMPRESS_RLE MQCOMPRESS_ZLIBFAST MQCOMPRESS_ZLIBHIGH MQCOMPRESS_ANY



Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
dataCollection.monitoring	<b>MQIA_MONITORING_CHANNEL</b>	off asQmgr low medium high	MQMON_OFF MQMON_Q_MGR MQMON_LOW MQMON_MEDIUM MQMON_HIGH
dataCollection.statistics	<b>MQIA_STATISTICS_CHANNEL</b>	off asQmgr low medium high	MQMON_OFF MQMON_Q_MGR MQMON_LOW MQMON_MEDIUM MQMON_HIGH
exits.message.name	<b>MQCACH_MSG_EXIT_NAME</b>		
exits.message.userData	<b>MQCACH_MSG_EXIT_USER_DATA</b>		
exits.messageRetry.name	<b>MQCACH_MR_EXIT_NAME</b>		
exits.messageRetry.userData	<b>MQCACH_MR_EXIT_USER_DATA</b>		
exits.receive.name	<b>MQCACH_RCV_EXIT_NAME</b>		
exits.receive.userData	<b>MQCACH_RCV_EXIT_USER_DATA</b>		
exits.security.name	<b>MQCACH_SEC_EXIT_NAME</b>		
exits.security.userData	<b>MQCACH_SEC_EXIT_USER_DATA</b>		
exits.send.name	<b>MQCACH_SEND_EXIT_NAME</b>		
exits.send.userData	<b>MQCACH_SEND_EXIT_USER_DATA</b>		
extended.channelAgentType	<b>MQIACH_MCA_TYPE</b>	process thread	MQMCAT_PROCESS MQMCAT_THREAD
extended.senderDataConversion	<b>MQIACH_DATA_CONVERSION</b>	false true	MQCDC_NO_SENDER_CONVERSION MQCDC_SENDER_CONVERSION
extended.messagePropertyControl	<b>MQIA_PROPERTY_CONTROL</b>	compatible none all	MQPROP_COMPATIBILITY MQPROP_NONE MQPROP_ALL

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)


REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
extended.sequenceNumberWrap	<b>MQIACH_SEQUENCE_NUMBER_WRAP</b>		
 extended.securityPolicyProtection	<b>MQIACH_SPL_PROTECTION</b>	passThrough removeAsPolicy	MQSPL_PASSTHRU MQSPL_REMOVE MQSPL_AS_POLICY
failedDelivery.retry.count	<b>MQIACH_MR_COUNT</b>		
failedDelivery.retry.interval	<b>MQIACH_MR_INTERVAL</b>		
failedDelivery.useDeadLetterQueue	<b>MQIA_USE_DEAD_LETTER_Q</b>	true false	MQUSEDLQ_YES MQUSEDLQ_NO
general.description	<b>MQCACH_DESC</b>		
general.maximumMessageLength	<b>MQIACH_MAX_MSG_LENGTH</b>		
batch.preCommitHeartbeat	<b>MQIACH_BATCH_HB</b>		
batch.timeExtend	<b>MQIACH_BATCH_INTERVAL</b>		
batch.dataLimit	<b>MQIACH_BATCH_DATA_LIMIT</b>		
batch.messageLimit	<b>MQIACH_BATCH_SIZE</b>		
batch.nonPersistentMessageSpeedFast currentStatus.batch.nonPersistentMessageSpeedFast	<b>MQIACH_NPM_SPEED</b>	true false	MQNPMS_FAST MQNPMS_NORMAL
queueSharingGroup.disposition	<b>MQIA_QSG_DISP</b>	copy group qmgr	MQQSDG_COPY MQQSDG_GROUP MQQSDG_QMGR
queueSharingGroup.defaultChannelDisposition	<b>MQIACH_DEF_CHANNEL_DISP</b>	private fixShared shared	MQCHLD_PRIVATE MQCHLD_FIXSHARED MQCHLD_SHARED
receiverSecurity.channelAgentUserId	<b>MQCACH_MCA_USER_ID</b>		

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
receiverSecurity.outputAuthority	<b>MQCACH_MCA_USER_ID</b>	default context alternateOrChannelAgent onlyChannelAgent	MQPA_DEFAULT MQPA_CONTEXT MQPA_ALTERNATE_OR_MCA MQPA_ONLY_MCA
transmissionSecurity.certificateLabel	<b>MQCA_CERT_LABEL</b>		
transmissionSecurity.cipherSpecification	<b>MQCACH_SSL_CIPHER_SPEC</b>		
transmissionSecurity.requirePartnerCertificate	<b>MQIACH_SSL_CLIENT_AUTH</b>	true false	MQSCA_REQUIRED MQSCA_OPTIONAL
transmissionSecurity.certificatePeerName	<b>MQCACH_SSL_PEER_NAME</b>		
timestamps.altered	<b>MQCA_ALTERATION_DATE</b> <b>MQCA_ALTERATION_TIME</b>		
currentStatus.inDoubt savedStatus.inDoubt	<b>MQIACH_INDOUBT_STATUS</b>	true false	MQCHIDS_INDOUBT MQCHIDS_NOT_INDOUBT
currentStatus.state	<b>MQIACH_CHANNEL_STATUS</b>	binding starting running paused stopping retrying stopped requesting switching initializing	MQCHS_BINDING MQCHS_STARTING MQCHS_RUNNING MQCHS_PAUSED MQCHS_STOPPING MQCHS_RETRYING MQCHS_STOPPED MQCHS_REQUESTING MQCHS_SWITCHING MQCHS_INITIALIZING
currentStatus.agent.jobName	<b>MQCACH_MCA_JOB_NAME</b>		
currentStatus.agent.running	<b>MQIACH_MCA_STATUS</b>	true false	MQMCAS_RUNNING MQMCAS_STOPPED

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
currentStatus.agent.state	<b>MQIACH_CHANNEL_SUBSTATE</b>	runningChannelAutoDefinitionExit compressingData processingEndOfBatch performingSecurityHandshake heartbeating executingMQGET executingMQI executingMQPUT runningRetryExit runningMessageExit communicatingWithNameServer connectingToNetwork undefined runningReceiveExit receivingFromNetwork resynchingWithPartner runningSecurityExit runningSendExit sendingToNetwork serializingAccessToQmgr	MQCHSSTATE_CHADEXIT MQCHSSTATE_COMPRESSING MQCHSSTATE_END_OF_BATCH MQCHSSTATE_HANDSHAKING MQCHSSTATE_HEARTBEATING MQCHSSTATE_IN MQGET MQCHSSTATE_IN MQI MQCHSSTATE_IN MQPUT MQCHSSTATE_MREXIT MQCHSSTATE_MSGEXIT MQCHSSTATE_NAME_SERVER MQCHSSTATE_NET_CONNECTING MQCHSSTATE_OTHER MQCHSSTATE_RCVEXIT MQCHSSTATE_RECEIVING MQCHSSTATE_RESYNCHING MQCHSSTATE_SCYEXIT MQCHSSTATE_SENDEXIT MQCHSSTATE_SENDING MQCHSSTATE_SERIALIZING
currentStatus.agent.userId	<b>MQCACH_MCA_USER_ID</b>		
currentStatus.batch.count	<b>MQIACH_BATCHES</b>		
currentStatus.batch.currentMessages savedStatus.batch.currentMessages	<b>MQIACH_CURRENT_MESSAGES</b>		
currentStatus.batch.luwid.current savedStatus.batch.luwid.current	<b>MQCACH_CURRENT_LUWID</b>		

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
currentStatus.batch.luwid.last savedStatus.batch.luwid.last	<b>MQCACH_LAST_LUWID</b>		
currentStatus.batch.sequenceNumber.current savedStatus.batch.sequenceNumber.current	<b>MQIACH_CURRENT_SEQ_NUMBER</b>		
currentStatus.batch.sequenceNumber.last savedStatus.batch.sequenceNumber.last	<b>MQIACH_LAST_SEQ_NUMBER</b>		
currentStatus.batch.size	<b>MQIACH_BATCH_SIZE</b>		
currentStatus.compression.header.default currentStatus.compression.header.lastMessage	<b>MQIACH_HDR_COMPRESSION</b>	none system unavailable (applies to lastMessage only)	MQCOMPRESS_NONE MQCOMPRESS_SYSTEM MQCOMPRESS_NOT_AVAILABLE
currentStatus.compression.message.default currentStatus.compression.message.lastMessage	<b>MQIACH_MSG_COMPRESSION</b>	none runLengthEncoding zlibFast zlibHigh  unavailable (applies to lastMessage only)	MQCOMPRESS_NONE MQCOMPRESS_RLE MQCOMPRESS_ZLIBFAST MQCOMPRESS_ZLIBHIGH  MQCOMPRESS_NOT_AVAILABLE
currentStatus.connectionManagement.heartbeatInterval	<b>MQIACH_HB_INTERVAL</b>		
currentStatus.connectionManagement.keepAliveInterval	<b>MQIACH_KEEP_ALIVE_INTERVAL</b>		

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
currentStatus.connectionManagement.localAddress.host currentStatus.connectionManagement.localAddress.port	<b>MQCACH_LOCAL_ADDRESS</b>		
currentStatus.connectionManagement.remainingRetries.long	<b>MQIACH_LONG_RETRIES_LEFT</b>		
currentStatus.connectionManagement.remainingRetries.short	<b>MQIACH_SHORT_RETRIES_LEFT</b>		
currentStatus.extended.bufferReceived	<b>MQIACH_BUFFERS_RCV D</b>		
currentStatus.extended.bufferSent	<b>MQIACH_BUFFERS_SENT</b>		
currentStatus.extended.bytesReceived	<b>MQIACH_BYTES_RCVD</b>		
currentStatus.extended.bytesSent	<b>MQIACH_BYTES_SENT</b>		
currentStatus.extended.messageCount	<b>MQIACH_MSGS</b>		
currentStatus.general.connection.host currentStatus.general.connection.port savedStatus.general.connection.host	<b>MQCACH_CONNECTION_NAME</b>		
currentStatus.general.transmissionQueueName savedStatus.general.transmissionQueueName	<b>MQCACH_XMIT_Q_NAME</b>		
currentStatus.general.maximumMessageLength	<b>MQIACH_MAX_MSG_LENGTH</b>		

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
currentStatus.general.stopRequested	<b>MQIACH_STOP_REQUESTED</b>	true false	MQCHSR_STOP_REQUESTED MQCHSR_STOP_NOT_REQUESTED
currentStatus.general.statistics	<b>MQIA_STATISTICS_CHANNEL</b>	disabledByQmgr off low medium high	MQMON_NONE MQMON_OFF MQMON_Q_MGR MQMON_LOW MQMON_MEDIUM MQMON_HIGH
currentStatus.monitoring.messagesInBatch.shortSamplePeriod currentStatus.monitoring.messagesInBatch.longSamplePeriod	<b>MQIACH_BATCH_SIZE_INDICATOR</b>	-1	MQMON_NOT_AVAILABLE
currentStatus.monitoring.rate	<b>MQIA_MONITORING_CHANNEL</b>	off low medium high	MQMON_OFF MQMON_LOW MQMON_MEDIUM MQMON_HIGH
currentStatus.monitoring.messagesInBatch.shortSamplePeriod currentStatus.monitoring.messagesInBatch.longSamplePeriod	<b>MQIACH_COMPRESSION_RATE</b>	-1	MQMON_NOT_AVAILABLE
currentStatus.monitoring.compressionTime.shortSamplePeriod currentStatus.monitoring.compressionTime.longSamplePeriod	<b>MQIACH_COMPRESSION_TIME</b>	-1	MQMON_NOT_AVAILABLE

Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
currentStatus.monitoring.exitTime.shortSamplePeriod currentStatus.monitoring.exitTime.longSamplePeriod	<b>MQIACH_EXIT_TIME_INDICATOR</b>	-1	MQMON_NOT_AVAILABLE
currentStatus.monitoring.messagesAvailable	<b>MQIACH_XMITQ_MSGS_AVAILABLE</b>	-1	MQMON_NOT_AVAILABLE
currentStatus.monitoring.networkTime.shortSamplePeriod currentStatus.monitoring.networkTime.longSamplePeriod	<b>MQIACH_NETWORK_TIME_INDICATOR</b>	-1	MQMON_NOT_AVAILABLE
currentStatus.monitoring.transmissionQueueTime.shortSamplePeriod currentStatus.monitoring.transmissionQueueTime.longSamplePeriod	<b>MQIACH_XMITQ_TIME_INDICATOR</b>	-1	MQMON_NOT_AVAILABLE
currentStatus.partner.productId	<b>MQCACH_REMOTE_PRODUCT</b>	MQMM MQMV MQCC MQNM MQJB MQJM MQJN MQJU MQXC MQXD MQXN MQXM MQXU MQNU	MQMM MQMV MQCC MQNM MQJB MQJM MQJN MQJU MQXC MQXD MQXN MQXM MQXU MQNU
currentStatus.partner.qmgrName	<b>MQCA_REMOTE_Q_MGR_NAME</b>		
currentStatus.partner.version	<b>MQCACH_REMOTE_VERSION</b>		



Table 357. Channel attributes for the REST API and equivalent PCF attributes. (continued)

REST API attribute	PCF attribute	Related values (REST API)	Related values (PCF)
currentStatus.queueSharingGroup.channelDisposition savedStatus.queueSharingGroup.channelDisposition	<b>MQIACH_CHANNEL_DISPOSITION</b>	private shared fixShared	MQCHLD_PRIVATE MQCHLD_SHARED MQCHLD_FIXSHARED
currentStatus.timeStamps.started	<b>MQCACH_CHANNEL_START_DATE</b> <b>MQCACH_CHANNEL_START_TIME</b>		
currentStatus.timeStamps.lastMessage	<b>MQCACH_LAST_MESSAGE_DATE</b> <b>MQCACH_LAST_MESSAGE_TIME</b>		
currentStatus.transmissionSecurity.certificateIssuerName	<b>MQCACH_SSL_CERT_ISSUER_NAME</b>		
currentStatus.transmissionSecurity.certificateUserId	<b>MQCACH_SSL_CERT_USER_ID</b>		
currentStatus.transmissionSecurity.keyLastReset	<b>MQCACH_SSL_KEY_RESET_DATE</b> <b>MQCACH_SSL_KEY_RESET_TIME</b>		
currentStatus.transmissionSecurity.keyResetCount	<b>MQIACH_SSL_KEY_RESETS</b>		
currentStatus.transmissionSecurity.protocol	<b>MQCACH_SSL_CERT_USER_ID</b>	none sslV30 tlsV10 tlsV12	MQSECPROT_NONE MQSECPROT_SSLV30 MQSECPROT_TLSV10 MQSECPROT_TLSV12
currentStatus.transmissionSecurity.shortPeerName	<b>MQCACH_SSL_SHORT_PEER_NAME</b>		

### Unsupported PCF parameters

The following parameters are not supported by the administrative REST API:

- **MQIACH\_CLIENT\_CHANNEL\_WEIGHT**
- **MQIACH\_CONNECTION\_AFFINITY**
- **MQIACH\_DEF\_RECONNECT**

- MQIACH\_IN\_DOUBT\_IN
- MQIACH\_IN\_DOUBT\_OUT
- MQCACH\_LAST\_MSG\_TIME
- MQIACH\_MAX\_INSTANCES
- MQIACH\_MAX\_INSTS\_PER\_CLIENT
- MQCACH\_MODE\_NAME
- MQIACH\_MSGS\_RECEIVED/MQIACH\_MSGS\_RCVD
- MQIACH\_MSGS\_SENT
- MQCACH\_PASSWORD
- MQIACH\_SHARING\_CONVERSATIONS
- MQCACH\_TP\_NAME
- MQIACH\_XMIT\_PROTOCOL\_TYPE
- MQCACH\_USER\_ID

Multi

## IBM MQ Administration Interface reference

---

Reference information for the IBM MQ Administration Interface (MQAI).

### Related tasks

[Using the MQAI to simplify the use of PCFs](#)

Multi

## MQAI calls

Reference information for MQAI calls.

There are two types of selector: *user selector* and *system selector*. These are described in [“MQAI selectors”](#) on page 2483.

There are three types of call:

- Data-bag manipulation calls for configuring data bags:
  - [“mqAddBag”](#) on page 2403
  - [“mqAddByteString”](#) on page 2405
  - [“mqAddByteStringFilter”](#) on page 2406
  - [“mqAddInquiry”](#) on page 2408
  - [“mqAddInteger”](#) on page 2410
  - [“mqAddInteger64”](#) on page 2412
  - [“mqAddIntegerFilter”](#) on page 2413
  - [“mqAddString”](#) on page 2415
  - [“mqAddStringFilter”](#) on page 2417
  - [“mqClearBag”](#) on page 2423
  - [“mqCountItems”](#) on page 2424
  - [“mqCreateBag”](#) on page 2425
  - [“mqDeleteBag”](#) on page 2428
  - [“mqDeleteItem”](#) on page 2430
  - [“mqInquireBag”](#) on page 2438
  - [“mqInquireByteString”](#) on page 2440
  - [“mqInquireByteStringFilter”](#) on page 2443
  - [“mqInquireInteger”](#) on page 2446

- [“mqInquireInteger64” on page 2448](#)
- [“mqInquireIntegerFilter” on page 2450](#)
- [“mqInquireItemInfo” on page 2452](#)
- [“mqInquireString” on page 2455](#)
- [“mqInquireStringFilter” on page 2458](#)
- [“mqSetByteString” on page 2463](#)
- [“mqSetByteStringFilter” on page 2466](#)
- [“mqSetInteger” on page 2469](#)
- [“mqSetInteger64” on page 2471](#)
- [“mqSetIntegerFilter” on page 2473](#)
- [“mqSetString” on page 2475](#)
- [“mqSetStringFilter” on page 2478](#)
- [“mqTruncateBag” on page 2482](#)
- Command calls for sending and receiving administration commands and PCF messages:
  - [“mqBagToBuffer” on page 2419](#)
  - [“mqBufferToBag” on page 2421](#)
  - [“mqExecute” on page 2432](#)
  - [“mqGetBag” on page 2436](#)
  - [“mqPutBag” on page 2462](#)
- Utility calls for handling blank-padded and null-terminated strings:
  - [“mqPad” on page 2460](#)
  - [“mqTrim” on page 2481](#)

These calls are described in alphabetical order in the following sections.

## **mqAddBag**

The mqAddBag call nests a bag in another bag.

### **Syntax for mqAddBag**

**mqAddBag** (*Bag, Selector, ItemValue, CompCode, Reason*)

### **Parameters for mqAddBag**

#### **Bag (MQHBAG) - input**

Bag handle into which the item is to be added.

The bag must be a user bag. This means that it must have been created using the MQCBO\_USER\_BAG option on the mqCreateBag call. If the bag was not created in this way, MQRC\_WRONG\_BAG\_TYPE results.

#### **Selector (MQLONG) - input**

Selector identifying the item to be nested.

If the selector is less than zero (that is, a system selector), MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If the selector is zero or greater (that is, a user selector) and the bag was created with the MQCBO\_CHECK\_SELECTORS option, the selector must be in the range MQGA\_FIRST through MQGA\_LAST; if not, again MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value of zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the data type of this occurrence must be the same as the data type of the first occurrence; MQRC\_INCONSISTENT\_ITEM\_TYPE results if it is not.

#### **ItemValue (MQHBAG) - input**

The bag which is to be nested.

If the bag is not a group bag, MQRC\_BAG\_WRONG\_TYPE results. If an attempt is made to add a bag to itself, MQRC\_HBAG\_ERROR results.

#### **CompCode (MQLONG) - output**

Completion code.

#### **Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddBag call:

##### **MQRC\_BAG\_WRONG\_TYPE**

Wrong type of bag for intended use (either Bag or ItemValue).

##### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

##### **MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of this occurrence of selector differs from data type of first occurrence.

##### **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

##### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

### **Usage notes for mqAddBag**

If a bag with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.

### **C language invocation for mqAddBag**

```
mqAddBag (Bag, Selector, ItemValue, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG Bag; /* Bag handle */
MQLONG Selector; /* Selector */
MQHBAG ItemValue; /* Nested bag handle */
MQLONG CompCode; /* Completion code */
MQLONG Reason; /* Reason code qualifying CompCode */
```

### **Visual Basic invocation for mqAddBag**

(Supported on Windows only.)

```
mqAddGroup Bag, Selector, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag As Long 'Bag handle'
```

```
Dim Selector As Long 'Selector'
Dim ItemValue As Long 'Nested bag handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

**Note:** The mqAddBag call can be used with user bags only; you cannot add nested bags to administration or command bags. You can only nest group bags.

## **Multi** mqAddByteString

The mqAddByteString call adds a byte string identified by a user selector to the end of a specified bag.

### Syntax for mqAddByteString

**mqAddByteString** (*Bag*, *Selector*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

### Parameters for mqAddByteString

#### **Bag (MQHBAG) - input**

Handle of the bag to be modified.

This value must be the handle of a bag created by the user, not the handle of a system bag. MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if the value you specify relates to a system bag.

#### **Selector (MQLONG) - input**

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQBA\_FIRST through MQBA\_LAST. MQRC\_SELECTOR\_OUT\_OF\_RANGE results if it is not in the correct range.

If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the data type of this occurrence must be the same as the data type of the first occurrence; MQRC\_INCONSISTENT\_ITEM\_TYPE results if it is not.

#### **BufferLength (MQLONG) - input**

The length in bytes of the string contained in the **Buffer** parameter. The value must be zero or greater.

#### **Buffer (MQBYTE - BufferLength) - input**

Buffer containing the byte string.

The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter. In all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

#### **CompCode (MQLONG) - output**

Completion code.

#### **Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqAddByteString call:

**MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of this occurrence of selector differs from data type of first occurrence.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**Usage notes for mqAddByteString**

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

**C language invocation for mqAddByteString**

```
mqAddByteString (hBag, Selector, BufferLength, Buffer, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */
MQLONG   Selector;      /* Selector */
MQLONG   BufferLength;  /* Buffer length */
PMQBYTE  Buffer;        /* Buffer containing item value */
MQLONG   CompCode;     /* Completion code */
MQLONG   Reason;       /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqAddByteString**

(Supported on Windows only.)

```
mqAddByteString Bag, Selector, BufferLength, Buffer, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long 'Bag handle'
Dim Selector      As Long 'Selector'
Dim BufferLength  As Long 'Buffer length'
Dim Buffer        As Byte 'Buffer containing item value'
Dim CompCode     As Long 'Completion code'
Dim Reason       As Long 'Reason code qualifying CompCode'
```

**Multi**

**mqAddByteStringFilter**

The mqAddByteStringFilter call adds a byte string filter identified by a user selector to the end of a specified bag.

## Syntax for mqAddByteStringFilter

**mqAddByteStringFilter** (*Bag*, *Selector*, *BufferLength*, *Buffer*, *Operator*, *CompCode*, *Reason*)

### Parameters for mqAddByteStringFilter

#### Bag (MQHBAG) - input

Handle of the bag to be modified.

This value must be the handle of a bag created by the user, not the handle of a system bag.

MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if the value you specify relates to a system bag.

#### Selector (MQLONG) - input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQBA\_FIRST through MQBA\_LAST. MQRC\_SELECTOR\_OUT\_OF\_RANGE results if it is not in the correct range.

If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the data type of this occurrence must be the same as the data type of the first occurrence; MQRC\_INCONSISTENT\_ITEM\_TYPE results if it is not.

#### BufferLength (MQLONG) - input

The length in bytes of the condition byte string contained in the **Buffer** parameter. The value must be zero or greater.

#### Buffer (MQBYTE x BufferLength) - input

Buffer containing the condition byte string.

The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter. In all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

#### Operator (MQLONG) - input

The byte string filter operator to be placed in the bag. Valid operators are of the form MQCFOP\_\*

#### CompCode (MQLONG) - output

Completion code.

#### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqAddByteStringFilter call:

##### **MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not accessible).

##### **MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

##### **MQRC\_FILTER\_OPERATOR\_ERROR**

Filter operator not valid.

##### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of this occurrence of selector differs from data type of first occurrence.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**Usage notes for mqAddByteStringFilter**

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

**C language invocation for mqAddByteStringFilter**

```
mqAddByteStringFilter (hBag, Selector, BufferLength, Buffer, Operator,
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    hBag;           /* Bag handle */
MQLONG    Selector;       /* Selector */
MQLONG    BufferLength;   /* Buffer length */
PMQBYTE   Buffer;         /* Buffer containing item value */
MQLONG    Operator;      /* Operator */
PMQLONG   CompCode;      /* Completion code */
PMQLONG   Reason;        /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqAddByteStringFilter**

(Supported on Windows only.)

```
mqAddByteStringFilter Bag, Selector, BufferLength, Buffer, Operator, CompCode,
Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long 'Bag handle'
Dim Selector      As Long 'Selector'
Dim BufferLength  As Long 'Buffer length'
Dim Buffer        As String 'Buffer containing item value'
Dim Operator     As Long 'Operator'
Dim CompCode     As Long 'Completion code'
Dim Reason       As Long 'Reason code qualifying CompCode'
```

**Multi mqAddInquiry**

The mqAddInquiry call can be used with administration bags only; it is specifically for administration purposes.

The mqAddInquiry call adds a selector to an administration bag. The selector refers to an IBM MQ object attribute that is to be returned by a PCF INQUIRE command. The value of the **Selector** parameter specified on this call is added to the end of the bag, as the value of a data item that has the selector value MQIACF\_INQUIRY.



## Syntax for mqAddInquiry

**mqAddInquiry** (*Bag*, *Selector*, *CompCode*, *Reason*)

## Parameters for mqAddInquiry

### Bag (MQHBAG) - input

Bag handle.

The bag must be an administration bag; that is, it must have been created with the MQCBO\_ADMIN\_BAG option on the mqCreateBag call. If the bag was not created this way, MQRC\_BAG\_WRONG\_TYPE results.

### Selector (MQLONG) - input

Selector of the IBM MQ object attribute that is to be returned by the appropriate INQUIRE administration command.

### CompCode (MQLONG) - output

Completion code.

### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddInquiry call:

#### **MQRC\_BAG\_WRONG\_TYPE**

Wrong type of bag for intended use.

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

#### **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

#### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

#### **MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

## Usage notes for mqAddInquiry

1. When the administration message is generated, the MQAI constructs an integer list with the MQIACF\_\*\_ATTRS or MQIACH\_\*\_ATTRS selector that is appropriate to the Command value specified on the mqExecute, mqPutBag, or mqBagToBuffer call. It then adds the values of the attribute selectors specified by the mqAddInquiry call.
2. If the Command value specified on the mqExecute, mqPutBag, or mqBagToBuffer call is not recognized by the MQAI, MQRC\_INQUIRY\_COMMAND\_ERROR results. Instead of using the mqAddInquiry call, this can be overcome by using the mqAddInteger call with the appropriate MQIACF\_\*\_ATTRS or MQIACH\_\*\_ATTRS selector and the **ItemValue** parameter of the selector being inquired.

## C language invocation for mqAddInquiry

```
mqAddInquiry (Bag, Selector, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */
MQLONG   Selector;      /* Selector */
MQLONG   CompCode;     /* Completion code */
MQLONG   Reason;       /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqAddInquiry

(Supported on Windows only.)

```
mqAddInquiry Bag, Selector, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

### Supported INQUIRE command codes

- MQCMD\_INQUIRE\_AUTH\_INFO
- MQCMD\_INQUIRE\_AUTH\_RECS
- MQCMD\_INQUIRE\_AUTH\_SERVICE
- MQCMD\_INQUIRE\_CHANNEL
- MQCMD\_INQUIRE\_CHANNEL\_STATUS
- MQCMD\_INQUIRE\_CLUSTER\_Q\_MGR
- MQCMD\_INQUIRE\_CONNECTION
- MQCMD\_INQUIRE\_LISTENER
- MQCMD\_INQUIRE\_LISTENER\_STATUS
- MQCMD\_INQUIRE\_NAMELIST
- MQCMD\_INQUIRE\_PROCESS
- MQCMD\_INQUIRE\_Q
- MQCMD\_INQUIRE\_Q\_MGR
- MQCMD\_INQUIRE\_Q\_MGR\_STATUS
- MQCMD\_INQUIRE\_Q\_STATUS
- MQCMD\_INQUIRE\_SECURITY

For an example that demonstrates the use of supported INQUIRE command codes, see [Inquiring about queues and printing information \(amqsailq.c\)](#).

Multi

### mqAddInteger

The mqAddInteger call adds an integer item identified by a user selector to the end of a specified bag.

### Syntax for mqAddInteger

**mqAddInteger** (*Bag, Selector, ItemValue, CompCode, Reason*)

### Parameters for mqAddInteger

#### Bag (MQHBAG) - input

Handle of the bag to be modified.

This must be the handle of a bag created by the user, not the handle of a system bag. MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if the value you specify identifies a system bag.

### Selector (MQLONG)

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If the selector is zero or greater (that is, a user selector) and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQIA\_FIRST through MQIA\_LAST; if not, again MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value of zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the data type of this occurrence must be the same as the data type of the first occurrence; MQRC\_INCONSISTENT\_ITEM\_TYPE results if it is not.

### ItemValue (MQLONG) - input

The integer value to be placed in the bag.

### CompCode (MQLONG) - output

Completion code.

### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddInteger call:

#### MQRC\_HBAG\_ERROR

Bag handle not valid.

#### MQRC\_INCONSISTENT\_ITEM\_TYPE

Data type of this occurrence of selector differs from data type of first occurrence.

#### MQRC\_SELECTOR\_OUT\_OF\_RANGE

Selector not within valid range for call.

#### MQRC\_STORAGE\_NOT\_AVAILABLE

Insufficient storage available.

#### MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE

System bag cannot be altered or deleted.

## Usage notes for mqAddInteger

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily next to the existing instance.
2. This call cannot be used to add a system selector to a bag.

## C language invocation for mqAddInteger

```
mqAddInteger (Bag, Selector, ItemValue, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG  Bag;          /* Bag handle */
MQLONG  Selector;     /* Selector */
MQLONG  ItemValue;    /* Integer value */
MQLONG  CompCode;     /* Completion code */
MQLONG  Reason;       /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqAddInteger

(Supported on Windows only.)

```
mqAddInteger Bag, Selector, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag          As Long 'Bag handle'  
Dim Selector     As Long 'Selector'  
Dim ItemValue   As Long 'Integer value'  
Dim CompCode    As Long 'Completion code'  
Dim Reason      As Long 'Reason code qualifying CompCode'
```

### mqAddInteger64

The mqAddInteger64 call adds a 64-bit integer item identified by a user selector to the end of a specified bag.

### Syntax for mqAddInteger64

**mqAddInteger64** (*Bag, Selector, ItemValue, CompCode, Reason*)

### Parameters for mqAddInteger64

#### Bag (MQHBAG) - input

Handle of the bag to be modified.

This must be the handle of a bag created by the user, not the handle of a system bag. MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if the value you specify identifies a system bag.

#### Selector (MQLONG) - input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If the selector is zero or greater (that is, a user selector) and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQIA\_FIRST through MQIA\_LAST; if not, again MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value of zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the data type of this occurrence must be the same as the data type of the first occurrence; MQRC\_INCONSISTENT\_ITEM\_TYPE results if it is not.

#### ItemValue (MQINT64) - input

The 64-bit integer value to be placed in the bag.

#### CompCode (MQLONG) - output

Completion code.

#### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the `mqAddInteger64` call:

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of this occurrence of selector differs from data type of first occurrence.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

## Usage notes for `mqAddInteger64`

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

## C language invocation for `mqAddInteger64`

```
mqAddInteger64 (Bag, Selector, ItemValue, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG    Bag;        /* Bag handle */
MQLONG    Selector;   /* Selector */
MQINT64   ItemValue;  /* Integer value */
MQLONG    CompCode;   /* Completion code */
MQLONG    Reason;     /* Reason code qualifying CompCode */
```

## Visual Basic invocation for `mqAddInteger64`

(Supported on Windows only.)

```
mqAddInteger64 Bag, Selector, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim Item Value As Long 'Integer value'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

Multi

## `mqAddIntegerFilter`

The `mqAddIntegerFilter` call adds an integer filter identified by a user selector to the end of a specified bag.

## Syntax for `mqAddIntegerFilter`

`mqAddIntegerFilter (Bag, Selector, ItemValue, Operator, CompCode, Reason)`

## Parameters for mqAddIntegerFilter

### Bag (MQHBAG) - input

Handle of the bag to be modified.

This must be the handle of a bag created by the user, not the handle of a system bag. MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if the value you specify identifies a system bag.

### Selector (MQLONG) - input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If the selector is zero or greater (that is, a user selector) and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQIA\_FIRST through MQIA\_LAST; if not, again MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value of zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the data type of this occurrence must be the same as the data type of the first occurrence; MQRC\_INCONSISTENT\_ITEM\_TYPE results if it is not.

### ItemValue (MQLONG) - input

The integer condition value to be placed in the bag.

### Operator (MQLONG) - input

The integer filter operator to be placed in the bag. Valid operators take the form MQCFOP\_\*

### CompCode (MQLONG) - output

Completion code.

### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddIntegerFilter call:

#### **MQRC\_FILTER\_OPERATOR\_ERROR**

Filter operator not valid.

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

#### **MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of this occurrence of selector differs from data type of first occurrence.

#### **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

#### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

#### **MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

## Usage notes for mqAddIntegerFilter

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

## C language invocation for mqAddIntegerFilter

```
mqAddIntegerFilter (Bag, Selector, ItemValue, Operator, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG  Bag;          /* Bag handle */
MQLONG  Selector;     /* Selector */
MQLONG  ItemValue;    /* Integer value */
MQLONG  Operator;     /* Item operator */
MQLONG  CompCode;    /* Completion code */
MQLONG  Reason;      /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqAddIntegerFilter

(Supported on Windows only.)

```
mqAddIntegerFilter Bag, Selector, ItemValue, Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemValue As Long 'Integer value'
Dim Operator As Long 'Item Operator'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

## mqAddString

The mqAddString call adds a character data item identified by a user selector to the end of a specified bag.

### Syntax for mqAddString

```
mqAddString (Bag, Selector, BufferLength, Buffer, CompCode, Reason)
```

### Parameters for mqAddString

#### Bag (MQHBAG) - input

Handle of the bag to be modified.

This value must be the handle of a bag created by the user, not the handle of a system bag. MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if the value you specify relates to a system bag.

#### Selector (MQLONG) - input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQCA\_FIRST through MQCA\_LAST. MQRC\_SELECTOR\_OUT\_OF\_RANGE results if it is not in the correct range.

If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the data type of this occurrence must be the same as the data type of the first occurrence; MQRC\_INCONSISTENT\_ITEM\_TYPE results if it is not.

**BufferLength (MQLONG) - input**

The length in bytes of the string contained in the **Buffer** parameter. The value must be zero or greater, or the special value MQBL\_NULL\_TERMINATED:

- If MQBL\_NULL\_TERMINATED is specified, the string is delimited by the first null encountered in the string. The null is not added to the bag as part of the string.
- If MQBL\_NULL\_TERMINATED is not specified, *BufferLength* characters are inserted into the bag, even if null characters are present. Nulls do not delimit the string.

**Buffer (MQCHAR x BufferLength) - input**

Buffer containing the character string.

The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter. In all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqAddString call:

**MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

**MQRC\_CODED\_CHAR\_SET\_ID\_ERROR**

Bag CCSID is MQCCSI\_EMBEDDED.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of this occurrence of selector differs from data type of first occurrence.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**Usage notes for mqAddString**

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.
3. The Coded Character Set ID associated with this string is copied from the current CCSID of the bag.

**C language invocation for mqAddString**

```
mqAddString (hBag, Selector, BufferLength, Buffer, &CompCode, &Reason);
```



Declare the parameters as follows:

```
MQHBAG    hBag;           /* Bag handle */
MQLONG    Selector;      /* Selector */
MQLONG    BufferLength;  /* Buffer length */
PMQCHAR   Buffer         /* Buffer containing item value */
MQLONG    CompCode;     /* Completion code */
MQLONG    Reason;       /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqAddString

(Supported on Windows only.)

```
mqAddString Bag, Selector, BufferLength, Buffer, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long 'Bag handle'
Dim Selector      As Long 'Selector'
Dim BufferLength  As Long 'Buffer length'
Dim Buffer         As String 'Buffer containing item value'
Dim CompCode     As Long 'Completion code'
Dim Reason       As Long 'Reason code qualifying CompCode'
```

## mqAddStringFilter

The mqAddStringFilter call adds a string filter identified by a user selector to the end of a specified bag.

### Syntax for mqAddStringFilter

**mqAddStringFilter** (*Bag, Selector, BufferLength, Buffer, Operator, CompCode, Reason*)

### Parameters for mqAddStringFilter

#### Bag (MQHBAG) - input

Handle of the bag to be modified.

This value must be the handle of a bag created by the user, not the handle of a system bag. MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if the value you specify relates to a system bag.

#### Selector (MQLONG) - input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC\_SELECTOR\_OUT\_OF\_RANGE results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQCA\_FIRST through MQCA\_LAST. MQRC\_SELECTOR\_OUT\_OF\_RANGE results if it is not in the correct range.

If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the data type of this occurrence must be the same as the data type of the first occurrence; MQRC\_INCONSISTENT\_ITEM\_TYPE results if it is not.

**BufferLength (MQLONG) - input**

The length in bytes of the character condition string contained in the **Buffer** parameter. The value must be zero or greater, or the special value MQBL\_NULL\_TERMINATED:

- If MQBL\_NULL\_TERMINATED is specified, the string is delimited by the first null encountered in the string. The null is not added to the bag as part of the string.
- If MQBL\_NULL\_TERMINATED is not specified, *BufferLength* characters are inserted into the bag, even if null characters are present. Nulls do not delimit the string.

**Buffer (MQCHAR x BufferLength) - input**

Buffer containing the character condition string.

The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter. In all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

**Operator (MQLONG) - input**

The string filter operator to be placed in the bag. Valid operators are of the form MQCFOP\_\*

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqAddStringFilter call:

**MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

**MQRC\_CODED\_CHAR\_SET\_ID\_ERROR**

Bag CCSID is MQCCSI\_EMBEDDED.

**MQRC\_FILTER\_OPERATOR\_ERROR**

Filter operator not valid.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of this occurrence of selector differs from data type of first occurrence.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**Usage notes for mqAddStringFilter**

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.
3. The Coded Character Set ID associated with this string is copied from the current CCSID of the bag.

## C language invocation for mqAddStringFilter

```
mqAddStringFilter (hBag, Selector, BufferLength, Buffer, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    hBag;           /* Bag handle */
MQLONG    Selector;       /* Selector */
MQLONG    BufferLength;   /* Buffer length */
PMQCHAR   Buffer;         /* Buffer containing item value */
MQLONG    Operator;      /* Operator */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqAddStringFilter

(Supported on Windows only.)

```
mqAddStringFilter Bag, Selector, BufferLength, Buffer, Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long 'Bag handle'
Dim Selector      As Long 'Selector'
Dim BufferLength  As Long 'Buffer length'
Dim Buffer         As String 'Buffer containing item value'
Dim Operator      As Long 'Item operator'
Dim CompCode     As Long 'Completion code'
Dim Reason        As Long 'Reason code qualifying CompCode'
```

## mqBagToBuffer

The mqBagToBuffer call converts the bag into a PCF message in the supplied buffer.

### Syntax for mqBagToBuffer

**mqBagToBuffer** (*OptionsBag, DataBag, BufferLength, Buffer, DataLength, CompCode, Reason*)

### Parameters for mqBagToBuffer

#### OptionsBag (MQHBAG) - input

Handle of the bag containing options that control the processing of the call. This is a reserved parameter; the value must be MQHB\_NONE.

#### DataBag (MQHBAG) - input

The handle of the bag to convert.

If the bag contains an administration message and mqAddInquiry was used to insert values into the bag, the value of the MQIASY\_COMMAND data item must be an INQUIRE command that is recognized by the MQAI; MQRC\_INQUIRY\_COMMAND\_ERROR results if it is not.

If the bag contains nested system bags, MQRC\_NESTED\_BAG\_NOT\_SUPPORTED results.

#### BufferLength (MQLONG) - input

Length in bytes of the buffer supplied.

If the buffer is too small to accommodate the message generated, MQRC\_BUFFER\_LENGTH\_ERROR results.

**Buffer (MQBYTE x BufferLength) - output**

The buffer to hold the message.

**DataLength (MQLONG) - output**

The length in bytes of the buffer required to hold the entire bag. If the buffer is not long enough, the contents of the buffer are undefined but the DataLength is returned.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqBagToBuffer call:

**MQRC\_BAG\_WRONG\_TYPE**

Input data bag is a group bag.

**MQRC\_BUFFER\_ERROR**

**Buffer** parameter not valid (invalid parameter address or buffer not accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid or buffer too small. (Required length returned in *DataLength*.)

**MQRC\_DATA\_LENGTH\_ERROR**

**DataLength** parameter not valid (invalid parameter address).

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INQUIRY\_COMMAND\_ERROR**

mqAddInquiry used with a command code that is not recognized as an INQUIRE command.

**MQRC\_NESTED\_BAG\_NOT\_SUPPORTED**

Input data bag contains one or more nested system bags.

**MQRC\_OPTIONS\_ERROR**

Options bag contains unsupported data items or a supported option has an invalid value.

**MQRC\_PARAMETER\_MISSING**

An administration message requires a parameter that is not present in the bag.

**Note:** This reason code occurs for bags created with the MQCBO\_ADMIN\_BAG or MQCBO\_REORDER\_AS\_REQUIRED options only.

**MQRC\_SELECTOR\_WRONG\_TYPE**

mqAddString or mqSetString was used to add the MQIACF\_INQUIRY selector to the bag.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**Usage notes for mqBagToBuffer**

1. The PCF message is generated with an encoding of MQENC\_NATIVE for the numeric data.
2. The buffer that holds the message can be null if the BufferLength is zero. This is useful if you use the mqBagToBuffer call to calculate the size of buffer necessary to convert your bag.

**C language invocation for mqBagToBuffer**

```
mqBagToBuffer (OptionsBag, DataBag, BufferLength, Buffer, &DataLength,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG OptionsBag;      /* Options bag handle */
MQHBAG DataBag;        /* Data bag handle */
MQLONG BufferLength;   /* Buffer length */
MQBYTE Buffer[n];      /* Buffer to contain PCF */
MQLONG DataLength;    /* Length of PCF returned in buffer */
MQLONG CompCode;      /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqBagToBuffer

(Supported on Windows only.)

```
mqBagToBuffer OptionsBag, DataBag, BufferLength, Buffer, DataLength,
CompCode, Reason
```

Declare the parameters as follows:

```
Dim OptionsBag As Long 'Options bag handle'
Dim DataBag As Long 'Data bag handle'
Dim BufferLength As Long 'Buffer length'
Dim Buffer As Long 'Buffer to contain PCF'
Dim DataLength As Long 'Length of PCF returned in buffer'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

Multi

## mqBufferToBag

The mqBufferToBag call converts the supplied buffer into bag form.

### Syntax for mqBufferToBag

**mqBufferToBag** (*OptionsBag, BufferLength, Buffer, DataBag, CompCode, Reason*)

### Parameters for mqBufferToBag

#### OptionsBag (MQHBAG) - input

Handle of the bag containing options that control the processing of the call. This is a reserved parameter; the value must be MQHB\_NONE.

#### BufferLength (MQLONG) - input

Length in bytes of the buffer.

#### Buffer (MQBYTE x BufferLength) - input

Pointer to the buffer containing the message to be converted.

#### Databag (MQHBAG) - input/output

Handle of the bag to receive the message. The MQAI performs an mqClearBag call on the bag before placing the message in the bag.

#### CompCode (MQLONG) - output

Completion code.

#### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqBufferToBag call:

**MQRC\_BAG\_CONVERSION\_ERROR**

Data could not be converted into a bag. This indicates a problem with the format of the data to be converted into a bag (for example, the message is not a valid PCF).

**MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of second occurrence of selector differs from data type of first occurrence.

**MQRC\_OPTIONS\_ERROR**

Options bag contains unsupported data items, or a supported option has a value that is not valid.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

## Usage notes for mqBufferToBag

The buffer must contain a valid PCF message. The encoding of numeric data in the buffer must be MQENC\_NATIVE.

The Coded Character Set ID of the bag is unchanged by this call.

## C language invocation for mqBufferToBag

```
mqBufferToBag (OptionsBag, BufferLength, Buffer, DataBag,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG OptionsBag; /* Options bag handle */  
MQLONG BufferLength; /* Buffer length */  
MQBYTE Buffer[n]; /* Buffer containing PCF */  
MQHBAG DataBag; /* Data bag handle */  
MQLONG CompCode; /* Completion code */  
MQLONG Reason; /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqBufferToBag

(Supported on Windows only.)

```
mqBufferToBag OptionsBag, BufferLength, Buffer, DataBag,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim OptionsBag As Long 'Options bag handle'  
Dim BufferLength As Long 'Buffer length'  
Dim Buffer As Long 'Buffer containing PCF'  
Dim DataBag As Long 'Data bag handle'
```

```
Dim CompCode      As Long 'Completion code'
Dim Reason        As Long 'Reason code qualifying CompCode'
```

## Multi **mqClearBag**

The `mqClearBag` call deletes all user items from the bag, and resets system items to their initial values.

### Syntax for `mqClearBag`

`mqClearBag (Bag, CompCode, Reason)`

### Parameters for `mqClearBag`

#### Bag (MQHBAG) - input

Handle of the bag to be cleared. This must be the handle of a bag created by the user, not the handle of a system bag. `MQRC_SYSTEM_BAG_NOT_ALTERABLE` results if you specify the handle of a system bag.

#### CompCode (MQLONG) - output

Completion code.

#### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the `mqClearBag` call:

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

#### **MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

### Usage notes for `mqClearBag`

1. If the bag contains system bags, they are also deleted.
2. The call cannot be used to clear system bags.

### C language invocation for `mqClearBag`

```
mqClearBag (Bag, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG  Bag;           /* Bag handle */
MQLONG  CompCode;      /* Completion code */
MQLONG  Reason;        /* Reason code qualifying CompCode */
```

### Visual Basic invocation for `mqClearBag`

(Supported on Windows only.)

```
mqClearBag Bag, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

## Multi **mqCountItems**

The `mqCountItems` call returns the number of occurrences of user items, system items, or both, that are stored in a bag with the same specific selector.

### Syntax for `mqCountItems`

`mqCountItems (Bag, Selector, ItemCount, CompCode, Reason)`

### Parameters for `mqCountItems`

#### Bag (MQHBAG) - input

Handle of the bag with items that are to be counted. This can be a user bag or a system bag.

#### Selector (MQLONG) - input

Selector of the data items to count.

If the selector is less than zero (a system selector), the selector must be one that is supported by the MQAI. MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

If the specified selector is not present in the bag, the call succeeds and zero is returned for *ItemCount*.

The following special values can be specified for *Selector*:

#### **MQSEL\_ALL\_SELECTORS**

All user and system items are to be counted.

#### **MQSEL\_ALL\_USER\_SELECTORS**

All user items are to be counted; system items are excluded from the count.

#### **MQSEL\_ALL\_SYSTEM\_SELECTORS**

All system items are to be counted; user items are excluded from the count.

#### ItemCount (MQLONG) - output

Number of items of the specified type in the bag (can be zero).

#### CompCode (MQLONG) - output

Completion code.

#### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the `mqCountItems` call:

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

#### **MQRC\_ITEM\_COUNT\_ERROR**

*ItemCount* parameter not valid (invalid parameter address).

#### **MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.



## **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

## **Usage notes for mqCountItems**

This call counts the number of data items, not the number of unique selectors in the bag. A selector can occur multiple times, so there might be fewer unique selectors in the bag than data items.

## **C language invocation for mqCountItems**

```
mqCountItems (Bag, Selector, &ItemCount, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */
MQLONG Selector;      /* Selector */
MQLONG ItemCount;     /* Number of items */
MQLONG CompCode;      /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

## **Visual Basic invocation for mqCountItems**

(Supported on Windows only.)

```
mqCountItems Bag, Selector, ItemCount, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag;           As Long 'Bag handle'
Dim Selector       As Long 'Selector'
Dim ItemCount      As Long 'Number of items'
Dim CompCode       As Long 'Completion code'
Dim Reason         As Long 'Reason code qualifying CompCode'
```

## **Multi mqCreateBag**

The mqCreateBag call creates a new bag.

## **Syntax for mqCreateBag**

**mqCreateBag** (*Options*, *Bag*, *CompCode*, *Reason*)

## **Parameters for mqCreateBag**

### **Options (MQLONG) - input**

Options for creation of the bag.

The following values are valid:

#### **MQCBO\_ADMIN\_BAG**

Specifies that the bag is for administering IBM MQ objects. MQCBO\_ADMIN\_BAG automatically implies the MQCBO\_LIST\_FORM\_ALLOWED, MQCBO\_REORDER\_AS\_REQUIRED, and MQCBO\_CHECK\_SELECTORS options.

Administration bags are created with the MQIASY\_TYPE system item set to MQCFT\_COMMAND.

### **MQCBO\_COMMAND\_BAG**

Specifies that the bag is a command bag. MQCBO\_COMMAND\_BAG is an alternative to the administration bag (MQCBO\_ADMIN\_BAG) and MQRC\_OPTIONS\_ERROR results if both are specified.

A command bag is processed in the same way as a user bag except that the value of the MQIASY\_TYPE system item is set to MQCFT\_COMMAND when the bag is created.

The command bag is also created for administering objects but they are not used to send administration messages to a command server as an administration bag is. The bag options assume the following default values:

- MQCBO\_LIST\_FORM\_INHIBITED
- MQCBO\_DO\_NOT\_REORDER
- MQCBO\_DO\_NOT\_CHECK\_SELECTORS

Therefore, the MQAI does not change the order of data items or create lists within a message as with administration bags.

### **MQCBO\_GROUP\_BAG**

Specifies that the bag is a group bag. This means that the bag is used to hold a set of grouped items. Group bags cannot be used for the administration of IBM MQ objects. The bag options assume the following default values:

- MQCBO\_LIST\_FORM\_ALLOWED
- MQCBO\_REORDER\_AS\_REQUIRED
- MQCBO\_DO\_NOT\_CHECK\_SELECTORS

Therefore, the MQAI can change the order of data items or create lists within a bag of grouped items.

Group bags are created with two system selectors: MQIASY\_BAG\_OPTIONS and MQIASY\_CODED\_CHAR\_SET\_ID.

If a group bag is nested in a bag in which MQCBO\_CHECK\_SELECTORS was specified, the group bag to be nested has its selectors checked at that point whether MQCBO\_CHECK\_SELECTORS was specified when the group bag was created.

### **MQCBO\_USER\_BAG**

Specifies that the bag is a user bag. MQCBO\_USER\_BAG is the default bag-type option. User bags can also be used for the administration of IBM MQ objects, but the MQCBO\_LIST\_FORM\_ALLOWED and MQCBO\_REORDER\_AS\_REQUIRED options must be specified to ensure correct generation of the administration messages.

User bags are created with the MQIASY\_TYPE system item set to MQCFT\_USER.

For user bags, one or more of the following options can be specified:

#### **MQCBO\_LIST\_FORM\_ALLOWED**

Specifies that the MQAI can use the more compact list form in the message sent whenever there are two or more adjacent occurrences of the same selector in the bag. However, the items cannot be reordered if this option is used. Therefore, if the occurrences of the selector are not adjacent in the bag, and MQCBO\_REORDER\_AS\_REQUIRED is not specified, the MQAI cannot use the list form for that particular selector.

If the data items are character strings, these strings must have the same Character Set ID and the same selector, in order to be compacted into list form. If the list form is used, the shorter strings are padded with blanks to the length of the longest string.

This option must be specified if the message to be sent is an administration message but MQCBO\_ADMIN\_BAG is not specified.

**Note:** MQCBO\_LIST\_FORM\_ALLOWED does not imply that the MQAI definitely uses the list form. The MQAI considers various factors in deciding whether to use the list form.

**MQCBO\_LIST\_FORM\_INHIBITED**

Specifies that the MQAI cannot use the list form in the message sent, even if there are adjacent occurrences of the same selector in the bag. MQCBO\_LIST\_FORM\_INHIBITED is the default list-form option.

**MQCBO\_REORDER\_AS\_REQUIRED**

Specifies that the MQAI can change the order of the data items in the message sent. This option does not affect the order of the items in the sending bag.

This option means that you can insert items into a data bag in any order. That is, the items do not need to be inserted in the way that they must be in the PCF message, because the MQAI can reorder these items as required.

If the message is a user message, the order of the items in the receiving bag is the same as the order of the items in the message. This order can be different from the order of the items in the sending bag.

If the message is an administration message, the order of the items in the receiving bag is determined by the message received.

This option must be specified if the message to be sent is an administration message but MQCBO\_ADMIN is not specified.

**MQCBO\_DO\_NOT\_REORDER**

Specifies that the MQAI cannot change the order of data items in the message sent. Both the message sent and the receiving bag contain the items in the same order as they occur in the sending bag. This option is the default ordering option.

**MQCBO\_CHECK\_SELECTORS**

Specifies that user selectors (selectors that are zero or greater) must be checked to ensure that the selector is consistent with the data type implied by the mqAddInteger, mqAddInteger64, mqAddIntegerFilter, mqAddString, mqAddStringFilter, mqAddByteString, mqAddByteStringFilter, mqSetInteger, mqSetInteger64, mqSetIntegerFilter, mqSetString, mqSetStringFilter, mqSetByteString, or mqSetByteStringFilter call:

- For the integer, 64-bit integer, and integer filter calls, the selector must be in the range MQIA\_FIRST through MQIA\_LAST.
- For the string and string filter calls, the selector must be in the range MQCA\_FIRST through MQCA\_LAST.
- For byte string and byte string filter calls, the selector must be in the range MQBA\_FIRST through MQBA\_LAST.
- For group bag calls, the selector must be in the range MQGA\_FIRST through MQGA\_LAST.
- For the handle calls, the selector must be in the range MQHA\_FIRST through MQHA\_LAST.

The call fails if the selector is outside the valid range. System selectors (selectors less than zero) are always checked, and if a system selector is specified, it must be one that is supported by the MQAI.

**MQCBO\_DO\_NOT\_CHECK\_SELECTORS**

Specifies that user selectors (selectors that are zero or greater) are not checked. Any selector that is zero or positive can be used with any call. This option is the default selectors option. System selectors (selectors less than zero) are always checked.

**MQCBO\_NONE**

Specifies that all options must have their default values. This option is provided to aid program documentation, and must not be specified with any of the options that have a nonzero value.

The following list summarizes the default option values:

- MQCBO\_USER\_BAG
  - MQCBO\_LIST\_FORM\_INHIBITED
  - MQCBO\_DO\_NOT\_REORDER

– MQCBO\_DO\_NOT\_CHECK\_SELECTORS

### **Bag (MQHBAG) - output**

The handle of the bag created by the call.

### **CompCode (MQLONG) - output**

Completion code.

### **Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqCreateBag call:

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid (invalid parameter address or the parameter location is read-only).

#### **MQRC\_OPTIONS\_ERROR**

Options not valid or not consistent.

#### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

## **Usage notes for mqCreateBag**

Any options used for creating your bag are contained in a system item within the bag when it is created.

## **C language invocation for mqCreateBag**

```
mqCreateBag (Options, &Bag, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQLONG Options;          /* Bag options */
MQHBAG Bag;              /* Bag handle */
MQLONG CompCode;        /* Completion code */
MQLONG Reason;          /* Reason code qualifying CompCode */
```

## **Visual Basic invocation for mqCreateBag**

(Supported on Windows only.)

```
mqCreateBag Options, Bag, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Options As Long 'Bag options'
Dim Bag As Long 'Bag handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

## **Multi mqDeleteBag**

The mqDeleteBag call deletes the specified bag.

## Syntax for mqDeleteBag

**mqDeleteBag** (*Bag*, *CompCode*, *Reason*)

## Parameters for mqDeleteBag

### Bag (MQHBAG) - input/output

The handle of the bag to be deleted. This must be the handle of a bag created by the user, not the handle of a system bag. MQRC\_SYSTEM\_BAG\_NOT\_DELETABLE results if you specify the handle of a system bag. The handle is reset to MQHB\_UNUSABLE\_HBAG.

If the bag contains system-generated bags, they are also deleted.

### CompCode (MQLONG) - output

Completion code.

### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqDeleteBag call:

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid, or invalid parameter address, or parameter location is read only.

#### **MQRC\_SYSTEM\_BAG\_NOT\_DELETABLE**

System bag cannot be deleted.

## Usage notes for mqDeleteBag

1. Delete any bags created with mqCreateBag.
2. Nested bags are deleted automatically when the containing bag is deleted.

## C language invocation for mqDeleteBag

```
mqDeleteBag (&Bag, CompCode, Reason);
```

Declare the parameters as follows:

```
MQHBAG  Bag;           /* Bag handle */
MQLONG  CompCode;      /* Completion code */
MQLONG  Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqDeleteBag

(Supported on Windows only.)

```
mqDeleteBag Bag, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag;           As Long 'Bag handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

## Multi **mqDeleteItem**

The `mqDeleteItem` call removes one or more user items from a bag.

### Syntax for `mqDeleteItem`

`mqDeleteItem (Bag, Selector, ItemIndex, CompCode, Reason)`

### Parameters for `mqDeleteItem`

#### **Hbag (MQHBAG) - input**

Handle of the bag to be modified.

This must be the handle of a bag created by the user, and not the handle of a system bag; `MQRC_SYSTEM_BAG_NOT_ALTERABLE` results if it is a system bag.

#### **Selector (MQLONG) - input**

Selector identifying the user item to be deleted.

If the selector is less than zero (that is, a system selector), `MQRC_SELECTOR_OUT_OF_RANGE` results.

The following special values are valid:

##### **MQSEL\_ANY\_SELECTOR**

The item to be deleted is a user item identified by the **ItemIndex** parameter, the index relative to the set of items that contains both user and system items.

##### **MQSEL\_ANY\_USER\_SELECTOR**

The item to be deleted is a user item identified by the **ItemIndex** parameter, the index relative to the set of user items.

If an explicit selector value is specified, but the selector is not present in the bag, the call succeeds if `MQIND_ALL` is specified for **ItemIndex**, and fails with reason code `MQRC_SELECTOR_NOT_PRESENT` if `MQIND_ALL` is not specified.

#### **ItemIndex (MQLONG) - input**

Index of the data item to be deleted.

The value must be zero or greater, or one of the following special values:

##### **MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, `MQRC_SELECTOR_NOT_UNIQUE` results. If `MQIND_NONE` is specified with one of the `MQSEL_XXX_SELECTOR` values, `MQRC_INDEX_ERROR` results.

##### **MQIND\_ALL**

This specifies that all occurrences of the selector in the bag are to be deleted. If `MQIND_ALL` is specified with one of the `MQSEL_XXX_SELECTOR` values, `MQRC_INDEX_ERROR` results. If `MQIND_ALL` is specified when the selector is not present within the bag, the call succeeds.

If `MQSEL_ANY_SELECTOR` is specified for the **Selector** parameter, the **ItemIndex** parameter is the index relative to the set of items that contains both user items and system items, and must be zero or greater. If **ItemIndex** identifies a system selector `MQRC_SYSTEM_ITEM_NOT_DELETABLE` results. If `MQSEL_ANY_USER_SELECTOR` is specified for the **Selector** parameter, the **ItemIndex** parameter is the index relative to the set of user items, and must be zero or greater.

If an explicit selector value is specified, **ItemIndex** is the index relative to the set of items that have that selector value, and can be `MQIND_NONE`, `MQIND_ALL`, zero, or greater.

If an explicit index is specified (that is, not `MQIND_NONE` or `MQIND_ALL`) and the item is not present in the bag, `MQRC_INDEX_NOT_PRESENT` results.

### CompCode (MQLONG) - output

Completion code.

### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the `mqDeleteItem` call:

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

#### **MQRC\_INDEX\_ERROR**

MQIND\_NONE or MQIND\_ALL specified with one of the MQSEL\_ANY\_XXX\_SELECTOR values.

#### **MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag.

#### **MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

#### **MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

#### **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

#### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

#### **MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag is read only and cannot be altered.

#### **MQRC\_SYSTEM\_ITEM\_NOT\_DELETABLE**

System item is read only and cannot be deleted.

## Usage notes for `mqDeleteItem`

1. Either a single occurrence of the specified selector can be removed, or all occurrences of the specified selector.
2. The call cannot remove system items from the bag, or remove items from a system bag. However, the call can remove the handle of a system bag from a user bag. This way, a system bag can be deleted.

## C language invocation for `mqDeleteItem`

```
mqDeleteItem (Bag, Selector, ItemIndex, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG   Hbag;           /* Bag handle */
MQLONG   Selector;      /* Selector */
MQLONG   ItemIndex;     /* Index of the data item */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for `mqDeleteItem`

(Supported on Windows only.)

```
mqDeleteItem Bag, Selector, ItemIndex, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long 'Bag handle'
Dim Selector     As Long 'Selector'
Dim ItemIndex    As Long 'Index of the data item'
Dim CompCode    As Long 'Completion code'
Dim Reason       As Long 'Reason code qualifying CompCode'
```

## Multi **mqExecute**

The mqExecute call sends an administration command message and waits for the reply (if expected).

### Syntax for mqExecute

**mqExecute** (*Hconn, Command, OptionsBag, AdminBag, ResponseBag, AdminQ, ResponseQ, CompCode, Reason*)

### Parameters for mqExecute

#### **Hconn (MQHCONN) - input**

MQI Connection handle.

This is returned by a preceding MQCONN call issued by the application.

#### **Command (MQLONG) - input**

The command to be executed.

This should be one of the MQCMD\_\* values. If it is a value that is not recognized by the MQAI servicing the mqExecute call, the value is still accepted. However, if mqAddInquiry was used to insert values in the bag, the **Command** parameter must be an INQUIRE command recognized by the MQAI; MQRC\_INQUIRY\_COMMAND\_ERROR results if it is not.

#### **OptionsBag (MQHBAG) - input**

Handle of a bag containing options that affect the operation of the call.

This must be the handle returned by a preceding mqCreateBag call or the following special value:

##### **MQHB\_NONE**

No options bag; all options assume their default values.

Only the options listed in this topic can be present in the options bag (MQRC\_OPTIONS\_ERROR results if other data items are present).

The appropriate default value is used for each option that is not present in the bag. The following option can be specified:

##### **MQIACF\_WAIT\_INTERVAL**

This data item specifies the maximum time in milliseconds that the MQAI should wait for each reply message. The time interval must be zero or greater, or the special value MQWI\_UNLIMITED; the default is thirty seconds. The mqExecute call completes either when all of the reply messages are received or when the specified wait interval expires without the expected reply message having been received.

**Note:** The time interval is an approximate quantity.

If the MQIACF\_WAIT\_INTERVAL data item has the wrong data type, or there is more than one occurrence of that selector in the options bag, or the value of the data item is not valid, MQRC\_WAIT\_INTERVAL\_ERROR results.

#### **AdminBag (MQHBAG) - input**

Handle of the bag containing details of the administration command to be issued.



All user items placed in the bag are inserted into the administration message that is sent. It is the application's responsibility to ensure that only valid parameters for the command are placed in the bag.

If the value of the MQIASY\_TYPE data item in the command bag is not MQCFT\_COMMAND, MQRC\_COMMAND\_TYPE\_ERROR results. If the bag contains nested system bags, MQRC\_NESTED\_BAG\_NOT\_SUPPORTED results.

### **ResponseBag (MQHBAG) - input**

Handle of the bag where reply messages are placed.

The MQAI performs an mqClearBag call on the bag before placing reply messages in the bag. To retrieve the reply messages, the selector, MQIACF\_CONVERT\_RESPONSE, can be specified.

Each reply message is placed into a separate system bag, with a handle that is then placed in the response bag. Use the mqInquireBag call with selector MQHA\_BAG\_HANDLE to determine the handles of the system bags within the reply bag, and those bags can then be inquired to determine their contents.

If some but not all of the expected reply messages are received, MQCC\_WARNING with MQRC\_NO\_MSG\_AVAILABLE results. If none of the expected reply messages is received, MQCC\_FAILED with MQRC\_NO\_MSG\_AVAILABLE results.

Group bags cannot be used as response bags.

### **AdminQ (MQHOBJ) - input**

Object handle of the queue on which the administration message is to be placed.

This handle was returned by a preceding MQOPEN call issued by the application. The queue must be open for output.

The following special value can be specified:

#### **MQHO\_NONE**

This indicates that the administration message should be placed on the SYSTEM.ADMIN.COMMAND.QUEUE belonging to the currently connected queue manager. If MQHO\_NONE is specified, the application need not use MQOPEN to open the queue.

### **ResponseQ**

Object handle of the queue on which reply messages are placed.

This handle was returned by a preceding MQOPEN call issued by the application. The queue must be open for input and for inquiry.

The following special value can be specified:

#### **MQHO\_NONE**

This indicates that the reply messages should be placed on a dynamic queue created automatically by the MQAI. The queue is created by opening SYSTEM.DEFAULT.MODEL.QUEUE, that must therefore have suitable characteristics. The queue created exists for the duration of the call only, and is deleted by the MQAI on exit from the mqExecute call.

### **CompCode**

Completion code.

### **Reason**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqExecute call:

#### **MQRC\_\***

Anything from the MQINQ, MQPUT, MQGET, or MQOPEN calls.

**MQRC\_BAG\_WRONG\_TYPE**

Input data bag is a group bag.

**MQRC\_CMD\_SERVER\_NOT\_AVAILABLE**

The command server that processes administration commands is not available.

**MQRC\_COMMAND\_TYPE\_ERROR**

The value of the MQIASY\_TYPE data item in the request bag is not MQCFT\_COMMAND.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INQUIRY\_COMMAND\_ERROR**

mqAddInteger call used with a command code that is not a recognized INQUIRE command.

**MQRC\_NESTED\_BAG\_NOT\_SUPPORTED**

Input data bag contains one or more nested system bags.

**MQRC\_NO\_MSG\_AVAILABLE**

Some reply messages received, but not all. Reply bag contains system-generated bags for messages that were received.

**MQRC\_NO\_MSG\_AVAILABLE**

No reply messages received during the specified wait interval.

**MQRC\_OPTIONS\_ERROR**

Options bag contains unsupported data items, or a supported option has a value which is not valid.

**MQRC\_PARAMETER\_MISSING**

Administration message requires a parameter which is not present in the bag. This reason code occurs for bags created with the MQCBO\_ADMIN\_BAG or MQCBO\_REORDER\_AS\_REQUIRED options only.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

Two or more instances of a selector exist within the bag for a mandatory parameter that permits one instance only.

**MQRC\_SELECTOR\_WRONG\_TYPE**

mqAddString or mqSetString was used to add the MQIACF\_INQUIRY selector to the bag.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRCCF\_COMMAND\_FAILED**

Command failed; details of failure are contained in system-generated bags within the reply bag.

## Usage notes for mqExecute

1. If no *AdminQ* is specified, the MQAI checks to see if the command server is active before sending the administration command message. However, if the command server is not active, the MQAI does not start it. If you are sending many administration command messages, you are recommended to open the SYSTEM.ADMIN.COMMAND.QUEUE yourself and pass the handle of the administration queue on each administration request.
2. Specifying the MQHO\_NONE value in the **ResponseQ** parameter simplifies the use of the mqExecute call, but if mqExecute is issued repeatedly by the application (for example, from within a loop), the response queue will be created and deleted repeatedly. In this situation, it is better for the application itself to open the response queue before any mqExecute call, and close it after all mqExecute calls have been issued.
3. If the administration command results in a message being sent with a message type of MQMT\_REQUEST, the call waits for the time given by the MQIACF\_WAIT\_INTERVAL data item in the options bag.
4. If an error occurs during the processing of the call, the response bag might contain some data from the reply message, but the data will typically be incomplete.

## C language invocation for mqExecute

```
mqExecute (Hconn, Command, OptionsBag, AdminBag, ResponseBag,  
AdminQ, ResponseQ, CompCode, Reason);
```

Declare the parameters as follows:

```
MQHCONN  Hconn;           /* MQI connection handle */  
MQLONG   Command;        /* Command to be executed */  
MQHBAG   OptionsBag;     /* Handle of a bag containing options */  
MQHBAG   AdminBag;       /* Handle of administration bag containing  
                          /* details of administration command */  
MQHBAG   ResponseBag;    /* Handle of bag for response messages */  
MQHOBJ   AdminQ;         /* Handle of administration queue for  
                          /* administration messages */  
MQHOBJ   ResponseQ;      /* Handle of response queue for response  
                          /* messages */  
MQLONG   pCompCode;      /* Completion code */  
MQLONG   pReason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqExecute

(Supported on Windows only.)

```
mqExecute (Hconn, Command, OptionsBag, AdminBag, ResponseBag,  
AdminQ, ResponseQ, CompCode, Reason);
```

Declare the parameters as follows:

```
Dim HConn      As Long 'MQI connection handle'  
Dim Command    As Long 'Command to be executed'  
Dim OptionsBag As Long 'Handle of a bag containing options'  
Dim AdminBag   As Long 'Handle of command bag containing details of  
                        administration command'  
Dim ResponseBag As Long 'Handle of bag for reply messages'  
Dim AdminQ     As Long 'Handle of command queue for  
                        administration messages'  
Dim ResponseQ  As Long 'Handle of response queue for reply messages'  
Dim CompCode   As Long 'Completion code'  
Dim Reason     As Long 'Reason code qualifying CompCode'
```

### **Example code for using the mqExecute call**

Two code examples showing how to use mqExecute to create a local queue and to inquire about queue attributes.

#### **Example: Using mqExecute to create a local queue**

The following example creates a local queue, with a maximum message length of 100 bytes, on a queue manager:

```
/* Create a bag for the data you want in your PCF message */  
mqCreateBag(MQCBO_ADMIN_BAG, &hbagRequest)  
  
/* Create a bag to be filled with the response from the command server */  
mqCreateBag(MQCBO_ADMIN_BAG, &hbagResponse)  
  
/* Create a queue */  
/* Supply queue name */  
mqAddString(hbagRequest, MQCA_Q_NAME, "QBERT")  
  
/* Supply queue type */  
mqAddString(hbagRequest, MQIA_Q_TYPE, MQQT_LOCAL)  
  
/* Maximum message length is an optional parameter */  
mqAddString(hbagRequest, MQIA_MAX_MSG_LENGTH, 100)
```

```

/* Ask the command server to create the queue */
mqExecute(MQCMD_CREATE_Q, hbagRequest, hbagResponse)

/* Tidy up memory allocated */
mqDeleteBag(hbagRequest)
mqDeleteBag(hbagResponse)

```

## Example: Using mqExecute to inquire about queue attributes

The following example inquires about all attributes of a particular queue. The mqAddInquiry call identifies all IBM MQ object attributes of a queue to be returned by the Inquire parameter on mqExecute:

```

/* Create a bag for the data you want in your PCF message */
mqCreateBag(MQCBO_ADMIN_BAG, &hbagRequest)

/* Create a bag to be filled with the response from the command server */
mqCreateBag(MQCBO_ADMIN_BAG, &hbagResponse)

/* Inquire about a queue by supplying its name */
/* (other parameters are optional) */
mqAddString(hbagRequest, MQCA_Q_NAME, "QBERT")

/* Request the command server to inquire about the queue */
mqExecute(MQCMD_INQUIRE_Q, hbagRequest, hbagResponse)

/* If it worked, the attributes of the queue are returned */
/* in a system bag within the response bag */
mqInquireBag(hbagResponse, MQHA_BAG_HANDLE, 0, &hbagAttributes)

/* Inquire the name of the queue and its current depth */
mqInquireString(hbagAttributes, MQCA_Q_NAME, &stringAttribute)
mqInquireString(hbagAttributes, MQIA_CURRENT_Q_DEPTH, &integerAttribute)

/* Tidy up memory allocated */
mqDeleteBag(hbagRequest)
mqDeleteBag(hbagResponse)

```

Using mqExecute is the simplest way of administering IBM MQ, but lower-level calls, [mqBagToBuffer](#) and [mqBufferToBag](#), can be used. For more information about the use of these calls, see [Using the MQAI to simplify the use of PCFs](#).

## mqGetBag

The mqGetBag call removes a message from the specified queue and converts the message data into a data bag.

### Syntax for mqGetBag

**mqGetBag (Hconn, Hobj, MsgDesc, GetMsgOpts, HBag, CompCode, Reason)**

### Parameters for mqGetBag

#### Hconn (MQHCONN) - input

MQI connection handle.

#### Hobj (MQHOBJ) - input

Object handle of the queue from which the message is to be retrieved. This handle was returned by a preceding MQOPEN call issued by the application. The queue must be open for input.

#### MsgDesc (MQMD) - input/output

Message descriptor (for more information, see [MQMD - Message descriptor](#)).

If the *Format* field in the message has a value other than MQFMT\_ADMIN, MQFMT\_EVENT, or MQFMT\_PCF, MQRC\_FORMAT\_NOT\_SUPPORTED results.

If, on entry to the call, the *Encoding* field in the application's MQMD has a value other than MQENC\_NATIVE and MQGMO\_CONVERT is specified, MQRC\_ENCODING\_NOT\_SUPPORTED results. Also, if MQGMO\_CONVERT is not specified, the value of the **Encoding** parameter must be the retrieving application's MQENC\_NATIVE; if not, again MQRC\_ENCODING\_NOT\_SUPPORTED results.

#### **GetMsgOpts (MQGMO) - input/output**

Get-message options (for more information, see [MQGMO - Get-message options](#) ).

MQGMO\_ACCEPT\_TRUNCATED\_MSG cannot be specified; MQRC\_OPTIONS\_ERROR results if it is. MQGMO\_LOCK and MQGMO\_UNLOCK are not supported in a 16-bit or 32-bit Window environment. MQGMO\_SET\_SIGNAL is supported in a 32-bit Window environment only.

#### **HBag (MQHBAG) - input/output**

Handle of a bag into which the retrieved message is placed. The MQAI performs an mqClearBag call on the bag before placing the message in the bag.

#### **MQHB\_NONE**

Gets the retrieved message. This provides a means of deleting messages from the queue.

If an option of MQGMO\_BROWSE\_\* is specified, this value sets the browse cursor to the selected message; it is not deleted in this case.

#### **CompCode (MQLONG) - output**

Completion code.

#### **Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating warning and error conditions can be returned from the mqGetBag call:

#### **MQRC\_\***

Anything from the MQGET call or bag manipulation.

#### **MQRC\_BAG\_CONVERSION\_ERROR**

Data could not be converted into a bag.

This indicates a problem with the format of the data to be converted into a bag (for example, the message is not a valid PCF).

If the message was retrieved destructively from the queue (that is, not browsing the queue), this reason code indicates that it has been discarded.

#### **MQRC\_BAG\_WRONG\_TYPE**

Input data bag is a group bag.

#### **MQRC\_ENCODING\_NOT\_SUPPORTED**

Encoding not supported; the value in the *Encoding* field of the MQMD must be MQENC\_NATIVE.

#### **MQRC\_FORMAT\_NOT\_SUPPORTED**

Format not supported; the *Format* name in the message is not MQFMT\_ADMIN, MQFMT\_EVENT, or MQFMT\_PCF. If the message was retrieved destructively from the queue (that is, not browsing the queue), this reason code indicates that it has been discarded.

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

#### **MQRC\_INCONSISTENT\_ITEM\_TYPE**

Data type of second occurrence of selector differs from data type of first occurrence.

#### **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

#### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

## **MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

### **Usage notes for mqGetBag**

1. Only messages that have a supported format can be returned by this call. If the message has a format that is not supported, the message is discarded, and the call completes with an appropriate reason code.
2. If the message is retrieved within a unit of work (that is, with the MQGMO\_SYNCPOINT option), and the message has an unsupported format, the unit of work can be backed out, reinstating the message on the queue. This allows the message to be retrieved by using the MQGET call in place of the mqGetBag call.

### **C language invocation for mqGetBag**

```
mqGetBag (hConn, hObj, &MsgDesc, &GetMsgOpts, hBag, CompCode, Reason);
```

Declare the parameters as follows:

```
MQHCONN  hConn;          /* MQI connection handle */
MQHOBJ   hObj;           /* Object handle */
MQMD     MsgDesc;        /* Message descriptor */
MQGMO    GetMsgOpts;     /* Get-message options */
MQHBAG   hBag;           /* Bag handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

### **Visual Basic invocation for mqGetBag**

(Supported on Windows only.)

```
mqGetBag (HConn, HObj, MsgDesc, GetMsgOpts, Bag, CompCode, Reason);
```

Declare the parameters as follows:

```
Dim HConn      As Long 'MQI connection handle'
Dim HObj       As Long 'Object handle'
Dim MsgDesc    As Long 'Message descriptor'
Dim GetMsgOpts As Long 'Get-message options'
Dim Bag        As Long 'Bag handle'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'
```

### **Multi mqInquireBag**

The mqInquireBag call inquires the value of a bag handle that is present in the bag. The data item can be a user item or a system item.

### **Syntax for mqInquireBag**

**mqInquireBag (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason)**

### **Parameters for mqInquireBag**

#### **Bag (MQHBAG) - input**

Bag handle to be inquired. The bag can be a user bag or a system bag.

**Selector (MQLONG) - input**

Selector identifying the item to be inquired.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The data type of the item must agree with the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

The following special values can be specified for **Selector**:

**MQSEL\_ANY\_SELECTOR**

The item to be inquired is a user or system item identified by the **ItemIndex** parameter.

**MQSEL\_ANY\_USER\_SELECTOR**

The item to be inquired is a user item identified by the **ItemIndex** parameter.

**MQSEL\_ANY\_SYSTEM\_SELECTOR**

The item to be inquired is a system item identified by the **ItemIndex** parameter.

**ItemIndex (MQLONG) - input**

Index of the data item to be inquired.

The value must be zero or greater, or the special value MQIND\_NONE. If the value is less than zero and not MQIND\_NONE, MQRC\_INDEX\_ERROR results. If the item is not already present in the bag, MQRC\_INDEX\_NOT\_PRESENT results.

The following special value can be specified:

**MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for the **Selector** parameter, the **ItemIndex** parameter is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for the **Selector** parameter, the **ItemIndex** parameter is the index relative to the set of system items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for the **Selector** parameter, the **ItemIndex** parameter is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, the **ItemIndex** parameter is the index relative to the set of items that have that selector value and can be MQIND\_NONE, zero, or greater.

**ItemValue (MQHBAG) - output**

Value of the item in the bag.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireBag call:

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE, or MQIND\_NONE specified with one of the MQSEL\_ANY\_XXX\_SELECTOR values).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_ITEM\_VALUE\_ERROR**

The **ItemValue** parameter is not valid (invalid parameter address).

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present within the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

## C language invocation for mqInquireBag

```
mqInquireBag (Bag, Selector, ItemIndex, &ItemValue, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG  Bag;           /* Bag handle */
MQLONG  Selector;      /* Selector */
MQLONG  ItemIndex;     /* Index of the data item to be inquired */
MQHBAG  ItemValue;     /* Value of item in the bag */
MQLONG  CompCode;      /* Completion code */
MQLONG  Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqInquireBag

(Supported on Windows only.)

```
mqInquireBag (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Index of the data item to be inquired'
Dim ItemValue As Long 'Value of item in the bag'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

## mqInquireByteString

The mqInquireByteString call requests the value of a byte string data item that is present in the bag. The data item can be a user item or a system item.



## Syntax for mqInquireByteString

**mqInquireByteString** (*Bag, Selector, ItemIndex, Bufferlength, Buffer, ByteStringLength, CompCode, Reason*)

## Parameters for mqInquireByteString

### Bag (MQHBAG) - input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

### Selector (MQLONG) - input

Selector of the item to which the inquiry relates.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The data type of the item must be the same as the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

The following special values can be specified for *Selector*:

#### **MQSEL\_ANY\_SELECTOR**

The item to be inquired about is a user or system item identified by *ItemIndex*.

#### **MQSEL\_ANY\_USER\_SELECTOR**

The item to be inquired about is a user item identified by *ItemIndex*.

#### **MQSEL\_ANY\_SYSTEM\_SELECTOR**

The item to be inquired about is a system item identified by *ItemIndex*.

### ItemIndex (MQLONG) - input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND\_NONE. If the value is less than zero and not MQIND\_NONE, MQRC\_INDEX\_ERROR results. If the item is not already present in the bag, MQRC\_INDEX\_NOT\_PRESENT results. The following special value can be specified:

#### **MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for the **Selector** parameter, **ItemIndex** is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for the **Selector** parameter, **ItemIndex** is the index relative to the set of user items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for **Selector**, **ItemIndex** is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, **ItemIndex** is the index relative to the set of items that have that selector value, and can be MQIND\_NONE, zero, or greater.

### BufferLength (MQLONG) - input

Length in bytes of the buffer to receive the byte string. Zero is a valid value.

**Buffer (MQBYTE x BufferLength) - output**

Buffer to receive the byte string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (non-null) address must be specified for the **Buffer** parameter.

The string is padded with nulls to the length of the buffer. If the string is longer than the buffer, the string is truncated to fit; in this case *ByteStringLength* indicates the size of the buffer needed to accommodate the string without truncation.

**ByteStringLength (MQLONG) - output**

The length in bytes of the string contained in the bag. If the **Buffer** parameter is too small, the length of the string returned is less than *ByteStringLength*.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the *mqInquireByteString* call:

**MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE, or MQIND\_NONE specified with one of the MQSEL\_ANY\_XXX\_SELECTOR values).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_STRING\_LENGTH\_ERROR**

*ByteStringLength* parameter not valid (invalid parameter address).

**MQRC\_STRING\_TRUNCATED**

Data too long for output buffer and has been truncated.

## C language invocation for mqInquireByteString

```
mqInquireByteString (Bag, Selector, ItemIndex,  
BufferLength, Buffer, &StringLength, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */  
MQLONG   Selector;      /* Selector */  
MQLONG   ItemIndex;     /* Item index */  
MQLONG   BufferLength;  /* Buffer length */  
PMQBYTE  Buffer;        /* Buffer to contain string */  
MQLONG   ByteStringLength; /* Length of byte string returned */  
MQLONG   CompCode;     /* Completion code */  
MQLONG   Reason;       /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqInquireByteString

(Supported on Windows only.)

```
mqInquireByteString Bag, Selector, ItemIndex,  
BufferLength, Buffer, StringLength, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long   'Bag handle'  
Dim Selector      As Long   'Selector'  
Dim ItemIndex     As Long   'Item index'  
Dim BufferLength   As Long   'Buffer length'  
Dim Buffer         As Byte   'Buffer to contain string'  
Dim ByteStringLength As Long 'Length of byte string returned'  
Dim CompCode      As Long   'Completion code'  
Dim Reason        As Long   'Reason code qualifying CompCode'
```

### mqInquireByteStringFilter

The mqInquireByteStringFilter call requests the value and operator of a byte string filter item that is present in the bag. The data item can be a user item or a system item.

## Syntax for mqInquireByteStringFilter

```
mqInquireByteStringFilter (Bag, Selector, ItemIndex, Bufferlength, Buffer,  
ByteStringLength, Operator, CompCode, Reason)
```

## Parameters for mqInquireByteStringFilter

### Bag (MQHBAG) - input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

### Selector (MQLONG) - input

Selector of the item to which the inquiry relates.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The data type of the item must be the same as the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

The following special values can be specified for *Selector*:

**MQSEL\_ANY\_SELECTOR**

The item to be inquired about is a user or system item identified by *ItemIndex*.

**MQSEL\_ANY\_USER\_SELECTOR**

The item to be inquired about is a user item identified by *ItemIndex*.

**MQSEL\_ANY\_SYSTEM\_SELECTOR**

The item to be inquired about is a system item identified by *ItemIndex*.

**ItemIndex (MQLONG) - input**

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND\_NONE. If the value is less than zero and not MQIND\_NONE, MQRC\_INDEX\_ERROR results. If the item is not already present in the bag, MQRC\_INDEX\_NOT\_PRESENT results. The following special value can be specified:

**MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for the **Selector** parameter, **ItemIndex** is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for the **Selector** parameter, **ItemIndex** is the index relative to the set of user items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for **Selector**, **ItemIndex** is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, **ItemIndex** is the index relative to the set of items that have that selector value, and can be MQIND\_NONE, zero, or greater.

**BufferLength (MQLONG) - input**

Length in bytes of the buffer to receive the condition byte string. Zero is a valid value.

**Buffer (MQBYTE x BufferLength) - output**

Buffer to receive the condition byte string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (non-null) address must be specified for the **Buffer** parameter.

The string is padded with blanks to the length of the buffer; the string is not null-terminated. If the string is longer than the buffer, the string is truncated to fit; in this case **ByteStringLength** indicates the size of the buffer needed to accommodate the string without truncation.

**ByteStringLength (MQLONG) - output**

The length in bytes of the condition string contained in the bag. If the **Buffer** parameter is too small, the length of the string returned is less than **StringLength**.

**Operator (MQLONG) - output**

Byte string filter operator in the bag.

**CompCode (MQLONG) - output**

Completion code.

## Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the `mqInquireByteStringFilter` call:

### **MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

### **MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

### **MQRC\_FILTER\_OPERATOR\_ERROR**

Filter operator not valid.

### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

### **MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE, or MQIND\_NONE specified with one of the MQSEL\_ANY\_xxx\_SELECTOR values).

### **MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

### **MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

### **MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

### **MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

### **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

### **MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

### **MQRC\_STRING\_LENGTH\_ERROR**

`ByteStringLength` parameter not valid (invalid parameter address).

### **MQRC\_STRING\_TRUNCATED**

Data too long for output buffer and has been truncated.

## C language invocation for `mqInquireByteStringFilter`

```
mqInquireByteStringFilter (Bag, Selector, ItemIndex,  
BufferLength, Buffer, &ByteStringLength, &Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */  
MQLONG   Selector;     /* Selector */  
MQLONG   ItemIndex;    /* Item index */  
MQLONG   BufferLength;  /* Buffer length */  
PMQBYTE  Buffer;        /* Buffer to contain string */  
MQLONG   ByteStringLength; /* Length of string returned */  
MQLONG   Operator;     /* Item operator */  
PMQLONG  CompCode;     /* Completion code */  
PMQLONG  Reason;       /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqInquireByteStringFilter

(Supported on Windows only.)

```
mqInquireByteStringFilter Bag, Selector, ItemIndex,  
BufferLength, Buffer, ByteStringLength,  
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long   'Bag handle'  
Dim Selector      As Long   'Selector'  
Dim ItemIndex     As Long   'Item index'  
Dim BufferLength  As Long   'Buffer length'  
Dim Buffer         As String 'Buffer to contain string'  
Dim ByteStringLength As Long 'Length of byte string returned'  
Dim Operator      As Long   'Operator'  
Dim CompCode     As Long   'Completion code'  
Dim Reason       As Long   'Reason code qualifying CompCode'
```

### mqInquireInteger

The mqInquireInteger call requests the value of an integer data item that is present in the bag. The data item can be a user item or a system item.

### Syntax for mqInquireInteger

**mqInquireInteger** (*Bag, Selector, ItemIndex, ItemValue, CompCode, Reason*)

### Parameters for mqInquireInteger

#### Bag (MQHBAG) - input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

#### Selector (MQLONG) - input

Selector identifying the item to which the inquiry relates.

If the selector is less than zero (a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The data type of the item must agree with the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

The following special values can be specified for *Selector*:

#### **MQSEL\_ANY\_SELECTOR**

The item to be inquired about is a user or system item identified by *ItemIndex*.

#### **MQSEL\_ANY\_USER\_SELECTOR**

The item to be inquired about is a user item identified by *ItemIndex*.

#### **MQSEL\_ANY\_SYSTEM\_SELECTOR**

The item to be inquired about is a system item identified by *ItemIndex*.

#### ItemIndex (MQLONG) - input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND\_NONE. If the value is less than zero and is not MQIND\_NONE, MQRC\_INDEX\_ERROR results. If the item is not already present in the bag, MQRC\_INDEX\_NOT\_PRESENT results. The following special value can be specified:

**MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND\_NONE, zero, or greater.

**ItemValue (MQLONG) - output**

The value of the item in the bag.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireInteger call:

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE, or MQIND\_NONE specified with one of the MQSEL\_ANY\_xxx\_SELECTOR values).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_ITEM\_VALUE\_ERROR**

*ItemValue* parameter not valid (invalid parameter address).

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**C language invocation for mqInquireInteger**

```
mqInquireInteger (Bag, Selector, ItemIndex, &ItemValue,
&CompCode, &Reason);
```

Declare the parameters as follows:

```

MQHBAG  Bag;           /* Bag handle */
MQLONG  Selector;     /* Selector */
MQLONG  ItemIndex;    /* Item index */
MQLONG  ItemValue;    /* Item value */
MQLONG  CompCode;     /* Completion code */
MQLONG  Reason;       /* Reason code qualifying CompCode */

```

## Visual Basic invocation for mqInquireInteger

(Supported on Windows only.)

```

mqInquireInteger Bag, Selector, ItemIndex, ItemValue,
CompCode, Reason

```

Declare the parameters as follows:

```

Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim ItemValue As Long 'Item value'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'

```

### mqInquireInteger64

The mqInquireInteger64 call requests the value of a 64-bit integer data item that is present in the bag. The data item can be a user item or a system item.

## Syntax for mqInquireInteger64

**mqInquireInteger64** (*Bag*, *Selector*, *ItemIndex*, *ItemValue*, *CompCode*, *Reason*)

## Parameters for mqInquireInteger64

### Bag (MQHBAG) - input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

### Selector (MQLONG) - input

Selector identifying the item to which the inquiry relates.

If the selector is less than zero (a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The data type of the item must agree with the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

The following special values can be specified for *Selector*:

#### **MQSEL\_ANY\_SELECTOR**

The item to be inquired about is a user or system item identified by *ItemIndex*.

#### **MQSEL\_ANY\_USER\_SELECTOR**

The item to be inquired about is a user item identified by *ItemIndex*.

#### **MQSEL\_ANY\_SYSTEM\_SELECTOR**

The item to be inquired about is a system item identified by *ItemIndex*.



**ItemIndex (MQLONG) - input**

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND\_NONE. If the value is less than zero and is not MQIND\_NONE, MQRC\_INDEX\_ERROR results. If the item is not already present in the bag, MQRC\_INDEX\_NOT\_PRESENT results. The following special value can be specified:

**MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND\_NONE, zero, or greater.

**ItemValue (MQINT64) - output**

The value of the item in the bag.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireInteger64 call:

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE, or MQIND\_NONE specified with one of the MQSEL\_ANY\_XXX\_SELECTOR values).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_ITEM\_VALUE\_ERROR**

*ItemValue* parameter not valid (invalid parameter address).

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

## C language invocation for mqInquireInteger64

```
mqInquireInteger64 (Bag, Selector, ItemIndex, &ItemValue,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */  
MQLONG   Selector;      /* Selector */  
MQLONG   ItemIndex;     /* Item index */  
MQINT64  ItemValue;     /* Item value */  
MQLONG   CompCode;      /* Completion code */  
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqInquireInteger64

(Supported on Windows only.)

```
mqInquireInteger64 Bag, Selector, ItemIndex, ItemValue,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim ItemIndex As Long 'Item index'  
Dim ItemValue As Long 'Item value'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

## mqInquireIntegerFilter

The mqInquireIntegerFilter call requests the value and operator of an integer filter item that is present in the bag. The data item can be a user item or a system item.

### Syntax for mqInquireIntegerFilter

```
mqInquireIntegerFilter (Bag, Selector, ItemIndex, ItemValue, Operator, CompCode,  
Reason)
```

### Parameters for mqInquireIntegerFilter

#### Bag (MQHBAG) - input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

#### Selector (MQLONG) - input

Selector identifying the item to which the inquiry relates.

If the selector is less than zero (a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The data type of the item must agree with the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

The following special values can be specified for *Selector*:

**MQSEL\_ANY\_SELECTOR**

The item to be inquired about is a user or system item identified by *ItemIndex*.

**MQSEL\_ANY\_USER\_SELECTOR**

The item to be inquired about is a user item identified by *ItemIndex*.

**MQSEL\_ANY\_SYSTEM\_SELECTOR**

The item to be inquired about is a system item identified by *ItemIndex*.

**ItemIndex (MQLONG) - input**

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND\_NONE. If the value is less than zero and is not MQIND\_NONE, MQRC\_INDEX\_ERROR results. If the item is not already present in the bag, MQRC\_INDEX\_NOT\_PRESENT results. The following special value can be specified:

**MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND\_NONE, zero, or greater.

**ItemValue (MQLONG) - output**

The condition value.

**Operator (MQLONG) - output**

Integer filter operator in the bag.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireIntegerFilter call:

**MQRC\_FILTER\_OPERATOR\_ERROR**

Filter operator not valid.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE, or MQIND\_NONE specified with one of the MQSEL\_ANY\_XXX\_SELECTOR values).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_ITEM\_VALUE\_ERROR**

*ItemValue* parameter not valid (invalid parameter address).

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**C language invocation for mqInquireIntegerFilter**

```
mqInquireIntegerFilter (Bag, Selector, ItemIndex, &ItemValue,
&Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */
MQLONG   Selector;      /* Selector */
MQLONG   ItemIndex;     /* Item index */
MQLONG   ItemValue;     /* Item value */
MQLONG   Operator;      /* Item operator */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqInquireIntegerFilter**

(Supported on Windows only.)

```
mqInquireIntegerFilter Bag, Selector, ItemIndex, ItemValue,
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim ItemValue As Long 'Item value'
Dim Operator As Long 'Item operator'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

**Multi mqInquireItemInfo**

The mqInquireItemInfo call returns information about a specified item in a bag. The data item can be a user item or a system item.

**Syntax for mqInquireItemInfo**

```
mqInquireItemInfo (Bag, Selector, ItemIndex, ItemType, OutSelector, CompCode,
Reason)
```

## Parameters for mqInquireItemInfo

### Bag (MQHBAG) - input

Handle of the bag to be inquired.

The bag can be a user bag or a system bag.

### Selector (MQLONG) - input

Selector identifying the item to be inquired.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The following special values can be specified for **Selector**:

#### MQSEL\_ANY\_SELECTOR

The item to be inquired is a user or system item identified by the **ItemIndex** parameter.

#### MQSEL\_ANY\_USER\_SELECTOR

The item to be inquired is a user item identified by the **ItemIndex** parameter.

#### MQSEL\_ANY\_SYSTEM\_SELECTOR

The item to be inquired is a system item identified by the **ItemIndex** parameter.

### ItemIndex (MQLONG) - input

Index of the data item to be inquired.

The item must be present within the bag; MQRC\_INDEX\_NOT\_PRESENT results if it is not. The value must be zero or greater, or the following special value:

#### MQIND\_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for the **Selector** parameter, the **ItemIndex** parameter is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for the **Selector** parameter, the **ItemIndex** parameter is the index relative to the set of system items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for the **Selector** parameter, the **ItemIndex** parameter is the index relative to the set of system items, and must be zero or greater. If an explicit selector value is specified, the **ItemIndex** parameter is the index relative to the set of items that have that selector value and can be MQIND\_NONE, zero, or greater.

### ItemType (MQLONG) - output

The data type of the specified data item.

The following can be returned:

#### MQITEM\_BAG

Bag handle item.

#### MQITEM\_BYTE\_STRING

Byte string.

#### MQITEM\_INTEGER

Integer item.

#### MQITEM\_INTEGER\_FILTER

Integer filter.

**MQITEM\_INTEGER64**

64-bit integer item.

**MQITEM\_STRING**

Character-string item.

**MQITEM\_STRING\_FILTER**

String filter.

**OutSelector (MQLONG) - output**

Selector of the specified data item.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the `mqInquireItemInfo` call:

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

MQIND\_NONE specified with one of the MQSEL\_ANY\_XXX\_SELECTOR values.

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_ITEM\_TYPE\_ERROR****ItemType** parameter not valid (invalid parameter address).**MQRC\_OUT\_SELECTOR\_ERROR****OutSelector** parameter not valid (invalid parameter address).**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**C language invocation for mqInquireItemInfo**

```
mqInquireItemInfo (Bag, Selector, ItemIndex, &OutSelector, &ItemType,
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */
MQLONG   Selector;      /* Selector identifying item */
MQLONG   ItemIndex;     /* Index of data item */
MQLONG   OutSelector;   /* Selector of specified data item */
MQLONG   ItemType;     /* Data type of data item */
```

```
MQLONG  CompCode;      /* Completion code */
MQLONG  Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqInquireItemInfo

(Supported on Windows only.)

```
mqInquireItemInfo Bag, Selector, ItemIndex, OutSelector, ItemType,
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long 'Bag handle'
Dim Selector      As Long 'Selector identifying item'
Dim ItemIndex     As Long 'Index of data item'
Dim OutSelector   As Long 'Selector of specified data item'
Dim ItemType      As Long 'Data type of data item'
Dim CompCode      As Long 'Completion code'
Dim Reason        As Long 'Reason code qualifying CompCode'
```

### Multi mqInquireString

The mqInquireString call requests the value of a character data item that is present in the bag. The data item can be a user item or a system item.

## Syntax for mqInquireString

**mqInquireString** (*Bag, Selector, ItemIndex, Bufferlength, Buffer, StringLength, CodedCharSetId, CompCode, Reason*)

## Parameters for mqInquireString

### Bag (MQHBAG) - input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

### Selector (MQLONG) - input

Selector of the item to which the inquiry relates.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The data type of the item must be the same as the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

The following special values can be specified for *Selector*:

#### **MQSEL\_ANY\_SELECTOR**

The item to be inquired about is a user or system item identified by *ItemIndex*.

#### **MQSEL\_ANY\_USER\_SELECTOR**

The item to be inquired about is a user item identified by *ItemIndex*.

#### **MQSEL\_ANY\_SYSTEM\_SELECTOR**

The item to be inquired about is a system item identified by *ItemIndex*.

**ItemIndex (MQLONG) - input**

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND\_NONE. If the value is less than zero and not MQIND\_NONE, MQRC\_INDEX\_ERROR results. If the item is not already present in the bag, MQRC\_INDEX\_NOT\_PRESENT results. The following special value can be specified:

**MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for the **Selector** parameter, **ItemIndex** is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for the **Selector** parameter, **ItemIndex** is the index relative to the set of user items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for **Selector**, **ItemIndex** is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, **ItemIndex** is the index relative to the set of items that have that selector value, and can be MQIND\_NONE, zero, or greater.

**BufferLength (MQLONG) - input**

Length in bytes of the buffer to receive the string. Zero is a valid value.

**Buffer (MQCHAR x BufferLength) - output**

Buffer to receive the character string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (non-null) address must be specified for the **Buffer** parameter.

The string is padded with blanks to the length of the buffer; the string is not null-terminated. If the string is longer than the buffer, the string is truncated to fit; in this case **StringLength** indicates the size of the buffer needed to accommodate the string without truncation.

**StringLength (MQLONG) - output**

The length in bytes of the string contained in the bag. If the **Buffer** parameter is too small, the length of the string returned is less than *StringLength*.

**CodedCharSetId (MQLONG) - output**

The coded character set identifier for the character data in the string. This parameter can be set to a null pointer if not required.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqInquireString call:

**MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.



**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE, or MQIND\_NONE specified with one of the MQSEL\_ANY\_XXX\_SELECTOR values).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_STRING\_LENGTH\_ERROR**

**StringLength** parameter not valid (invalid parameter address).

**MQRC\_STRING\_TRUNCATED**

Data too long for output buffer and has been truncated.

**C language invocation for mqInquireString**

```
mqInquireString (Bag, Selector, ItemIndex,
BufferLength, Buffer, &StringLength, &CodedCharSetId,
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    Bag;           /* Bag handle */
MQLONG    Selector;      /* Selector */
MQLONG    ItemIndex;     /* Item index */
MQLONG    BufferLength;   /* Buffer length */
PMQCHAR   Buffer;        /* Buffer to contain string */
MQLONG    StringLength;  /* Length of string returned */
MQLONG    CodedCharSetId /* Coded Character Set ID */
MQLONG    CompCode;     /* Completion code */
MQLONG    Reason;       /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqInquireString**

(Supported on Windows only.)

```
mqInquireString Bag, Selector, ItemIndex,
BufferLength, Buffer, StringLength, CodedCharSetId,
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long   'Bag handle'
Dim Selector      As Long   'Selector'
Dim ItemIndex     As Long   'Item index'
Dim BufferLength   As Long   'Buffer length'
Dim Buffer         As String  'Buffer to contain string'
Dim StringLength  As Long   'Length of string returned'
```

Dim CodedCharSetId	As Long	'Coded Character Set ID'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

## Multi **mqInquireStringFilter**

The `mqInquireStringFilter` call requests the value and operator of a string filter item that is present in the bag. The data item can be a user item or a system item.

### Syntax for `mqInquireStringFilter`

**`mqInquireStringFilter` (*Bag, Selector, ItemIndex, Bufferlength, Buffer, StringLength, CodedCharSetId, Operator, CompCode, Reason*)**

### Parameters for `mqInquireStringFilter`

#### Bag (MQHBAG) - input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

#### Selector (MQLONG) - input

Selector of the item to which the inquiry relates.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

The data type of the item must be the same as the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

The following special values can be specified for *Selector*:

##### **MQSEL\_ANY\_SELECTOR**

The item to be inquired about is a user or system item identified by *ItemIndex*.

##### **MQSEL\_ANY\_USER\_SELECTOR**

The item to be inquired about is a user item identified by *ItemIndex*.

##### **MQSEL\_ANY\_SYSTEM\_SELECTOR**

The item to be inquired about is a system item identified by *ItemIndex*.

#### ItemIndex (MQLONG) - input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND\_NONE. If the value is less than zero and not MQIND\_NONE, MQRC\_INDEX\_ERROR results. If the item is not already present in the bag, MQRC\_INDEX\_NOT\_PRESENT results. The following special value can be specified:

##### **MQIND\_NONE**

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

If MQSEL\_ANY\_SELECTOR is specified for the **Selector** parameter, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL\_ANY\_USER\_SELECTOR is specified for the **Selector** parameter, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL\_ANY\_SYSTEM\_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND\_NONE, zero, or greater.

**BufferLength (MQLONG) - input**

Length in bytes of the buffer to receive the condition string. Zero is a valid value.

**Buffer (MQCHAR x BufferLength) - output**

Buffer to receive the character condition string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

The string is padded with blanks to the length of the buffer; the string is not null-terminated. If the string is longer than the buffer, the string is truncated to fit; in this case *StringLength* indicates the size of the buffer needed to accommodate the string without truncation.

**StringLength (MQLONG) - output**

The length in bytes of the condition string contained in the bag. If the **Buffer** parameter is too small, the length of the string returned is less than *StringLength*.

**CodedCharSetId (MQLONG) - output**

The coded character set identifier for the character data in the string. This parameter can be set to a null pointer if not required.

**Operator (MQLONG) - output**

String filter operator in the bag.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqInquireStringFilter call:

**MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

**MQRC\_FILTER\_OPERATOR\_ERROR**

Filter operator not valid.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE, or MQIND\_NONE specified with one of the MQSEL\_ANY\_XXX\_SELECTOR values).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_STRING\_LENGTH\_ERROR**

**StringLength** parameter not valid (invalid parameter address).

**MQRC\_STRING\_TRUNCATED**

Data too long for output buffer and has been truncated.

**C language invocation for mqInquireStringFilter**

```
mqInquireStringFilter (Bag, Selector, ItemIndex,
BufferLength, Buffer, &StringLength, &CodedCharSetId,
&Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    Bag;           /* Bag handle */
MQLONG    Selector;      /* Selector */
MQLONG    ItemIndex;     /* Item index */
MQLONG    BufferLength;   /* Buffer length */
PMQCHAR   Buffer;        /* Buffer to contain string */
MQLONG    StringLength;  /* Length of string returned */
MQLONG    CodedCharSetId /* Coded Character Set ID */
MQLONG    Operator;      /* Item operator */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqInquireStringFilter**

(Supported on Windows only.)

```
mqInquireStringFilter Bag, Selector, ItemIndex,
BufferLength, Buffer, StringLength, CodedCharSetId,
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long   'Bag handle'
Dim Selector      As Long   'Selector'
Dim ItemIndex     As Long   'Item index'
Dim BufferLength   As Long   'Buffer length'
Dim Buffer         As String  'Buffer to contain string'
Dim StringLength  As Long   'Length of string returned'
Dim CodedCharSetId As Long   'Coded Character Set ID'
Dim Operator      As Long   'Item operator'
Dim CompCode      As Long   'Completion code'
Dim Reason        As Long   'Reason code qualifying CompCode'
```

**Multi mqPad**

The mqPad call pads a null-terminated string with blanks.

## Syntax for mqPad

**mqPad** (*String*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

## Parameters for mqPad

### String (PMQCHAR) - input

Null-terminated string. The null pointer is valid for the address of the **String** parameter, and denotes a string of zero length.

### BufferLength (MQLONG) - input

Length in bytes of the buffer to receive the string padded with blanks. Must be zero or greater.

### Buffer (MQCHAR x BufferLength) - output

Buffer to receive the blank-padded string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

If the number of characters preceding the first null in the **String** parameter is greater than the **BufferLength** parameter, the excess characters are omitted and MQRC\_DATA\_TRUNCATED results.

### CompCode (MQLONG) - output

Completion code.

### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqPad call:

#### **MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

#### **MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

#### **MQRC\_STRING\_ERROR**

String parameter not valid (invalid parameter address or buffer not completely accessible).

#### **MQRC\_STRING\_TRUNCATED**

Data too long for output buffer and has been truncated.

## Usage notes for mqPad

1. If the buffer pointers are the same, the padding is done in place. If not, at most *BufferLength* characters are copied into the second buffer; any space remaining, including the null-termination character, is overwritten with spaces.
2. If the *String* and **Buffer** parameters partially overlap, the result is undefined.

## C language invocation for mqPad

```
mqPad (String, BufferLength, Buffer, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQCHAR   String;           /* String to be padded */
MQLONG   BufferLength;     /* Buffer length */
PMQCHAR  Buffer;           /* Buffer to contain padded string */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

**Note:** This call is not supported in Visual Basic.

## **mqPutBag**

The mqPutBag call converts the contents of the specified bag into a PCF message and sends the message to the specified queue. The contents of the bag are unchanged after the call.

### **Syntax for mqPutBag**

**mqPutBag** (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *Bag*, *CompCode*, *Reason*)

### **Parameters for mqPutBag**

#### **Hconn (MQHCONN) - input**

MQI connection handle.

#### **Hobj (MQHOBJ) - input**

Object handle of the queue on which the message is to be placed. This handle was returned by a preceding MQOPEN call issued by the application. The queue must be open for output.

#### **MsgDesc (MQMD) - input/output**

Message descriptor. (For more information, see [MQMD - Message descriptor](#).)

If the *Format* field has a value other than MQFMT\_ADMIN, MQFMT\_EVENT, or MQFMT\_PCF, MQRC\_FORMAT\_NOT\_SUPPORTED results.

If the *Encoding* field has a value other than MQENC\_NATIVE, MQRC\_ENCODING\_NOT\_SUPPORTED results.

#### **PutMsgOpts (MQPMO) - input/output**

Put-message options. (For more information, see [MQPMO - Put-message options](#).)

#### **Bag (MQHBAG) - input**

Handle of the data bag to be converted to a message.

If the bag contains an administration message, and mqAddInquiry was used to insert values into the bag, the value of the MQIASY\_COMMAND data item must be an INQUIRE command recognized by the MQAI; MQRC\_INQUIRY\_COMMAND\_ERROR results if it is not.

If the bag contains nested system bags, MQRC\_NESTED\_BAG\_NOT\_SUPPORTED results.

#### **CompCode (MQLONG) - output**

Completion code.

#### **Reason (MQLONG) - output**

Reason code qualifying *CompCode*. The following reason codes indicating error and warning conditions can be returned from the mqPutBag call:

##### **MQRC\_\***

Anything from the MQPUT call or bag manipulation.

##### **MQRC\_BAG\_WRONG\_TYPE**

Input data bag is a group bag.

##### **MQRC\_ENCODING\_NOT\_SUPPORTED**

Encoding not supported (value in *Encoding* field in MQMD must be MQENC\_NATIVE).

### **MQRC\_FORMAT\_NOT\_SUPPORTED**

Format not supported (name in *Format* field in MQMD must be MQFMT\_ADMIN, MQFMT\_EVENT, or MQFMT\_PCF).

### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

### **MQRC\_INQUIRY\_COMMAND\_ERROR**

mqAddInquiry call used with a command code that is not a recognized INQUIRE command.

### **MQRC\_NESTED\_BAG\_NOT\_SUPPORTED**

Input data bag contains one or more nested system bags.

### **MQRC\_PARAMETER\_MISSING**

Administration message requires a parameter that is not present in the bag. This reason code occurs for bags created with the MQCBO\_ADMIN\_BAG or MQCBO\_REORDER\_AS\_REQUIRED options only.

### **MQRC\_SELECTOR\_WRONG\_TYPE**

mqAddString or mqSetString was used to add the MQIACF\_INQUIRY selector to the bag.

### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

## **C language invocation for mqPutBag**

```
mqPutBag (HConn, HObj, &MsgDesc, &PutMsgOpts, Bag,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHCONN  HConn;           /* MQI connection handle */  
MQHOBJ   HObj;           /* Object handle */  
MQMD     MsgDesc;        /* Message descriptor */  
MQPMO    PutMsgOpts;     /* Put-message options */  
MQHBAG   Bag;           /* Bag handle */  
MQLONG   CompCode;       /* Completion code */  
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

## **Visual Basic invocation for mqPutBag**

(Supported on Windows only.)

```
mqPutBag (HConn, HObj, MsgDesc, PutMsgOpts, Bag,  
CompCode, Reason);
```

Declare the parameters as follows:

```
Dim HConn      As Long 'MQI connection handle'  
Dim HObj       As Long 'Object handle'  
Dim MsgDesc    As MQMD 'Message descriptor'  
Dim PutMsgOpts As MQPMO 'Put-message options'  
Dim Bag        As Long 'Bag handle'  
Dim CompCode   As Long 'Completion code'  
Dim Reason     As Long 'Reason code qualifying CompCode'
```

## **Multi mqSetByteString**

The mqSetByteString call either modifies a byte string data item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but certain system-data items can also be modified.

## Syntax for mqSetByteString

**mqSetByteString** (*Bag*, *Selector*, *ItemIndex*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

## Parameters for mqSetByteString

### Bag (MQHBAG) - input

Handle of the bag to be set. This must be the handle of a bag created by the user, not the handle of a system bag; MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if you specify the handle of a system bag.

### Selector (MQLONG) - input

Selector of the item to be modified.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read only, MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC\_MULTIPLE\_INSTANCE\_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQBA\_FIRST through MQBA\_LAST; MQRC\_SELECTOR\_OUT\_OF\_RANGE results if it is not. If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND\_ALL is not specified for the **ItemIndex** parameter, the specified selector must already be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

If MQIND\_ALL is not specified for the **ItemIndex** parameter, the data type of the item must be the same as the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

### ItemIndex (MQLONG) - input

This identifies which occurrence of the item with the specified selector is to be modified. The value must be zero or greater, or one of the special values described in this topic; if it is none of these, MQRC\_INDEX\_ERROR results.

#### Zero or greater

The item with the specified index must already be present in the bag; MQRC\_INDEX\_NOT\_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

#### MQIND\_NONE

This specifies that there must be only one occurrence of the specified selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

#### MQIND\_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

### BufferLength (MQLONG) - input

The length in bytes of the byte string contained in the **Buffer** parameter. The value must be zero or greater.

### Buffer (MQBYTE x BufferLength) - input

Buffer containing the byte string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.



### CompCode (MQLONG) - output

Completion code.

### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the `mqSetByteString` call:

#### **MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

#### **MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

#### **MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE or MQIND\_ALL).

#### **MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

#### **MQRC\_MULTIPLE\_INSTANCE\_ERROR**

Multiple instances of system selector not valid.

#### **MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

#### **MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

#### **MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

#### **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

#### **MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

#### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

#### **MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

#### **MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE**

System item is read-only and cannot be altered.

## C language invocation for `mqSetByteString`

```
mqSetByteString (Bag, Selector, ItemIndex, BufferLength, Buffer,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */  
MQLONG   Selector;      /* Selector */  
MQLONG   ItemIndex;     /* Item index */  
MQLONG   BufferLength;   /* Buffer length */  
PMQBYTE  Buffer;         /* Buffer containing string */  
MQLONG   CompCode;      /* Completion code */  
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqSetByteString

(Supported on Windows only.)

```
mqSetByteString Bag, Selector, ItemIndex, BufferLength, Buffer,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long   'Bag handle'  
Dim Selector      As Long   'Selector'  
Dim ItemIndex     As Long   'Item index'  
Dim BufferLength  As Long   'Buffer length'  
Dim Buffer         As Byte   'Buffer containing string'  
Dim CompCode     As Long   'Completion code'  
Dim Reason       As Long   'Reason code qualifying CompCode'
```

Multi

## mqSetByteStringFilter

The `mqSetByteStringFilter` call either modifies a byte string filter item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but certain system-data items can also be modified.

### Syntax for mqSetByteStringFilter

**mqSetByteStringFilter** (*Bag, Selector, ItemIndex, Bufferlength, Buffer, Operator, CompCode, Reason*)

### Parameters for mqSetByteStringFilter

#### Bag (MQHBAG) - input

Handle of the bag to be set. This must be the handle of a bag created by the user, not the handle of a system bag; `MQRC_SYSTEM_BAG_NOT_ALTERABLE` results if you specify the handle of a system bag.

#### Selector (MQLONG) - input

Selector of the item to be modified.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; `MQRC_SELECTOR_NOT_SUPPORTED` results if it is not.

If the selector is a supported system selector, but is one that is read only, `MQRC_SYSTEM_ITEM_NOT_ALTERABLE` results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, `MQRC_MULTIPLE_INSTANCE_ERROR` results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the `MQCBO_CHECK_SELECTORS` option or as an administration bag (`MQCBO_ADMIN_BAG`), the selector must be in the range `MQBA_FIRST` through `MQBA_LAST`; `MQRC_SELECTOR_OUT_OF_RANGE` results if it is not. If `MQCBO_CHECK_SELECTORS` was not specified, the selector can be any value zero or greater.

If `MQIND_ALL` is not specified for the **ItemIndex** parameter, the specified selector must already be present in the bag; `MQRC_SELECTOR_NOT_PRESENT` results if it is not.

If `MQIND_ALL` is not specified for the **ItemIndex** parameter, the data type of the item must be the same as the data type implied by the call; `MQRC_SELECTOR_WRONG_TYPE` results if it is not.

**ItemIndex (MQLONG) - input**

This identifies which occurrence of the item with the specified selector is to be modified. The value must be zero or greater, or one of the special values described in this topic; if it is none of these, MQRC\_INDEX\_ERROR results.

**Zero or greater**

The item with the specified index must already be present in the bag; MQRC\_INDEX\_NOT\_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

**MQIND\_NONE**

This specifies that there must be only one occurrence of the specified selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

**MQIND\_ALL**

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

**BufferLength (MQLONG) - input**

The length in bytes of the condition byte string contained in the **Buffer** parameter. The value must be zero or greater.

**Buffer (MQBYTE x BufferLength) - input**

Buffer containing the condition byte string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

**Operator (MQLONG x Operator) - input**

Byte string filter operator to be placed in the bag. Valid operators are of the form MQCFOP\_\*

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqSetByteStringFilter call:

**MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

**MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

**MQRC\_FILTER\_OPERATOR\_ERROR**

Bag handle not valid.

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE or MQIND\_ALL).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_MULTIPLE\_INSTANCE\_ERROR**

Multiple instances of system selector not valid.

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE**

System item is read-only and cannot be altered.

**C language invocation for mqSetByteStringFilter**

```
mqSetByteStringFilter (Bag, Selector, ItemIndex, BufferLength, Buffer,
Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    Bag;           /* Bag handle */
MQQLONG   Selector;     /* Selector */
MQQLONG   ItemIndex;    /* Item index */
MQQLONG   BufferLength;  /* Buffer length */
PMQBYTE   Buffer;       /* Buffer containing string */
MQQLONG   Operator;     /* Operator */
PMQLONG   CompCode;     /* Completion code */
PMQLONG   Reason;      /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqSetByteStringFilter**

(Supported on Windows only.)

```
mqSetByteStringFilter Bag, Selector, ItemIndex, BufferLength, Buffer,
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long   'Bag handle'
Dim Selector      As Long   'Selector'
Dim ItemIndex     As Long   'Item index'
Dim BufferLength   As Long   'Buffer length'
Dim Buffer         As String  'Buffer containing string'
Dim Operator      As Long   'Item operator'
Dim CompCode     As Long   'Completion code'
Dim Reason        As Long   'Reason code qualifying CompCode'
```

## Multi **mqSetInteger**

The `mqSetInteger` call either modifies an integer item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but specific system-data items can also be modified.

### Syntax for `mqSetInteger`

`mqSetInteger (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason)`

### Parameters for `mqSetInteger`

#### Bag (MQHBAG) - input

Handle of the bag to be set. This must be the handle of a bag created by the user, and not the handle of a system bag; `MQRC_SYSTEM_BAG_NOT_ALTERABLE` results if the handle you specify refers to a system bag.

#### Selector (MQLONG) - input

Selector of the item to be modified. If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; `MQRC_SELECTOR_NOT_SUPPORTED` results if it is not.

If the selector is a supported system selector, but is one that is read-only, `MQRC_SYSTEM_ITEM_NOT_ALTERABLE` results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, `MQRC_MULTIPLE_INSTANCE_ERROR` results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the `MQCBO_CHECK_SELECTORS` option or as an administration bag (`MQCBO_ADMIN_BAG`), the selector must be in the range `MQIA_FIRST` through `MQIA_LAST`; `MQRC_SELECTOR_OUT_OF_RANGE` results if it is not. If `MQCBO_CHECK_SELECTORS` was not specified, the selector can be any value zero or greater.

If `MQIND_ALL` is not specified for the **ItemIndex** parameter, the specified selector must already be present in the bag; `MQRC_SELECTOR_NOT_PRESENT` results if it is not.

If `MQIND_ALL` is not specified for the **ItemIndex** parameter, the data type of the item must agree with the data type implied by the call; `MQRC_SELECTOR_WRONG_TYPE` results if it is not.

#### ItemIndex (MQLONG) - input

This value identifies the occurrence of the item with the specified selector that is to be modified. The value must be zero or greater, or one of the special values described in this topic; if it is none of these, `MQRC_INDEX_ERROR` results.

#### Zero or greater

The item with the specified index must already be present in the bag; `MQRC_INDEX_NOT_PRESENT` results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

#### MQIND\_NONE

This specifies that there must be one occurrence only of the specified selector in the bag. If there is more than one occurrence, `MQRC_SELECTOR_NOT_UNIQUE` results.

#### MQIND\_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

**Note:** For system selectors, the order is not changed.

**ItemValue (MQLONG) - input**

The integer value to be placed in the bag.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the `mqSetInteger` call:

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE or MQIND\_ALL).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_MULTIPLE\_INSTANCE\_ERROR**

Multiple instances of system selector not valid.

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not in valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE**

System item is read only and cannot be altered.

**C language invocation for mqSetInteger**

```
mqSetInteger (Bag, Selector, ItemIndex, ItemValue, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */
MQLONG   Selector;      /* Selector */
MQLONG   ItemIndex;     /* Item index */
MQLONG   ItemValue;     /* Integer value */
MQLONG   CompCode;     /* Completion code */
MQLONG   Reason;       /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqSetInteger**

(Supported on Windows only.)

```
mqSetInteger Bag, Selector, ItemIndex, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long 'Bag handle'  
Dim Selector      As Long 'Selector'  
Dim ItemIndex     As Long 'Item index'  
Dim ItemValue     As Long 'Integer value'  
Dim CompCode     As Long 'Completion code'  
Dim Reason       As Long 'Reason code qualifying CompCode'
```

## Multi **mqSetInteger64**

The `mqSetInteger64` call either modifies a 64-bit integer item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but specific system-data items can also be modified.

### Syntax for `mqSetInteger64`

**mqSetInteger64** (*Bag, Selector, ItemIndex, ItemValue, CompCode, Reason*)

### Parameters for `mqSetInteger64`

#### Bag (MQHBAG) - input

Handle of the bag to be set. This must be the handle of a bag created by the user, and not the handle of a system bag; `MQRC_SYSTEM_BAG_NOT_ALTERABLE` results if the handle you specify refers to a system bag.

#### Selector (MQLONG) - input

Selector of the item to be modified. If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; `MQRC_SELECTOR_NOT_SUPPORTED` results if it is not.

If the selector is a supported system selector, but is one that is read-only, `MQRC_SYSTEM_ITEM_NOT_ALTERABLE` results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, `MQRC_MULTIPLE_INSTANCE_ERROR` results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the `MQCBO_CHECK_SELECTORS` option or as an administration bag (`MQCBO_ADMIN_BAG`), the selector must be in the range `MQIA_FIRST` through `MQIA_LAST`; `MQRC_SELECTOR_OUT_OF_RANGE` results if it is not. If `MQCBO_CHECK_SELECTORS` was not specified, the selector can be any value zero or greater.

If `MQIND_ALL` is not specified for the **ItemIndex** parameter, the specified selector must already be present in the bag; `MQRC_SELECTOR_NOT_PRESENT` results if it is not.

If `MQIND_ALL` is not specified for the **ItemIndex** parameter, the data type of the item must agree with the data type implied by the call; `MQRC_SELECTOR_WRONG_TYPE` results if it is not.

#### ItemIndex (MQLONG) - input

This value identifies the occurrence of the item with the specified selector that is to be modified. The value must be zero or greater, or one of the special values described in this topic; if it is none of these, `MQRC_INDEX_ERROR` results.

#### Zero or greater

The item with the specified index must already be present in the bag;

`MQRC_INDEX_NOT_PRESENT` results if it is not. The index is counted relative to the items in

the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

**MQIND\_NONE**

This specifies that there must be one occurrence only of the specified selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

**MQIND\_ALL**

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

**Note:** For system selectors, the order is not changed.

**ItemValue (MQINT64) - input**

The integer value to be placed in the bag.

**CompCode (MQLONG) - output**

Completion code.

**Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqSetInteger64 call:

**MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE or MQIND\_ALL).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_MULTIPLE\_INSTANCE\_ERROR**

Multiple instances of system selector not valid.

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not in valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE**

System item is read only and cannot be altered.

## C language invocation for mqSetInteger64

```
mqSetInteger64 (Bag, Selector, ItemIndex, ItemValue, &CompCode, &Reason);
```



Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */
MQLONG   Selector;      /* Selector */
MQLONG   ItemIndex;     /* Item index */
MQINT64  ItemValue;     /* Integer value */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqSetInteger64

(Supported on Windows only.)

```
mqSetInteger64 Bag, Selector, ItemIndex, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector  As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim ItemValue As Long 'Integer value'
Dim CompCode  As Long 'Completion code'
Dim Reason    As Long 'Reason code qualifying CompCode'
```

## mqSetIntegerFilter

The mqSetIntegerFilter call either modifies an integer filter item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but specific system-data items can also be modified.

### Syntax for mqSetIntegerFilter

**mqSetIntegerFilter** (*Bag, Selector, ItemIndex, ItemValue, Operator, CompCode, Reason*)

### Parameters for mqSetIntegerFilter

#### Bag (MQHBAG) - input

Handle of the bag to be set. This must be the handle of a bag created by the user, and not the handle of a system bag; MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if the handle you specify refers to a system bag.

#### Selector (MQLONG) - input

Selector of the item to be modified. If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read-only, MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC\_MULTIPLE\_INSTANCE\_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQIA\_FIRST through MQIA\_LAST; MQRC\_SELECTOR\_OUT\_OF\_RANGE results if it is not. If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND\_ALL is not specified for the **ItemIndex** parameter, the specified selector must already be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

If MQIND\_ALL is not specified for the **ItemIndex** parameter, the data type of the item must agree with the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

#### **ItemIndex (MQLONG) - input**

This value identifies the occurrence of the item with the specified selector that is to be modified. The value must be zero or greater, or one of the special values described in this topic; if it is none of these, MQRC\_INDEX\_ERROR results.

##### **Zero or greater**

The item with the specified index must already be present in the bag; MQRC\_INDEX\_NOT\_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

##### **MQIND\_NONE**

This specifies that there must be one occurrence only of the specified selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

##### **MQIND\_ALL**

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

**Note:** For system selectors, the order is not changed.

#### **ItemValue (MQLONG) - input**

The integer condition value to be placed in the bag.

#### **Operator (MQLONG) - input**

The integer filter operator to be placed in the bag. Valid operators are of the form MQCFOP\_\*

#### **CompCode (MQLONG) - output**

Completion code.

#### **Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqSetIntegerFilter call:

##### **MQRC\_FILTER\_OPERATOR\_ERROR**

Filter operator not valid.

##### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

##### **MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE or MQIND\_ALL).

##### **MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

##### **MQRC\_MULTIPLE\_INSTANCE\_ERROR**

Multiple instances of system selector not valid.

##### **MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

##### **MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

##### **MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not in valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE**

System item is read only and cannot be altered.

**C language invocation for mqSetIntegerFilter**

```
mqSetIntegerFilter (Bag, Selector, ItemIndex, ItemValue, Operator,
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */
MQLONG   Selector;      /* Selector */
MQLONG   ItemIndex;     /* Item index */
MQLONG   ItemValue;     /* Integer value */
MQLONG   Operator;      /* Item operator */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqSetIntegerFilter**

(Supported on Windows only.)

```
mqSetIntegerFilter Bag, Selector, ItemIndex, ItemValue, Operator,
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim ItemValue As Long 'Integer value'
Dim Operator As Long 'Item operator'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

**Multi mqSetString**

The mqSetString call either modifies a character data item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but certain system-data items can also be modified.

**Syntax for mqSetString**

```
mqSetString (Bag, Selector, ItemIndex, Bufferlength, Buffer, CompCode, Reason)
```

## Parameters for mqSetString

### Bag (MQHBAG) - input

Handle of the bag to be set. This must be the handle of a bag created by the user, not the handle of a system bag; MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if you specify the handle of a system bag.

### Selector (MQLONG) - input

Selector of the item to be modified.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read only, MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC\_MULTIPLE\_INSTANCE\_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQCA\_FIRST through MQCA\_LAST; MQRC\_SELECTOR\_OUT\_OF\_RANGE results if it is not. If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND\_ALL is not specified for the **ItemIndex** parameter, the specified selector must already be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

If MQIND\_ALL is not specified for the **ItemIndex** parameter, the data type of the item must be the same as the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

### ItemIndex (MQLONG) - input

This identifies which occurrence of the item with the specified selector is to be modified. The value must be zero or greater, or one of the special values described in this topic; if it is none of these, MQRC\_INDEX\_ERROR results.

#### Zero or greater

The item with the specified index must already be present in the bag; MQRC\_INDEX\_NOT\_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

#### MQIND\_NONE

This specifies that there must be only one occurrence of the specified selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

#### MQIND\_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

### BufferLength (MQLONG) - input

The length in bytes of the string contained in the **Buffer** parameter. The value must be zero or greater, or the special value MQBL\_NULL\_TERMINATED.

If MQBL\_NULL\_TERMINATED is specified, the string is delimited by the first null encountered in the string.

If MQBL\_NULL\_TERMINATED is not specified, *BufferLength* characters are inserted into the bag, even if null characters are present; the nulls do not delimit the string.

### Buffer (MQCHAR x BufferLength) - input

Buffer containing the character string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

### **CompCode (MQLONG) - output**

Completion code.

### **Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the `mqSetString` call:

#### **MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

#### **MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

#### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

#### **MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE or MQIND\_ALL).

#### **MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

#### **MQRC\_MULTIPLE\_INSTANCE\_ERROR**

Multiple instances of system selector not valid.

#### **MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

#### **MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

#### **MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

#### **MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

#### **MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

#### **MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

#### **MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

#### **MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE**

System item is read-only and cannot be altered.

## **Usage notes for mqSetString**

The Coded Character Set ID (CCSID) associated with this string is copied from the current CCSID of the bag.

## **C language invocation for mqSetString**

```
mqSetString (Bag, Selector, ItemIndex, BufferLength, Buffer,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag; /* Bag handle */
```

```

MQLONG Selector; /* Selector */
MQLONG ItemIndex; /* Item index */
MQLONG BufferLength; /* Buffer length */
PMQCHAR Buffer; /* Buffer containing string */
MQLONG CompCode; /* Completion code */
MQLONG Reason; /* Reason code qualifying CompCode */

```

## Visual Basic invocation for mqSetString

(Supported on Windows only.)

```

mqSetString Bag, Selector, ItemIndex, BufferLength, Buffer,
CompCode, Reason

```

Declare the parameters as follows:

```

Dim Bag As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim BufferLength As Long 'Buffer length'
Dim Buffer As String 'Buffer containing string'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'

```

### mqSetStringFilter

The mqSetStringFilter call either modifies a string filter item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but certain system-data items can also be modified.

## Syntax for mqSetStringFilter

**mqSetStringFilter** (*Bag, Selector, ItemIndex, Bufferlength, Buffer, Operator, CompCode, Reason*)

## Parameters for mqSetStringFilter

### Bag (MQHBAG) - input

Handle of the bag to be set. This must be the handle of a bag created by the user, not the handle of a system bag; MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE results if you specify the handle of a system bag.

### Selector (MQLONG) - input

Selector of the item to be modified.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC\_SELECTOR\_NOT\_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read only, MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC\_MULTIPLE\_INSTANCE\_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO\_CHECK\_SELECTORS option or as an administration bag (MQCBO\_ADMIN\_BAG), the selector must be in the range MQCA\_FIRST through MQCA\_LAST; MQRC\_SELECTOR\_OUT\_OF\_RANGE results if it is not. If MQCBO\_CHECK\_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND\_ALL is not specified for the **ItemIndex** parameter, the specified selector must already be present in the bag; MQRC\_SELECTOR\_NOT\_PRESENT results if it is not.

If MQIND\_ALL is not specified for the **ItemIndex** parameter, the data type of the item must be the same as the data type implied by the call; MQRC\_SELECTOR\_WRONG\_TYPE results if it is not.

#### **ItemIndex (MQLONG) - input**

This identifies which occurrence of the item with the specified selector is to be modified. The value must be zero or greater, or one of the special values described in this topic; if it is none of these, MQRC\_INDEX\_ERROR results.

##### **Zero or greater**

The item with the specified index must already be present in the bag; MQRC\_INDEX\_NOT\_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

##### **MQIND\_NONE**

This specifies that there must be only one occurrence of the specified selector in the bag. If there is more than one occurrence, MQRC\_SELECTOR\_NOT\_UNIQUE results.

##### **MQIND\_ALL**

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

#### **BufferLength (MQLONG) - input**

The length in bytes of the condition string contained in the **Buffer** parameter. The value must be zero or greater, or the special value MQBL\_NULL\_TERMINATED.

If MQBL\_NULL\_TERMINATED is specified, the string is delimited by the first null encountered in the string.

If MQBL\_NULL\_TERMINATED is not specified, *BufferLength* characters are inserted into the bag, even if null characters are present; the nulls do not delimit the string.

#### **Buffer (MQCHAR x BufferLength) - input**

Buffer containing the character condition string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

#### **Operator (MQLONG x Operator) - input**

String filter operator to be placed in the bag. Valid operators are of the form MQCFOP\_\*

#### **CompCode (MQLONG) - output**

Completion code.

#### **Reason (MQLONG) - output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqSetStringFilter call:

##### **MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

##### **MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

##### **MQRC\_FILTER\_OPERATOR\_ERROR**

Bag handle not valid.

##### **MQRC\_HBAG\_ERROR**

Bag handle not valid.

**MQRC\_INDEX\_ERROR**

Index not valid (index negative and not MQIND\_NONE or MQIND\_ALL).

**MQRC\_INDEX\_NOT\_PRESENT**

No item with the specified index is present within the bag for the selector given.

**MQRC\_MULTIPLE\_INSTANCE\_ERROR**

Multiple instances of system selector not valid.

**MQRC\_SELECTOR\_NOT\_PRESENT**

No item with the specified selector is present within the bag.

**MQRC\_SELECTOR\_NOT\_SUPPORTED**

Specified system selector not supported by the MQAI.

**MQRC\_SELECTOR\_NOT\_UNIQUE**

MQIND\_NONE specified when more than one occurrence of the specified selector is present in the bag.

**MQRC\_SELECTOR\_OUT\_OF\_RANGE**

Selector not within valid range for call.

**MQRC\_SELECTOR\_WRONG\_TYPE**

Data item has wrong data type for call.

**MQRC\_STORAGE\_NOT\_AVAILABLE**

Insufficient storage available.

**MQRC\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

**MQRC\_SYSTEM\_ITEM\_NOT\_ALTERABLE**

System item is read-only and cannot be altered.

**Usage notes for mqSetStringFilter**

The Coded Character Set ID (CCSID) associated with this string is copied from the current CCSID of the bag.

**C language invocation for mqSetStringFilter**

```
mqSetStringFilter (Bag, Selector, ItemIndex, BufferLength, Buffer,
Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG   Bag;           /* Bag handle */
MQLONG   Selector;      /* Selector */
MQLONG   ItemIndex;     /* Item index */
MQLONG   BufferLength;  /* Buffer length */
PMQCHAR  Buffer;        /* Buffer containing string */
MQLONG   Operator;     /* Item operator */
MQLONG   CompCode;     /* Completion code */
MQLONG   Reason;       /* Reason code qualifying CompCode */
```

**Visual Basic invocation for mqSetStringFilter**

(Supported on Windows only.)

```
mqSetStringFilter Bag, Selector, ItemIndex, BufferLength, Buffer,
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag           As Long   'Bag handle'
```



Dim Selector	As Long	'Selector'
Dim ItemIndex	As Long	'Item index'
Dim BufferLength	As Long	'Buffer length'
Dim Buffer	As String	'Buffer containing string'
Dim Operator	As Long	'Item operator'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

## Multi mqTrim

The mqTrim call trims the blanks from a blank-padded string, then terminates it with a null.

### Syntax for mqTrim

**mqTrim** (*BufferLength*, *Buffer*, *String*, *CompCode*, *Reason*)

### Parameters for mqTrim

#### BufferLength (MQLONG) - input

Length in bytes of the buffer containing the string padded with blanks. Must be zero or greater.

#### Buffer (MQCHAR × BufferLength) - input

Buffer containing the blank-padded string. The length is given by the **BufferLength** parameter. If zero is specified for **BufferLength**, the null pointer can be specified for the address of the **Buffer** parameter; in all other cases, a valid (nonnull) address must be specified for the **Buffer** parameter.

#### String (MQCHAR × (BufferLength + 1)) - output

Buffer to receive the null-terminated string. The length of this buffer must be at least one byte greater than the value of the **BufferLength** parameter.

#### CompCode (MQLONG) - output

Completion code.

#### Reason (MQLONG) - output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqTrim call:

#### **MQRC\_BUFFER\_ERROR**

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

#### **MQRC\_BUFFER\_LENGTH\_ERROR**

Buffer length not valid.

#### **MQRC\_STRING\_ERROR**

String parameter not valid (invalid parameter address or buffer not completely accessible).

### Usage notes for mqTrim

1. If the two buffer pointers are the same, the trimming is done in place. If they are not the same, the blank-padded string is copied into the null-terminated string buffer. After copying, the buffer is scanned backwards from the end until a nonspace character is found. The byte following the nonspace character is then overwritten with a null character.
2. If *String* and *Buffer* partially overlap, the result is undefined.

### C language invocation for mqTrim

```
mqTrim (BufferLength, Buffer, String, &CompCode, &Reason);
```

Declare the parameters as follows:

```

MQLONG   BufferLength;      /* Buffer length */
PMQCHAR  Buffer;           /* Buffer containing blank-padded string */
MQCHAR   String[n+1];    /* String with blanks discarded */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */

```

**Note:** This call is not supported in Visual Basic.

## **mqTruncateBag**

The `mqTruncateBag` call reduces the number of user items in a user bag to the specified value, by deleting user items from the end of the bag.

### Syntax for `mqTruncateBag`

`mqTruncateBag (Bag, ItemCount, CompCode, Reason)`

### Parameters for `mqTruncateBag`

#### Bag (MQHBAG) - input

Handle of the bag to be truncated. This must be the handle of a bag created by the user, not the handle of a system bag; `MQR_SYSTEM_BAG_NOT_ALTERABLE` results if you specify the handle of a system bag.

#### ItemCount (MQLONG) - input

The number of user items to remain in the bag after truncation. Zero is a valid value.

**Note:** The `ItemCount` parameter is the number of data items, not the number of unique selectors. (If there are one or more selectors that occur multiple times in the bag, there will be fewer selectors than data items before truncation.) Data items are deleted from the end of the bag, in the opposite order to which they were added to the bag.

If the number specified exceeds the number of user items currently in the bag, `MQR_ITEM_COUNT_ERROR` results.

#### CompCode (MQLONG) - output

Completion code.

#### Reason (MQLONG) - output

Reason code qualifying `CompCode`.

The following reason codes indicating error conditions can be returned from the `mqTruncateBag` call:

#### **MQR\_HBAG\_ERROR**

Bag handle not valid.

#### **MQR\_ITEM\_COUNT\_ERROR**

`ItemCount` parameter not valid (value exceeds the number of user data items in the bag).

#### **MQR\_SYSTEM\_BAG\_NOT\_ALTERABLE**

System bag cannot be altered or deleted.

### Usage notes for `mqTruncateBag`

1. System items in a bag are not affected by `mqTruncateBag`; the call cannot be used to truncate system bags.
2. `mqTruncateBag` with an `ItemCount` of zero is not the same as the `mqClearBag` call. The former deletes all of the user items but leaves the system items intact, and the latter deletes all of the user items and resets the system items to their initial values.

## C language invocation for mqTruncateBag

```
mqTruncateBag (Bag, ItemCount, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    hBag;           /* Bag handle */
MQLONG    ItemCount;     /* Number of items to remain in bag */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

## Visual Basic invocation for mqTruncateBag

(Supported on Windows only.)

```
mqTruncateBag Bag, ItemCount, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim ItemCount As Long 'Number of items to remain in bag'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```



## MQAI selectors

Items in bags are identified by a *selector* that acts as an identifier for the item. There are two types of selector, *user selector* and *system selector*.

### User selectors

User selectors have values that are zero or positive. For the administration of MQSeries objects, valid user selectors are already defined by the following constants:

- MQCA\_\* and MQIA\_\* (object attributes)
- MQCACF\_\* and MQIACF\_\* (items relating specifically to PCF)
- MQCACH\_\* and MQIACH\_\* (channel attributes)

For user messages, the meaning of a user selector is defined by the application.

The following additional user selectors are introduced by the MQAI:

#### **MQIACF\_INQUIRY**

Identifies an IBM MQ object attribute to be returned by an Inquire command.

#### **MQHA\_BAG\_HANDLE**

Identifies a bag handle residing within another bag.

#### **MQHA\_FIRST**

Lower limit for handle selectors.

#### **MQHA\_LAST**

Upper limit for handle selectors.

#### **MQHA\_LAST\_USED**

Upper limit for last handle selector allocated.

#### **MQCA\_USER\_LIST**

Default user selector. Supported on Visual Basic only. This selector supports character type and represents the default value used if the **Selector** parameter is omitted on the mqAdd\*, mqSet\*, or mqInquire\* calls.

## **MQIA\_USER\_LIST**

Default user selector. Supported on Visual Basic only. This selector supports integer type and represents the default value used if the **Selector** parameter is omitted on the mqAdd\*, mqSet\*, or mqInquire\* calls.

## **System selectors**

System selectors have negative values. The following system selectors are included in the bag when it is created:

### **MQIASY\_BAG\_OPTIONS**

Bag-creation options. A summation of the options used to create the bag. This selector cannot be changed by the user.

### **MQIASY\_CODED\_CHAR\_SET\_ID**

Character-set identifier for the character data items in the bag. The initial value is the queue manager's character set.

The value in the bag is used on entry to the mqExecute call and set on exit from the mqExecute call. This also applies when character strings are added to or modified in the bag.

### **MQIASY\_COMMAND**

PCF command identifier. Valid values are the MQCMD\_\* constants. For user messages, the value MQCMD\_NONE should be used. The initial value is MQCMD\_NONE.

The value in the bag is used on entry to the mqPutBag and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag and mqBufferToBag calls.

### **MQIASY\_COMP\_CODE**

Completion code. Valid values are the MQCC\_\* constants. The initial value is MQCC\_OK.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

### **MQIASY\_CONTROL**

PCF control options. Valid values are the MQCFC\_\* constants. The initial value is MQCFC\_LAST.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

### **MQIASY\_MSG\_SEQ\_NUMBER**

PCF message sequence number. Valid values are 1 or greater. The initial value is 1.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

### **MQIASY\_REASON**

Reason code. Valid values are the MQRC\_\* constants. The initial value is MQRC\_NONE.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

### **MQIASY\_TYPE**

PCF command type. Valid values are the MQCFT\_\* constants. For user messages, the value MQCFT\_USER should be used. The initial value is MQCFT\_USER for bags created as user bags and MQCFT\_COMMAND for bags created as administration or command bags.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

### **MQIASY\_VERSION**

PCF version. Valid values are the MQCFH\_VERSION\_\* constants. The initial value is MQCFH\_VERSION\_1.

If the value in the bag is set to a value other than MQCFH\_VERSION\_1, the value is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls. If the value in the bag is MQCFH\_VERSION\_1,

the PCF version is the lowest value required for the parameter structures that are present in the message.

The value in the bag is set on exit from the `mqExecute`, `mqGetBag`, and `mqBufferToBag` calls.

## Managed File Transfer administration reference

---

Use the following reference information to help you administer Managed File Transfer.

### Related reference

[“MFT commands reference” on page 1977](#)

All Managed File Transfer (MFT) commands are listed with links to their detailed descriptions.

## How MFT agents allocate source transfer slots to new requests

A managed file transfer (MFT) agent contains a number of source transfer slots. Each source transfer slot holds either details of a managed transfer that the agent is currently acting as the source agent for, or details of a managed call that the agent is currently processing.

The number of source transfer slots on an agent is specified by the agent property **maxSourceTransfers**, which has a default value of 25.

An agent also has a number of queued transfer slots as well. These slots are used to hold managed transfer or managed call requests that are currently on the agent's backlog waiting to be processed. The number of queued transfer slots is specified by the agent property **maxQueuedTransfers**. The default value of this property is 1000.

When an agent receives either a managed transfer request asking it to act as the source agent or a managed call request, it checks to see if it has a free source transfer slot.

If the agent does have a free transfer slot, the managed transfer or managed calls is assigned to one of the slots and the agent starts processing it.

If all of the source transfer slots are occupied, the agent assigns the managed transfer or managed call a queued transfer slot, so that it can be processed later.

However, if all of the queued transfer slots are full, the managed transfer request is rejected and the agent writes the following message to its event log:

```
BFGSS0030W: The agent is already acting as the source agent for the maximum number of file transfer operations and unable to queue further requests, due to the queued transfer limit of <maxQueuedTransfers> being reached. The new transfer request will not be carried out.
```

When a managed transfer or managed call completes (either successfully, or due to an error), its source transfer slot is released. The agent then moves a managed transfer or managed call from a queued transfer slot to the free source transfer slot, and starts processing it.

See the [Advanced agent properties: Transfer limit](#) section of the topic [The MFT agent.properties file](#) for more information about the **maxSourceTransfers** and **maxQueuedTransfers** properties.

## MFT agent status values

The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

### ACTIVE

The agent is running and is sending or receiving files. The agent is publishing its status at regular intervals. The last update was received within the expected time period.

### ENDED UNEXPECTEDLY

The agent has ended unexpectedly. The agent will be automatically restarted, unless there have been more than **maxRestartCount** restarts within the **maxRestartInterval** time period and the **maxRestartDelay** value is less than or equal to 0. For more information about these properties, see [The agent.properties file](#).

## NO\_INFORMATION

The agent is not publishing updates in a form that this command can process.

## PROBLEM

The agent command handler might not be working. The agent is publishing status messages, but these status messages are out of date.

## READY

The agent is running, but is not sending or receiving files. The agent is publishing its status at regular intervals. The last update was received within the expected time period.

## STARTING

The agent is starting, but is not yet ready to perform transfers.

## STOPPED

The agent has been stopped.

## V9.3.0 V9.3.0 STOPPING

The agent has been stopped in a controlled manner and is in a transient state. While the agent is in this state, it does not accept any new managed transfer requests and waits for any in-progress transfers to complete before it shuts itself down. For more information, see [Stopping an MFT agent](#).

## UNKNOWN

The status of the agent cannot be determined. It might have published a status which is not recognized by this tool. If you have mixed product versions on your network, upgrading the installation version of this tool might fix this problem.

From IBM MQ 9.1.0, when you run commands or look at the list of agents connecting to a coordination manager and their individual properties, you can see a new **Status Age** value for the agent that shows the age of their last reported status. For more information, see [What to do if an agent is shown as being in an UNKNOWN state](#).

## Related concepts

[What to do if you think that your file transfer is stuck](#)

[What to do if an agent is shown as being in an UNKNOWN state](#)

## Related reference

[“MFT agent transfer states” on page 2617](#)

A Managed File Transfer Agent that is started publishes its details to the SYSTEM.FTE topic on its coordination queue manager. These details include the states of each of the current transfers that involved that agent.

[“fteListAgents \(list the MFT agents for a coordination queue manager\)” on page 2085](#)

Use the **fteListAgents** command to list all of the Managed File Transfer agents that are registered with a particular coordination queue manager.

[“fteShowAgentDetails \(display MFT agent details\)” on page 2129](#)

Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

## Multi MFT process controller overview

The IBM MQ Managed File Transfer (MFT) process controller is responsible for starting an MFT agent, and restarting that process if it ends for any reason. There is one process controller for every agent process.

**Note:** The process controller is applicable to IBM MQ for Multiplatforms only.

**z/OS** On IBM MQ for z/OS the agent process is restarted by Automatic Restart Manager (ARM). For more information on this, see [Configuring MFT for the z/OS Automatic Restart Manager \(ARM\)](#)

## How the process controller works

When the **fteStartAgent** command is run, it starts up an instance of the process controller for that agent and the process controller then starts the agent process.

When the **fteStopAgent** command is run, it connects to the process controller for that agent and sends it a stop request. The process controller receives the request, stops the agent process and then shuts itself down.

The process controller monitors the agent process. If the agent process stops unexpectedly, the process controller restarts it.

By default, if an agent process stops five times within a two minute period, the process controller shuts itself down and does not try to restart the agent again. In this situation, you need to restart the agent manually, using the **fteStartAgent** command.

You can change this behavior by modifying the following agent properties:

- **maxRestartCount**
- **maxRestartDelay**
- **maxRestartInterval**

If you have configured an agent to connect to its agent queue manager using the BINDINGS transport, the process controller creates a connection to this queue manager when it starts up. The process controller then monitors this connection.

If the connection is broken because the queue manager has become unavailable, the process controller stops the agent and then attempts to re-establish the connection at regular intervals.

The time period between reconnection attempts is determined by the agent property **agentQMgrRetryInterval**. Once the queue manager is available again and the process controller has been able to connect to it, the process controller restarts the agent process.

**Note:** When an agent is configured to connect to its agent queue manager using the CLIENT transport, the agent process remains active if it becomes disconnected from the queue manager. In this situation, the agent process tries to reconnect itself at regular intervals.

For more information on the four properties mentioned in this section, see the [Advanced agent properties: Process controller](#) section of *The MFT agent.properties* file topic.

## Process controller log files

The process controller writes informational messages to its event log. This is a file called peventN.log, where N is a number, which can be found in the following directory: MQ\_DATA\_PATH/mqft/logs/coordination\_qmgr\_name/agents/agent\_name/logs/

The size of each process controller event log file, and the number of historical files, is determined by the agent properties **outputLogSize** and **outputLogFiles**.

For more information on the properties mentioned in this section, see the [Advanced agent properties: Tracing and logging](#) section of *The MFT agent.properties* file topic.

**Note:** These properties are also used to determine the size and number of agent log files (called outputN.log) as well as the process controller log files.

The messages written to the process controller event log include the process identifier of the process controller, and the process identifier of the agent process. Some examples of these messages are shown here:

```
[21/06/2022 16:17:40.000 GMT Daylight Time] 00000000000049e0
ProcessContro I BFGPC0003I: IBM MQ Managed File Transfer process controller started.
Log files located at: C:\ProgramData\IBM\MQ\mqft\logs\QM1\agents\AGENT1.
```

```
[21/06/2022 16:17:55.000 GMT Daylight Time] 00000000000049e0
ProcessContro I BFGPC0007I: IBM MQ Managed File Transfer process controller with process
identifier 18736 started AGENT1@QM1 with process identifier 1748.
```

```
[21/06/2022 16:19:20.000 GMT Daylight Time] 00000000000049e0
ProcessContro I BFGPC0027W: Process has ended with return code 1 and will be
restarted to attempt to recover the problem.
```

```
[21/06/2022 16:19:20.000 GMT Daylight Time] 00000000000049e0
ProcessContro I BFGPC0007I: IBM MQ Managed File Transfer process controller with process
identifier 18736 started AGENT1@QM1 with process identifier 1304.
```

Here, the process controller associated with the agent AGENT1 was running with process identifier 18736.

Initially, it started the agent process – the process identifier for this process was 1748.

Shortly after the agent started, the process controller detected that it had stopped unexpectedly and so restarted it. Following the restart, the process identifier for the agent process is 1304.

#### **Related reference**

[“MFT process controller exit codes” on page 2490](#)

If the Managed File Transfer process controller ends, a BFGPC0004I message is generated with an exit code that gives the reason why the process controller ended.

## **MFT agent process controller status values**

The **fteShowAgentDetails** command produces agent process controller status information. There are several possible values for this status.

#### **WAITING**

The agent process controller is waiting for the queue manager to become available before starting the agent.

#### **STARTED**

The agent process controller has started the agent process.

#### **STOPPED**

The agent process controller has been stopped, either because of a request to stop the agent or because there have been too many agent process restarts within the restart interval.

#### **RECOVERING**

The agent process unexpectedly stopped and the process controller will attempt to restart it.

#### **ISTOPPING**

The agent process has received a request to shut down immediately. When the agent process has stopped, the process controller will stop.

#### **CSTOPPING**

The agent process has received a request to shut down in a controlled manner. When the agent process has stopped, the process controller will stop.

#### **UNKNOWN**

The agent process controller status cannot be determined. It might be that the agent process controller is not running, or that it is running on a different system from where the fteShowAgentDetails command was run.

#### **Related reference**

[fteShowAgentDetails](#)



Use the **fteShowAgentDetails** command to display the details of a particular Managed File Transfer Agent. These are the details that are stored by the agent's Managed File Transfer coordination queue manager.

## MFT logger status values

The **fteShowLoggerDetails** commands produce logger status information. There are several possible values for this status.

### ACTIVE

The logger is running and is sending or receiving files. The logger is publishing its status at regular intervals. The last update was received within the expected time period.

### READY

The logger is running, but is not sending or receiving files. The logger is publishing its status at regular intervals. The last update was received within the expected time period.

### STARTING

The logger is starting, but is not yet ready to perform transfers.

### UNREACHABLE

Logger status updates were not received at the expected time intervals. The logger might have stopped running due to an error, or have been shut down abruptly, or be running but experiencing communication problems.

### STOPPED

The logger has been stopped. It was shut down in a controlled manner.

### ENDED UNEXPECTEDLY

The logger has ended unexpectedly. The logger will be automatically restarted, unless there have been more than `maxRestartCount` restarts within the `maxRestartInterval` time period and the `maxRestartDelay` value is less than or equal to 0. For more information about these properties, see [MFT logger configuration properties](#).

For the **fteShowLoggerDetails** command the details for the this status will include a status code, which is the logger process exit code. See "Process Exit Codes" for a list of known exit codes.

### NO\_INFORMATION

The logger version might be IBM WebSphere MQ File Transfer Edition 7.0.2 earlier. The logger is not publishing updates in a form that this command can process.

### UNKNOWN

The status of the logger cannot be determined. It might have published a status which is not recognized by this tool. If you have mixed product versions on your network, upgrading the installation version of this tool might fix this problem.

### PROBLEM

The logger command handler might not be working. The logger is publishing status messages, but these status messages are out of date.

### Related reference

[“fteShowLoggerDetails \(display MFT logger details\)” on page 2137](#)

Use the **fteShowLoggerDetails** command to display the details of a particular Managed File Transfer logger.

## MFT logger process controller status values

The **fteShowLoggerDetails** command produces logger process controller status information. There are several possible values for this status.

### WAITING

The logger process controller is waiting for the queue manager to become available before starting the logger.

**STARTED**

The logger process controller has started the logger process.

**STOPPED**

The logger process controller has been stopped, either because of a request to stop the logger or because there have been too many logger process restarts within the restart interval.

**RECOVERING**

The logger process unexpectedly stopped and the process controller will attempt to restart it.

**ISTOPPING**

The logger process has received a request to shut down immediately. When the logger process has stopped, the process controller will stop.

**CSTOPPING**

The logger process has received a request to shut down in a controlled manner. When the logger process has stopped, the process controller will stop.

**UNKNOWN**

The logger process controller status cannot be determined. It might be that the logger process controller is not running, or that it is running on a different system from where the `fteShowLoggerDetails` command was run.

**Related reference**

[“fteShowLoggerDetails \(display MFT logger details\)” on page 2137](#)

Use the **fteShowLoggerDetails** command to display the details of a particular Managed File Transfer logger.

## MFT process controller exit codes

If the Managed File Transfer process controller ends, a BFGPC0004I message is generated with an exit code that gives the reason why the process controller ended.

The following message appears to indicate that process controller has ended:

```
BFGPC0004I IBM MQ Managed File Transfer process controller ended with exit code reason_code.
```

where *reason\_code* shows the reason why the process controller has ended.

**Note:** Exit codes from the process controller mostly reflect the standard operating system exit codes, but some exit codes are defined for specific purposes and are always accompanied with a specific message in the process controller log file.

Reason code	Description
RC_SUCCESS = 0	The process controller ended successfully.
RC_FAILURE = 1	General process controller failure return code (should in general not be returned).
RC_EXIT = 2	The process controller was forced to exit (for example, a diagnostic system requested the process controller to terminate).
RC_ABEND = 70	The process controller has had an unrecoverable problem and is forcibly terminating.
RC_QMUNAVAIL = 75	The process controller cannot continue because the queue manager for the process controller is unavailable.
RC_CONFIG = 78	The process controller cannot continue because there is a problem with the startup configuration data.

These exit codes are written to `pceventX.log`, where *X* can be any number, for example the log file name can be `pcevent0.log`.

## Related reference

[Managed File Transfer diagnostic messages: BFGPC0001 - BFGPC9999](#)

## Guidelines for transferring files

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

Read the relevant topics for further information.

### Related tasks

[“Transferring text files between Connect:Direct and MFT” on page 2510](#)

Text transfer involves converting the text of a file from one code page to another, and converting CRLF (carriage return line feed) characters between systems.

### Related reference

[“Transferring files and data sets between z/OS and distributed systems” on page 2492](#)

You can transfer files and supported data set types between z/OS and distributed file systems by using Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

[“Transferring between data sets on z/OS” on page 2493](#)

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

[“Transferring data sets to and from Connect:Direct nodes” on page 2500](#)

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

[“Mappings between Connect:Direct process statement parameters and BPXWDYN keys” on page 2501](#)

When you submit a transfer request for a data set where either the source or destination is a Connect:Direct node, any supported BPXWDYN keys that you provide are converted to a format that is accepted by Connect:Direct processes.

[“BPXWDYN properties you must not use with MFT” on page 2507](#)

Some BPXWDYN options must not be specified when using the **fteCreateTemplate** command, the **fteCreateTransfer** command or the **bpxwdynAllocAdditionalOptions** property in the `agent.properties` file.

[“Transferring text files with MFT” on page 2507](#)

Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of Managed File Transfer.

[“Transferring files to or from protocol bridge agents” on page 2510](#)

You can transfer files to and from an FTP or SFTP file server outside your Managed File Transfer network using a protocol bridge agent.

[“Transferring files to or from IBM i systems” on page 2511](#)

If you transfer files to or from IBM i systems using Managed File Transfer in text mode and you want to convert the data in the files, consider the information in this topic.

[“Transferring save files located in QSYS.LIB on IBM i” on page 2515](#)

Managed File Transfer supports the transfer of save files located in the QSYS.LIB file system between two IBM i systems. Consider the following information when requesting file transfers of save files.

[“Transferring generation data groups \(GDGs\)” on page 2517](#)

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

[“Using wildcard characters with MFT” on page 2518](#)

You can use wildcard characters when you specify source file names and source file paths for file transfers. This allows you to select multiple files simultaneously.

## **Transferring files and data sets between z/OS and distributed systems**

You can transfer files and supported data set types between z/OS and distributed file systems by using Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

When you transfer a file or data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes passed in the transfer definition. If no attributes are specified, attributes are set to the same as those attributes for the source data set or are set to the default values when the source is a file. The attributes of an existing tape data set are ignored.

### **Transferring from a file to a data set - binary transfers**

The format of the destination data set determines the destination record length. Ensure the data set exists on the destination system or specify the destination data set with the correct attributes so that the data set is created properly. If you do not specify attributes, the system specifies the following default: a physical sequential data set with an undefined record format and the maximum block size (BLKSIZE) for the device (as returned by the DEVTYPE macro). For example, for DASD the size is 6144 and for tape the size is 32760. If you want to transfer a file on a distributed system to a z/OS data set in binary mode, note the following behavior:

#### **Physical sequential (PS) destination data sets:**

- The source file on the distributed system is read sequentially to fill each record or block.
- On variable format data sets, each record is filled to capacity.

#### **Partitioned data set (PDS) destination data sets:**

- Each source file is copied to a PDS member with the same or equivalent name. If the file name is longer than the maximum allowed length of a member name, the file name is converted to a valid member name. For more information about member names, see [Object naming conventions](#). If the source file is a directory, each file in that directory becomes a member of the PDS.
- If a PDS member exists, the member is overwritten if you have specified overwrite existing destination files for the transfer. If you do not specify overwrite, the transfer fails.
- The source file on the distributed system's is read sequentially to fill each record or block for the member.
- On variable format PDS members, each record is filled to capacity.

### **Transferring from a file to a data set - text transfers**

The format of the destination data set determines the destination record length. Ensure the data set exists on the destination system or specify the destination data set with the correct attributes so the data set is created properly. If you want to transfer from a file on a distributed system to a z/OS data set as text, note the following behavior:

#### **Physical sequential (PS) destination data sets:**

- Each line of text becomes a record (or a block for undefined record format (RECFM=U) data sets). End-of-line characters are not present in data set records (for non-ASA data sets only).
- When ASA format control characters are used in the destination data set, end-of-line characters are effectively converted to equivalent ASA format control code.

- When a line is longer than a record, the line is split at the record boundary and flows onto the next record.

#### **PDS destination data sets:**

- Each source file is copied to a PDS member with the same or equivalent name. If the file name is longer than the maximum allowed length of a member name, the file name is converted to a valid member name. For more information about member names, see [Object naming conventions](#). If the source file is a directory, each file in that directory becomes a member of the PDS.
- If a PDS member exists, the member is overwritten if you have specified overwrite existing destination files for the transfer. If you do not specify overwrite, the transfer fails.
- Each line of text becomes a record (or a block for undefined record format (RECFM=U) data sets). End-of-line characters are not present in member records (for non-ASA data sets only).
- When ASA format control characters are used in the destination data set, end-of-line characters are effectively converted to equivalent ASA format control code.
- When a line is longer than a record, the line is split at the record boundary and flows onto the next record.

### **Transferring from a data set to a file - binary and text transfers**

If you want to transfer from a data set to a file as binary or text, note the following behavior:

- The content of each record is transferred in binary form to a file; no record, block format information, or ASA format control characters are transferred.
- For text transfers only, each data set record becomes a line with text converted to the code page of the destination agent. That is, a carriage return-line feed (CRLF) is appended for a Windows destination system and carriage return (CR) is appended for an AIX destination system.
- **Non-VSAM and PS source data sets.** The records for the source data set are transferred to the destination file and concatenated together. If the destination file exists, the file is overwritten, depending on the destination file behavior option you have specified for the file transfer. If the destination is specified as a directory rather than a file, the destination filename will be the data set name excluding the high-level qualifier (HLQ).
- **PDS source data sets.** Each specified member, or all members if no member is specified, is extracted to the destination. If the destination specifies a directory, members are extracted to separate files. Otherwise each specified member is written to the destination file, resulting in effectively only one member being transferred. If the destination file exists for a member, the file is overwritten, depending on the destination file behavior option you have specified for the file transfer.

#### **Related reference**

[“Guidelines for transferring files” on page 2491](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring between data sets on z/OS” on page 2493](#)

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

### **Transferring between data sets on z/OS**

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

Managed File Transfer does not support uncataloged data sets either on disk or tape. Existing data sets must be cataloged and new data sets will be cataloged.

Managed File Transfer can transfer most data set types, with some restrictions which are described below. If a particular data set type, or set of characteristics is not supported, you can use the alternative methods described in [“Using Managed File Transfer in combination with z/OS utilities to transfer files”](#) on page 2496.

Consider the following cases:

**If you copy or move a data set between z/OS systems and the destination does not exist.**

By default, the destination data set is created with the identical characteristics to the source. You can specify attributes for the destination data set to override the default characteristics. If you do this, a compatibility check is performed to ensure the transfer is possible.

**If you copy or move a data set between z/OS systems and the destination already exists.**

If you specify attributes for the destination data set to override the default characteristics, a compatibility check is performed to ensure the destination data set can be accessed in the required way. However, you cannot override the following attributes:

- Base data set organization and type
- Logical record length (LRECL)
- Block size (BLKSIZE)

**If you are transferring a data set to tape.**

When you transfer a data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes passed in the transfer definition. If no attributes are specified, attributes are set to the same as those for the source data set or are set to the default values when the source is a file. The attributes of an existing tape data set are ignored.

In addition, the user identifier that the destination agent is running as needs to have the correct authority to mount tapes. Refer to the documentation for the external security manager being used by your enterprise for information on how to do this.

**If you are transferring from tape to a data set.**

In order to access a dataset on tape, the user identifier that the source agent is running as needs to have the appropriate authority to mount tapes. Refer to the documentation for the external security manager being used by your enterprise for information on how to do this.

## Data set compatibility

Review the following behavior and restrictions for data set compatibility:

**Record format and length differences:**

Variable-format records use a 4 byte record length field in the record data. Therefore for a transfer from a fixed record to a variable record data set, the variable record length must be greater than or equal to the fixed record length plus 4. For a transfer from a variable format record data set to a fixed format record data set, the fixed format record data set record length must be greater than or equal to the variable record length minus 4.

**Block size differences:**

- For fixed- and variable-format record data, block size differences makes the source and destination data set layout different.
- For undefined format records, provided the destination block size is greater or equal to the source data set block size, you can transfer a data set.
- For undefined format data sets, you cannot transfer if the source block size is greater than the destination block size.

**Partitioned data sets (PDS) and partitioned data set extended (PDSE) data sets**

The following behavior and restrictions apply equally to PDS and PDSE:

- When you transfer a PDS or PDSE to a destination PDS or PDSE, the member information and statistics are not preserved. For example, if you transfer a load library that is stored as a PDSE, the destination PDSE is not usable as a load library. See [“Using Managed File Transfer in combination](#)

[with z/OS utilities to transfer files” on page 2496](#) for methods that can be used to transfer the PDSE so that it can be used as a load library.

- If you transfer a PDS or PDSE member to a destination PDS or PDSE, a member of the destination PDS or PDSE is created. If the destination PDS or PDSE member already exists, the member is overwritten. If you transfer a PDS or PDSE member to a non-PDS or non-PDSE destination data set, the destination data set is created to contain the member data. If the destination data set already exists, the data set is overwritten.
- If you attempt to transfer a PDS or PDSE to a non-PDS or non-PDSE destination, this results in all members of the PDS or PDSE being written to the non-PDSE destination. Each subsequent member transfer overwrites the previous contents of the non-PDSE destination or fails, depending on the transfer options.
- When you transfer a PDS or PDSE to a destination PDS or PDSE, a copy of the entire PDS or PDSE is created at the destination. If the destination PDS or PDSE already exists, members from the source are added. If a PDS or PDSE member already exists at the destination, the member is overwritten.
- The transfer of a non-PDS or non-PDSE to a destination PDS or PDSE, adds the contents of the non-PDS or non-PDSE as a new member of the PDS or PDSE. If the PDS member already exists, the member is overwritten. If you do not specify a name for a new member, a name is generated from the source data set or DD name.
- There is a known limitation with transfers to PDS and PDSE data sets on systems where disk space is limited. See [Troubleshooting common MFT problems](#).

#### **V 9.3.0 VSAM data sets**

Managed File Transfer does not support transfers to, or from, VSAM data sets.

#### **V 9.3.0 Sequential data sets**

Managed File Transfer supports logical record lengths (LRECL) only in the range of 4 – 32756 for variable format data sets.

Managed File Transfer supports logical record lengths (LRECL) only in the range of 0 – 32760 for fixed format data sets.

## **Binary and text transfers**

Binary transfer for data sets is defined as the record data in its binary form, as read from the data set using the default record format (type=record). Data is read and written on a record by record basis. The system service performs the necessary record and block conversion (where the data sets have different record and block settings) and the necessary ASA and machine control code conversion. If one data set is defined for ASA format control characters and the other is not appropriate, conversion to normal control codes is performed using the C/C++ system library function behavior.

## **Generation data groups (GDGs)**

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must already exist.

### **Related reference**

[“Guidelines for transferring files” on page 2491](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring generation data groups \(GDGs\)” on page 2517](#)

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

[“Transferring data sets to and from Connect:Direct nodes” on page 2500](#)

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

[“Transferring files and data sets between z/OS and distributed systems” on page 2492](#)

You can transfer files and supported data set types between z/OS and distributed file systems by using Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

## **Using Managed File Transfer in combination with z/OS utilities to transfer files**

The various methods you can use with Managed File Transfer (MFT) to transfer a wide range of data sets between z/OS systems.

You can use these methods for all data sets that MFT supports, but these methods are particularly useful when used to transfer data sets for which MFT supports with restrictions, or has no support for at all. These approaches work for all supported versions of MFT.

For example, these methods can be used to transfer PDSE data sets between systems without losing directory information.

Each of the methods use the same general approach of using JCL to:

- Run one or more z/OS utilities to convert the source data set into a format that MFT can transfer.
- Schedule MFT to transfer the converted data set to the target system, and wait until the transfer is complete.
- Schedule JCL on the target system to run one or more z/OS utilities to convert the converted data set into a target data set that is the same as the original source data set.

As well as the methods described in this topic, there is an alternative approach described in [vsamtransfer](#), which describes how Ant tasks can be used to run commands before and after a transfer to do a similar thing. While the sample demonstrates the transfer of VSAM data sets, the approach can be extended to other data set types, subject to the limitations of the [REPRO](#) command.

### **Method 1: Using the TRANSMIT (XMIT) and RECEIVE commands with MFT**

This method uses the [TRANSMIT \(XMIT\)](#) TSO command to convert a data set into a sequential data set, and transfer it using MFT. Once the transfer is complete the sequential data set is converted back into the original data set type using the [RECEIVE](#) command.

This method can be used with any data set supported by the XMIT command. A list of supported data sets, and attributes are listed in [Transmitting data sets](#). For example, this method can be used to transfer PDSEs while preserving directory information, but it cannot be used to transfer VSAM data sets.

This method is implemented using two JCL jobs and you need to adjust these jobs so that they are suitable for your environment, and the type of data being transferred. You need to change the values inside `< >`. In most environments extra job steps need to be added to delete earlier versions of the data sets, or alternatively you can use generation data groups.

You submit the first of these jobs, XMITJOB1 shown in the following example, on the sending side.

The XMIT step runs the XMIT command to convert the source data set into a sequential format data set. X.X is specified for the node and user name to pass the command validation checks, but a proper node and user name are not needed.

The MFT step initiates a file transfer from the source agent, SRC, to the destination agent, DEST. The **-w** flag means that the `fteCreatetransfer` command waits until the transfer has completed. The **-ds** flag indicates that a sequential data set is to be created on the destination agent and provides the correct DCB characteristics, so that there is sufficient space when the data set is dynamically allocated.



In this case, both data set names are surrounded with double quotes, indicating that fully qualified data set names are used. If double quotes are not used, the default high level qualifier of the source or destination agent is used.

The SUBMIT step only runs if the MFT step successfully completes. This step submits the RECVJOB1 job which restores the transferred data set to its original format on the destination system.

### Example XMITJOB1 JCL

```
//XMITJOB1 JOB NOTIFY=&SYSUID
//*
//*****
//* Use the XMIT command to unload the data set to fix block,
//* 80 logical record format
//*****
//XMIT EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
XMIT X.X DSN('USER1.SOURCE.DATASET')          +
OUTDATA('USER1.SOURCE.DATASET.UNLOADED')
/*
//*****
//* Invoke MFT fteCreateTransfer
//*****
//MFT EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD SYSOUT=*
//STDOUT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH <MFT path>/fteCreateTransfer      +
-w                                             +
-sa SRC
+
-da DEST
+
-ds "'USER1.TARGET.DATASET.UNLOADED';        +
RECFM(F,B);BLKSIZE(3120);LRECL(80);SPACE(10,10); +
CYL;RELEASE"                                  +
"'"USER1.SOURCE.DATASET.UNLOADED'"
/*
//*****
//* Submit the restore job to the internal reader
//
//*****

//SUBMIT EXEC PGM=IEBGENER,COND=(0,NE)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=USER1.JCL.MFT(RECVJOB1)
//SYSUT2 DD SYSOUT=(A,INTRDR),DCB=BLKSIZE=80
//SYSIN DD DUMMY
```

The RECVJOB1 JCL is shown in the following example. When it is submitted by XMITJOB1, it is routed by JES2 to the target node as indicated on the ROUTE command on the second line of the job. Depending on the settings of your installation, you might need to provide USER and PASSWORD parameters on the JOB step.

The RECEIVE step takes the data set that has been transferred by MFT and uses the TSO RECEIVE command to convert it back into its original format.

### Example RECVJOB1 JCL

```
//RECVJOB1 JOB NOTIFY=&SYSUID
/*ROUTE XEQ NODE2
//*
//*****
//* Convert the data set back into its original format
//*****
//RECEIVE EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//UNLOAD DD DISP=SHR,DSN='USER1.TARGET.DATASET.UNLOADED'
//SYSTSIN DD *
```

```
RECEIVE INFILE (UNLOAD)
DSN ('USER1.TARGET.DATASET')
/*
```

## Method 2: Using the ADDRSSU utility with MFT

This method uses the DUMP and RESTORE commands of the ADDRSSU utility to convert data sets to and from a format that MFT can transfer. This method can be used with a wider range of data sets than method one, including VSAM data sets, and for transfer of multiple data sets at the same time.

Information on data sets that are not supported with DUMP is described in [Special considerations for DUMP](#).

As before, this method is implemented using two JCL jobs and you need to adjust these jobs so that they are suitable for your environment, and the type of data being transferred. You need to change the values inside < >. In most environments extra job steps need to be added to delete earlier versions of the data sets, or alternatively you can use generation data groups.

You submit the first of these jobs, DUMPJOB1 shown in the following example, on the sending side.

The DUMP step runs the ADDRSSU DUMP command to convert the source data set into a sequential data set. This step can be adjusted to dump multiple data sets if needed.

The XMIT step converts the dumped data set into a fix block, 80 logical record format. This step is not strictly necessary but provides consistency with the approach used in XMITJOB1. X.X is specified for the node and user name to pass the command validation checks, but a proper node and user name are not needed.

The MFT step initiates a file transfer from the source agent, SRC, to the destination agent, DEST. The **-w** flag means that the `fteCreatetransfer` command waits until the transfer has completed. The **-ds** flag indicates that a sequential data set is to be created on the destination agent and provides the correct DCB characteristics, so that there is sufficient space when the data set is dynamically allocated.

In this case, both data set names are surrounded with double quotes, indicating that fully qualified data set names are used. If double quotes are not used, the default high level qualifier of the source or destination agent is used.

The SUBMIT step only runs if the MFT step successfully completes. This step submits the RESTJOB1 job which restores the transferred data set to its original format on the destination system.

### Example DUMPJOB1 JCL

```
//DUMPJOB1 JOB NOTIFY=&SYSUID,REGION=0M
/*
//*****
//* Invoke ADDRSSU to unload the selected data sets
//
//*****

//DUMP EXEC PGM=ADDRSSU,REGION=2048K
//SYSPRINT DD SYSOUT=*
//DUMPDD DD DSN=USER1.SOURCE.DATASET.BACKUP,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(CYL,(200,100,0),RLSE)
//SYSIN DD *
DUMP DATASET(INCLUDE(USER1.SOURCE.DATASET)) -
OPTIMIZE(4) OUTDDNAME(DUMPDD) TOLERATE(ENQF)
/*
//*****
//* Convert the contents to fix block, 80 logical record format
//
//*****

//XMIT EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//DUMPDD DD DISP=SHR,DSN=USER1.SOURCE.DATASET.BACKUP
//XMITDD DD DISP=(,CATLG),DSN=USER1.SOURCE.DATASET.BACKUP.UNLOAD,
// DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120),
// UNIT=SYSDA,SPACE=(CYL,(200,100,0),RLSE)
//SYSTSIN DD *
```

```

XMIT X.X DDNAME(DUMPDD) +
OUTDD(XMITDD)
/*
//*****
//* Invoke MFT fteCreateTransfer
//
*****

//MFT EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD SYSOUT=*
//STDOUT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH <MFT path>/fteCreateTransfer      +
-w                                             +
-sa SRC                                       +
-da DEST                                       +
-ds "'USER1.TARGET.DATASET.BACKUP.UNLOAD';   +
RECFM(F,B);BLKSIZE(3120);LRECL(80);SPACE(50,50); +
CYL;RELEASE;UNIT(SYSDA)"                    +
"'"USER1.SOURCE.DATASET.BACKUP.UNLOAD'"
/*
//*****
//* Submit the restore job to the internal reader
//
*****

//SUBMIT EXEC PGM=IEBGENER,COND=(0,NE)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=USER1.JCL.MFT(RESTJOB1)
//SYSUT2 DD SYSOUT=(A,INTRDR),DCB=BLKSIZE=80
//SYSIN DD DUMMY

```

The RESTJOB1 JCL is shown in the following example. When the job is submitted by DUMPJOB1, it is routed by JES2 to the target node as indicated on the ROUTE command on the second line of the job. Depending on the settings of your installation, you might need to provide USER and PASSWORD parameters on the JOB step.

The RECEIVE step takes the data set that has been transferred by MFT and uses the TSO RECEIVE command to convert it back into the format expected by the ADRDSSU RECEIVE command.

The RESTORE step then uses ADRDSSU RECEIVE to convert the data set into its original format. The RENAMEU parameter could be used here to change the data set prefixes if needed.

### Example RESTJOB1 JCL

```

//RESTJOB1 JOB NOTIFY=&SYSUID,REGION=0M
//*
//*****
//* Convert the data set back into the form accepted by
//* RECEIVE
//*****
//RECEIVE EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//UNLOAD DD DISP=SHR,DSN=USER1.TARGET.DATASET.BACKUP.UNLOAD
//SYSTSIN DD *
RECEIVE INFILE(UNLOAD)
DSN('USER1.TARGET.DATASET.BACKUP')
/*
//*****
//* Convert the data set back into its original format
//
*****

//RESTORE EXEC PGM=ADRDSSU,REGION=2048K
//SYSPRINT DD SYSOUT=*
//DUMPDD DD DISP=SHR,DSN=USER1.TARGET.DATASET.BACKUP
//SYSIN DD *
RESTORE DATASET(INCLUDE(**)) -
INDDNAME(DUMPDD) -
CATALOG
/*

```

## Related reference

[“Guidelines for transferring files” on page 2491](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring files and data sets between z/OS and distributed systems” on page 2492](#)

You can transfer files and supported data set types between z/OS and distributed file systems by using Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

## Transferring data sets to and from Connect:Direct nodes

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

### Specifying data set names

To specify a data set on a Connect:Direct node in a transfer request, use the syntax that is used for data set transfers between Managed File Transfer agents, but with two changes:

- You must prefix the data set name with the Connect:Direct node name and a colon (:). The syntax is as follows:

```
cdNode:data_set_name{;attrib1;...;attribN}
```

For example, to specify a partitioned data set called OBJECT.LIB on the system where the Connect:Direct node CD\_NODE1 is located, use the following syntax:

```
CD_NODE1:// 'OBJECT.LIB';RECFM(F,B);BLKSIZE(800);LRECL(80)
```

In this example, three optional attributes are specified by the text RECFM(F,B);BLKSIZE(800);LRECL(80).

- The specified data set name is interpreted as a fully qualified data set name, regardless of whether it is enclosed by single quotation mark characters. The system never adds any prefix. If you want to specify a prefix, such as the user ID that the agent runs under, you must specify it as part of the data set name. This differs from the behavior for data set transfers that involve only Managed File Transfer agents, where if the specified data set name is not enclosed by single quotation mark characters, the system adds a prefix of the default high-level qualifier for the destination agent.

Except for these two changes, specify the data set name and any optional attributes using the same syntax that is used for data set transfers between Managed File Transfer agents, which has the following rules:

- You must prefix the data set name with two forward slash characters (//).
- If you want to specify data set attributes, provide these after the data set name, separated by semicolons. Attributes must be provided in the format *key(value)*, which is suitable for BPXWDYN.

For more information about specifying data sets in a transfer request, see [“fteCreateTransfer \(start a new file transfer\)” on page 2048](#) and [“fteCreateTemplate \(create new file transfer template\)” on page 2033](#).

### Parameters to use in your transfer request

For most transfer requests that involve data sets on Connect:Direct nodes, you can specify the source and destination data sets in the same way as you would for a data set transfer that involves only Managed File Transfer agents. Use the **source\_specification**, **-ds**, and **-dp** parameters with the **fteCreateTransfer** or **fteCreateTemplate** commands.

**Note:** If the destination of the transfer is a PDS and the destination agent is the Connect:Direct bridge agent, you must specify the **-de** parameter with the value of `overwrite`.

## Specifying data set attributes

Certain data set attributes are set by Managed File Transfer and passed through as parameters to the Connect:Direct **COPY** process. You can also supply certain attributes in the transfer request, by specifying the appropriate BPXWDYN key. The Connect:Direct bridge converts keys that have equivalent Connect:Direct properties to the format that is required by Connect:Direct. For example, in the data set specification `CD_NODE1: // 'OBJECT.LIB' ;RECFM(F,B) ;BLKSIZE(800) ;LRECL(80)`, the attributes `RECFM(F,B) ;BLKSIZE(800) ;LRECL(80)` are converted to `DCB=(RECFM=FB, BLKSIZE=800, LRECL=80)`.

For details of the mappings between these two types of parameter, including details of the BPXWDYN keys that are supported for use with a Connect:Direct transfer, see [“Mappings between Connect:Direct process statement parameters and BPXWDYN keys” on page 2501](#). Not all BPXWDYN keys have an equivalent Connect:Direct process parameter, and not all Connect:Direct process parameters have an equivalent BPXWDYN key.

## Additional considerations

- If your transfer destination is a partitioned data set at a Connect:Direct node, you must create the partitioned data set before the transfer, because the Connect:Direct node does not create it for you.

### Related concepts

[Connect:Direct file paths specified with a double forward slash](#)

### Related tasks

 [Transferring a data set to a Connect:Direct node on z/OS](#)

### Related reference

The [Connect:Direct bridge](#)

[“Transferring between data sets on z/OS” on page 2493](#)

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[“fteCreateTemplate \(create new file transfer template\)” on page 2033](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use.

The only required parameter is the **-tn** *template\_name* parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

## Mappings between Connect:Direct process statement parameters and BPXWDYN keys

When you submit a transfer request for a data set where either the source or destination is a Connect:Direct node, any supported BPXWDYN keys that you provide are converted to a format that is accepted by Connect:Direct processes.

For more information about IBM Sterling Connect:Direct process statements, download the [Connect:Direct Process Language Reference Guide](#).

Table 359. Parameters to the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Parameter to Connect:Direct COPY statement	BPXWDYN key
DSN	DSN (valid for transfers to and from data sets). Specifying this key overrides the parameter value that is assigned by Managed File Transfer, which is based on the source or destination file specifications that are provided in the transfer request.
FILE	No mapping for data sets.
PNODE	No mapping. The primary node for the transfer is identified by Managed File Transfer. If you attempt to provide a value for this parameter, an error is produced.
SNODE	No mapping. The secondary node for the transfer is identified by Managed File Transfer. If you attempt to provide a value for this parameter, an error is produced.
DCB	See <a href="#">Mappings for subparameters of DCB</a>
DISP	See <a href="#">Mappings for subparameters of DISP for a COPY From statement</a> and <a href="#">Mappings for subparameters of DISP for a COPY To statement</a>
RESGDG	No mapping
LABEL	See <a href="#">Mappings for subparameters of LABEL</a>
MSVGP	No mapping
UNIT	UNIT
VOL	See <a href="#">Mappings for subparameters of VOL</a>
ALIAS	No mapping
EXCLUDE	No mapping
PDS.DIR	No mapping. Managed File Transfer sets the value of this process parameter to N, so no user-related information that is in the directory is sent.
REPLACE   NOREPLACE	No BPXWDYN equivalent. The behavior when a destination data set already exists on the destination system is defined by the value of the <b>-de</b> (destination_file_behavior) parameter in the transfer request. For more information about the default behavior of Managed File Transfer when a destination data set already exists, see <a href="#">“Transferring between data sets on z/OS” on page 2493</a> .
SELECT	No BPXWDYN equivalent. The data set members that are selected for copying are defined by the source file specification in the transfer request.
BUFND	No mapping
IOEXIT	No mapping
DATAEXIT	No mapping

Table 359. Parameters to the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer (continued)

Parameter to Connect:Direct COPY statement	BPXWDYN key
SYSOPTS	See <a href="#">Mappings for subparameters of SYSOPTS</a>
TYPE	No mapping
AVGREC	No mapping
DATACLAS	DATACLAS
DSNTYPE	DSNTYPE. Specifying a value of PDS for this key overrides the parameter value that is assigned by Managed File Transfer, which is LIBRARY. There are no mappings for any other value - EXTPREF, EXTREQ, BASIC, or LARGE. Specifying any of these unsupported values produces an error. Specifying PDS or LIBRARY for a sequential data set produces an error.
KEYLEN	No mapping
KEYOFF	No mapping
LIKE	LIKE
LRECL	No mapping
MGMTCLAS	MGMTCLAS
RECORG	No mapping
SECMODEL	No mapping
STORCLAS	STORCLAS
SPACE	See <a href="#">Mappings for subparameters of SPACE</a>
SYSOUT	No mapping
CKPT	No mapping
COMPRESS	No mapping
SECURE	No mapping

Table 360. Subparameters of the DCB parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the DCB parameter	BPXWDYN key
model-file-name	No mapping
BLKSIZE	BLKSIZE
NCP	BUFNO
DEN	No mapping
DSORG	DSORG
KEYLEN	No mapping
LIMCT	No mapping
LRECL	LRECL
OPTCD	No mapping

Table 360. Subparameters of the **DCB** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer (continued)

Subparameters of the DCB parameter	BPXWDYN key
RECFM	RECFM
RKP	No mapping
TRTCH	TRTCH

Table 361. Subparameters of the **DISP** parameter for the Connect:Direct **COPY From** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the DISP parameter for a COPY From statement	BPXWDYN key	Details
[OLD   SHR]	[OLD   SHR]	Specifies the status of the data set before the transfer. Managed File Transfer sets this subparameter to <b>SHR</b> .
[KEEP   DELETE]	[KEEP   DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed successfully. The value set by Managed File Transfer depends on the source file disposition, defined by the <b>-sd</b> parameter.
[KEEP   DELETE]	[KEEP   DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed abnormally. Managed File Transfer sets this subparameter to <b>KEEP</b> .

Table 362. Subparameters of the **DISP** parameter for the Connect:Direct **COPY To** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the DISP parameter for a COPY To statement	BPXWDYN key	Details
[NEW   OLD   MOD   RPL   SHR]	[NEW   OLD   MOD   SHR]	Specifies the status of the data set before the transfer. The value set by Managed File Transfer depends on the value of the <b>-de</b> (destination_file_behavior) parameter in the transfer request. If the destination data set does not already exist, the subparameter value is <b>NEW</b> . If the data set already exists, the subparameter value is <b>RPL</b> . Managed File Transfer does not support the key <b>RPL</b> being provided in a transfer request.
[KEEP   CATLG]	[KEEP   CATLOG] or PATHDISP	Specifies the status of the data set after the transfer has completed successfully. Managed File Transfer sets this subparameter to <b>CATLOG</b> .



Table 362. Subparameters of the **DISP** parameter for the Connect:Direct **COPY To** statement, and the equivalent BPXWDYN keys used by Managed File Transfer (continued)

Subparameters of the <b>DISP</b> parameter for a <b>COPY To</b> statement	BPXWDYN key	Details
[KEEP   CATLG   DELETE]	[KEEP   DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed abnormally. Managed File Transfer sets this subparameter to <b>DELETE</b> .

Table 363. Subparameters of the **LABEL** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the <b>LABEL</b> parameter for a <b>COPY</b> statement	BPXWDYN key	Details
file-sequence-number	SEQUENCE	
[SL   AL   BLP   LTM   NL]	LABEL( <i>type</i> )	The possible values of <i>type</i> are NL, SL, NSL, SUL, BLP, LTM, AL, and AUL. Connect:Direct accepts a subset of these values. If you specify a value that is not supported by Connect:Direct, Connect:Direct produces an error message.
[PASSWORD   NOPWREAD]	No mapping	
[IN   OUT]	No mapping	
[RETPD   EXPDT]	RETPD	EXPDT not supported

Table 364. Subparameters of the **VOL** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the <b>VOL</b> parameter for a <b>COPY</b> statement	BPXWDYN key
PRIVATE	No mapping
RETAIN	No mapping
volume-sequence-no	No mapping
volume-count	MAXVOL
SER	VOL
REF	No mapping

Table 365. Subparameters of the **SYSOPTS** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the <b>SYSOPTS</b> parameter for a <b>COPY</b> statement	BPXWDYN key
DBCS	No mapping


Table 365. Subparameters of the **SYSOPTS** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer (continued)

Subparameters of the <b>SYSOPTS</b> parameter for a <b>COPY</b> statement	BPXWDYN key
CODEPAGE	Value is dependent on Managed File Transfer transfer options. For more information, see <a href="#">“Transferring text files with MFT”</a> on page 2507.
DATATYPE	No mapping. Managed File Transfer sets this value to TEXT for text transfers to or from a data set, and otherwise to BINARY.
XLATE	No mapping. Managed File Transfer sets this value to NO when the value of <b>DATATYPE</b> is TEXT.
STRIP.BLANKS	No mapping. Managed File Transfer sets this value to YES when the value of <b>DATATYPE</b> is TEXT.
PERMISS	No mapping
PRECOMP	No mapping
UNIQUE	No mapping
SYSOUT	No mapping

Table 366. Subparameters of the **SPACE** parameter for the Connect:Direct **COPY** statement, and the equivalent BPXWDYN keys used by Managed File Transfer

Subparameters of the <b>SPACE</b> parameter for a <b>COPY</b> statement	BPXWDYN key
CYL	CYL
TRK	TRACKS
blk	BLOCKS
av-rec-len	No mapping
prim, [sec], [dir]	SPACE(prim[,sec]), DIR
RLSE	RELEASE
CONTIG	No mapping
ROUND	No mapping

### Related tasks

 [Transferring a data set to a Connect:Direct node on z/OS](#)

### Related reference

[Transferring data sets to and from Connect:Direct nodes](#)

You can transfer data sets between Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

[The Connect:Direct bridge](#)

## BPXWDYN properties you must not use with MFT

Some BPXWDYN options must not be specified when using the **fteCreateTemplate** command, the **fteCreateTransfer** command or the **bpxwdynAllocAdditionalOptions** property in the `agent.properties` file.

There are a number of BPXWDYN options that must not be specified with Managed File Transfer because they are used by the agent or they are not supported. If you use these options they can cause unpredictable behavior; the options are listed in the following table.

BPXWDYN options	Description
DA DSN	Specifies the data set name to allocate.
FI DD	Specifies the ddname to allocate.
FILEDATA	Specifies, to the sequential access method services, whether the data is treated as text or binary.
OLD SHR MOD NEW SYSOUT	Specifies the data set status.
REUSE	Specifies that the named data set is freed before the function is performed.
HOLD	Specifies that the output data set is to be held until released by the user or operator.
KEEP DELETE CATALOG UNCATALOG	Specifies the data set disposition after it is freed.
RECORG(LS)	Creates a VSAM linear data set.
MSG	Directs allocation messages. <b>Note:</b> This option can be used, but because Managed File Transfer uses this option to direct error information to the transfer log, using it can cause unpredictable behavior.

### Related reference

[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[“fteCreateTemplate \(create new file transfer template\)” on page 2033](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn template\_name** parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

[The agent.properties file](#)

### Transferring text files with MFT

Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of Managed File Transfer.

Unless you specify otherwise, conversion is from the default code page of the file's source system to the default code page of its destination system. Additionally, text file transfer performs new line conversion, which means that new line characters for the destination file are those native to its destination platform. You can override the use of the default code pages on a system by specifying the code page to use for reading the source file and writing the destination file. You can also specify the end-of-line character sequence to use for the destination file. For more information, see the topics [“fteCreateTransfer \(start a new file transfer\)” on page 2048](#).

Text file transfers perform simple code point substitutions between code pages. Text file transfers do not perform complex transfers or translations of data, for example, conversions between visual and logical forms of bidi data or text shaping.

<i>Table 368. Text file transfer behavior for all platforms</i>		
<b>Area</b>	<b>Default behavior</b>	<b>Can you change this behavior?</b>
Source file encoding	Source platform encoding	Yes  When you specify source file encoding and the source is a data set, the encoding must be an EBCDIC code page, otherwise the transfer fails. Similarly, if the destination is a data set, the destination encoding must be an EBCDIC code page.
Source file end of line character sequence	Convert a single (LF) or (CRLF) sequence to the destination end of line character sequence	No
Destination file encoding	Destination platform encoding	Yes  When you specify source file encoding and the source is a data set, the encoding must be an EBCDIC code page, otherwise the transfer fails. Similarly, if the destination is a data set, the destination encoding must be an EBCDIC code page.
Destination file end of line character sequence	Destination platform EOL	Yes
Text replacement character sequence for unmappable or malformed characters in the source or destination	Blank, meaning the transfer fails if unmappable characters or malformed characters are present. You can use the <code>textReplacementCharacterSequence</code> property to specify the replacement text, which is described in <a href="#">The agent.properties file</a> .	Yes

## **z/OS data sets**



When data set records are accessed in text mode, each record represents a single line. New line characters do not exist in the record but for ASA format data sets an ASA format control code character is set that represents a new line (or other control character). When a line of text with a terminating new line character is written to a record, the new line character is either automatically removed or an appropriate ASA control code is set, as appropriate. When a record is read a new line character is automatically

appended to the return data. For ASA format data sets this character can be multiple new lines or a form feed, as appropriate for the ASA control code of the record.

Additionally, for fixed-format data sets when a record is read the new line is appended after the last character in the record that is not a space character, thus making fixed-format data sets suitable for storing text.

Area	Default behavior	Can you change this behavior?
Maximum line length	Destination data set LRECL or BLKSIZE setting, as appropriate	No
Wrap over length lines	Wrap. The line is split over multiple records and blocks as required.	No

When the Managed File Transfer agent is run, the environment variable `_EDC_ZERO_RECLLEN` is always set to "Y". This setting makes Managed File Transfer text transfer behavior the same as FTP for variable and fixed block data sets. However, for undefined format data sets, Managed File Transfer converts single space lines to an empty line and preserves empty lines. FTP converts empty lines to single space lines and preserves single space lines. Table 3 describes the Managed File Transfer behavior and how FTP behavior differs.

The format of the data set also determines how each line of text is written to a record. For non-ASA format data sets newline and carriage-return characters are not written to the record. For ASA format data sets, the first byte of each record is an ASA control code representing end of lines, a form feed, and other codes, as appropriate. Because ASA control codes are at the start of each record, if the source text file does not start with a new line character sequence, a blank ( ' ') ASA control character sequence (which equates to a newline) is inserted. This means that if the ASA data set is transferred to a file, a blank line is present at the start of the file.

Data set format	Original text line in file	Data set record	Read of data set record	FTP Read behavior
Fixed block	Empty line	Space filled record	Empty line	Same as MFT
Fixed block	Single space	Space filled record	Empty line	Same as MFT
Variable block	Empty line	Empty record	Empty line	Same as MFT
Variable block	Single space	Single space record	Single space	Same as MFT
Undefined	Empty line	Single space record	Empty line	Single space
Undefined	Single space	Single space record	Empty line	Single space

### Related tasks

[“Transferring text files between Connect:Direct and MFT” on page 2510](#)

Text transfer involves converting the text of a file from one code page to another, and converting CRLF (carriage return line feed) characters between systems.

### Related reference

[“Guidelines for transferring files” on page 2491](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Available code pages for MFT” on page 2554](#)

This reference topic lists all character encoding formats available for text file conversion on the various platforms supported by Managed File Transfer.

## Transferring text files between Connect:Direct and MFT

Text transfer involves converting the text of a file from one code page to another, and converting CRLF (carriage return line feed) characters between systems.

### About this task





Ensure that the network map of the Connect:Direct bridge node and any Connect:Direct nodes that are used as a transfer destination include the correct platform description.

For information about the behavior of text transfers in Managed File Transfer, see [“Transferring text files with MFT” on page 2507](#).

### Procedure

- For each remote node in your network map, select the correct operating system value.

For example, if your Connect:Direct bridge node is on a Windows system, ensure that for each remote node in your network map you select the correct value from the **Operating System** list:

-  If the remote node is on a Windows system, select Windows.
-   If the remote node is on an AIX or Linux system, select UNIX.
-  If the remote node is on a z/OS system, select OS/390.

Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.

- For each remote node you transfer a file to or from, specify the operating system type of the remote Connect:Direct node in the `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory.

For more information, see [Configure the ConnectDirectNodeProperties.xml file to include information about the remote Connect:Direct nodes](#) and [Connect:Direct node properties file format](#).

## Transferring files to or from protocol bridge agents

You can transfer files to and from an FTP or SFTP file server outside your Managed File Transfer network using a protocol bridge agent.

When you transfer files using the protocol bridge, the bridge must have permission to read the source or destination directory containing the files you want to transfer. For example, if you want to transfer files from the directory `/home/fte/bridge` that has only execute permissions (`d--x--x--x`), any transfers you attempt from this directory fail with the following error message:

```
BFGBR0032E: Attempt to read filename from the protocol file server has failed with server error 550
Failed to open file.
```

During file transfer, files are typically written as temporary files at the destination and are then renamed when the transfer is complete. However, if the transfer destination is a protocol file server that is configured as limited write (users can upload files to the protocol file server but cannot change those uploaded files in any way; effectively users can write once only), transferred files are written to the destination directly. This means that if a problem occurs during the transfer, the partially-written files remain on the destination protocol file server and Managed File Transfer cannot delete or edit these files. In this situation the transfer fails.

Ensure that you have another agent in your Managed File Transfer network in addition to the protocol bridge agent. The protocol bridge agent is a bridge to the FTP or SFTP server only and does not write transferred files to the local disk. If you want to transfer files to or from the FTP or SFTP server you must

use the protocol bridge agent as the destination or source for the file transfer (representing the FTP or SFTP server) and another standard agent as the corresponding source or destination.

## Managed transfer requests that require a new directory to be created on an SFTP file server

Managed File Transfer protocol bridge agents use the third-party JSch library to communicate with file servers using the SFTP protocol. If the protocol bridge agent attempts to transfer a file into a directory that does not exist on a file server and JSch is unable to perform the requested SFTP operation to create that directory, because the user that the protocol bridge agent logs into the file server with does not have permission to do so, JSch throws an exception back to the protocol bridge agent. The protocol bridge agent then marks the managed transfer as "Failed" and generates a supplementary message. If JSch has provided more information about the failure, the protocol bridge agent includes this information in the supplementary message:

```
BFGTR0072E: The transfer failed to complete due to the exception:  
BFGBR0119E: Bridge agent was unable to create directory directory name because message from JSch  
exception
```

From IBM MQ 9.1.0 Fix Pack 5 and IBM MQ 9.1.5, if the JSch exception does not contain any more information about the failure, the protocol bridge agent generates the following supplementary message:

```
BFGTR0072E: The transfer failed to complete due to the exception:  
BFGBR0209E: Bridge agent was unable to create directory directory name
```

### Related reference

[The protocol bridge](#)

## Transferring files to or from IBM i systems

If you transfer files to or from IBM i systems using Managed File Transfer in text mode and you want to convert the data in the files, consider the information in this topic.

Each file on an IBM i system is tagged with a coded character set ID (CCSID) value that identifies the data encoding of the file. For example, a file containing EBCDIC data might have a CCSID value of 037 and a file containing ASCII data might have a CCSID value of 819.

For text mode transfers, Managed File Transfer converts data when there are file encoding differences between source and destination files. However, Managed File Transfer currently ignores CCSID tags associated with files on IBM i systems. Instead, it uses the JVM file encoding property of the JVMs running the source agent and destination agent. The default value of this property is based on locale (but you can override this default on your IBM i system using the `SystemDefault.properties` file described in the following section: [“Changing the file.encoding record in the SystemDefault.properties file” on page 2512](#)). With this default implementation, an agent that transfers files in text mode is limited in its ability to handle text files with different file encodings. For example, you cannot use the same agent to transfer files containing EBCDIC text and also files containing ASCII text without stopping and restarting the agent with the appropriate (that is, EBCDIC or ASCII) file encoding override in place. On IBM i V6R1 systems, you can check the file encoding value of the JVM that is running the agent job by using WRKJVMJOB, option 7 to Display Current® Java System Properties. (The WRKJVMJOB command does not exist on IBM i V5R4 systems.)

If you plan to use Managed File Transfer to transfer text files with different file encodings, consider creating multiple agents and multiple users who start those agents, so that each unique encoding has an agent that is ready and enabled to transfer that type of data.

For example, if you want to transfer a file containing EBCDIC text with CCSID value of 037 from an IBM i system (source) to another IBM i V6R1 system (destination) where you want the file content at the destination to be converted to ASCII text with CCSID value of 819, complete the following steps:

1. Select a source agent with a JVM file encoding of Cp037.
2. Select a destination agent with a JVM file encoding of ISO8859\_1.
3. Select text mode transfer, and other specifications as needed.

## Changing the file.encoding record in the SystemDefault.properties file

To enable a JVM running an agent for a particular encoding, complete the following steps:

1. Determine which user starts the agent that runs on the IBM i system. This is the agent that services the Managed File Transfer file transfer request.

Create a `SystemDefault.properties` file in the home directory of that user as needed. For example, if you start the agent, use Qshell to run the following command:

```
touch -C 819 /home/your_userID/SystemDefault.properties
```

2. Using Qshell, run the `/qibm/proddata/mqm/bin/fteStopAgent` command to stop the agent as needed.
3. Update the `SystemDefault.properties` file that is described in step 1 to ensure that the file contains a record like the following:

```
file.encoding=java_encoding
```

where *java\_encoding* corresponds to the type of data that is contained in the file, and matches a `file.encoding` value from the following table: [File.encoding values and System i5® CCSID](#).

4. The user identified in step 1 must complete the following steps:
  - a. On IBM i V5R4 only: add the `QIBM_PASE_DESCRIPTOR_STDIO` environment variable (\*JOB scope) to 'B' if using EBCDIC file encoding, or 'T' if using ASCII encoding. For example:

```
ADDENVVAR ENVVAR('QIBM_PASE_DESCRIPTOR_STDIO') VALUE('B') REPLACE(*YES)
```

- b. If Qshell is active, press **F3=Exit** to end Qshell.
- c. Start Qshell and run the `/qibm/proddata/mqm/bin/fteStartAgent` command as appropriate to restart the agent.

When the file encoding of the JVM running the agent has been changed, the agent log is written with that encoding. If you want to read the contents of the agent log, you must use a viewer that is enabled for that encoding.

## Using a transfer definition for data conversion

An alternative way to convert data when files are being transferred is to create a transfer definition that specifies file encoding, or to use the `-sce` and `-dce` parameters of the `fteCreateTransfer` command. If you use these parameters when the destination is an IBM i system, this can result in files that have incorrect CCSID tags. For this reason, the recommended approach for controlling data conversion with files that are located on IBM i systems is to use `SystemDefault.properties` as described in the preceding section.

## Protocol bridge limitation

On IBM i, you cannot transfer EBCDIC files to or from an SFTP server using a protocol bridge agent.

### Related tasks

[Installing IBM MQ server on IBM i](#)

### Related reference

[“Guidelines for transferring files” on page 2491](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring save files located in QSYS.LIB on IBM i” on page 2515](#)



Managed File Transfer supports the transfer of save files located in the QSYS.LIB file system between two IBM i systems. Consider the following information when requesting file transfers of save files.

## **IBM i** Transferring physical files located in QSYS.LIB on IBM i

Managed File Transfer supports the transfer of physical file members in the QSYS.LIB file system between two IBM i systems. Consider the following information when you request file transfers of physical file members.

This support is limited to transferring file members in program-described files only and does not support the use of externally-described files or source physical files. You can transfer file members to a destination file member on another IBM i system, or to a stream file residing in an IBM i system, or other platforms, such as Windows or AIX. You can also transfer stream files to a destination file member.

When transferring to a file that does not exist, a program-described file is created with a record length of 5000. There is currently no support for specifying the record length, CCSID, or other attributes for creating the file during the transfer. If you want to specify a value or attribute, you must create the destination file before the transfer occurs, although you can also do this using a predestination transfer task.

You can transfer file members in text mode only. The data is automatically converted from EBCDIC.

A physical file member on IBM i is located in a physical file, which in turn is located in a library on IBM i. A library can be one of the standard libraries that ship with the operating system (for example, QSYS or QGPL) or it can be a library that you have created.

Physical files in the QSYS.LIB file system are identified in two different ways on IBM i. When you run CL commands on an IBM i command line, use the following naming syntax:

```
FILE(library name/file name) MBR(member name)
```

For example, a physical file member that is called MYMBR is in a file that called MYFILE in a library that is called SOMELIB is identified as FILE(SOMELIB/MYFILE) MBR(MYMBR). You can also identify the same physical file member by specifying a UNIX-like path name that follows the Integrated File System (IFS) naming convention. Using the IFS naming convention, MYMBR in MYFILE in SOMELIB has the following path name:

```
/QSYS.LIB/SOMELIB.LIB/MYFILE.FILE/MYMBR.MBR
```

For more information, see [Path names in the QSYS.LIB file system](#).

Managed File Transfer on IBM i recognizes the IFS naming convention but does not support the syntax used by CL commands. The following examples illustrate valid and invalid path names for MFT. The following example is a valid path name for a physical file member:

```
/QSYS.LIB/SOMELIB.LIB/MYFILE.FILE/MYMBR.MBR
```

This example assumes MYFILE is a physical file in the library SOMELIB and contains a member that is called MYMBR.

The following examples are invalid path names for physical file member transfers:

- /QSYS.LIB/SOMELIB.LIB/MYFILE.FILE (.FILE assumes a SAVF, not a physical file. If MYFILE is a physical file, the transfer fails with an invalid file type error)
- /QSYS.LIB/MYLIB.LIB/ (physical file and member names are required)
- /QSYS.LIB/SOMELIB.LIB/MYFILE.FILE/MYMBR (the member name must contain an extension of .MBR)
- /QSYS.LIB/SOMELIB.LIB/MYFILE/MYMBR.MBR (the physical file name extension must be .FILE)

## Transferring multiple physical file members from a physical file in a single transfer request

Managed File Transfer on IBM i supports the transfer of multiple physical file members from a single physical file as a single transfer request. You can specify an appropriate path name that includes wildcard characters as shown in the following examples:

- ABCLIB contains a physical file MYFILE with multiple members. To transfer all these members in a single request, specify the following path name: /QSYS.LIB/ABCLIB.LIB/MYFILE.FILE/\* .MBR
- XYZLIB contains a physical file MYFILE whose member names differ by a single character, that is: TEST1.MBR, TEST2.MBR, TEST3.MBR, and so on. To transfer all these members in a single request, specify the following path name: /QSYS.LIB/XYZLIB.LIB/MYFILE.FILE/TEST? .MBR.

The following types of transfer requests are not supported for transferring multiple physical file members and result in an error:

- /QSYS.LIB/MYLIB.LIB/\*.\*
- /QSYS.LIB/MYLIB.LIB/\*
- /QSYS.LIB/MYLIB.LIB/\* .FILE/MYMBR.MBR
- /QSYS.LIB/MYLIB.LIB/MYFILE\*.FILE/\* .MBR (there is no support for wildcarding on file names, only on member names)
- /QSYS.LIB/MYLIB.LIB/\* .FILE/\* .MBR
- /QSYS.LIB/MYLIB.LIB/MYFILE.FILE (.FILE assumes a SAVF not a physical file, so if MYFILE is a physical file, the transfer fails with invalid file type error)

## Transferring physical file members to and from non-IBM i systems

MFT supports the transfer of physical file members to and from non-IBM i systems, such as AIX, Linux, and Windows. All transfers must be done in text mode. The following examples illustrate some of the supported **fteCreateTransfer** requests when working with non-IBM i systems:

- This command transfers physical file member FILE(FROMIBMI/FILE1) MBR(FILE1) on IBM i to text file /home/qfte/fromibmi/linux.mbr.txt on Linux:

```
fteCreateTransfer -da linux -dm QM1 -sa ibmi -sm QM1 -t text -df /home/qfte/fromibmi/
linux.mbr.txt /qsys.lib/fromibmi.lib/file1.file/file1.mbr
```

- This command transfers physical file member FILE(FROMIBMI/FILE1) MBR(FILE1) on IBM i to text file C:\FTE\fromibmi\windows.mbr.txt on Windows:

```
fteCreateTransfer -da windows -dm QM1 -sa ibmi -sm QM1 -t text -df
C:\FTE\fromibmi\windows.mbr.txt /qsys.lib/fromibmi.lib/file1.file/file1.mbr
```

- This command transfers text file C:\FTE\toibmi\file.txt on Windows to physical file member FILE(TOIBMI/EXISTS) MBR(WINDOWS) on IBM i:

```
fteCreateTransfer -da ibmi -dm QM1 -sa windows -sm QM1 -t text -df /qsys.lib/toibmi.lib/
exists.file/windows.mbr C:\FTE\toibmi\file.txt
```

The following commands are examples of invalid physical file member transfers with non-IBM i systems:

- This command fails because the source file on Windows has a .txt file extension but a destination directory of .file has been specified. When transferring using the destination directory parameter

to specify a destination physical file, the source file extension must be .mbr file, for example, C:\FTE\toibmi\file.mbr

```
fteCreateTransfer -da ibmi -dm QM1 -sa windows -sm QM1 -t text -dd /qsys.lib/toibmi.lib/windows.file C:\FTE\toibmi\file.txt
```

- The default transfer mode is binary and text mode must be specified when transferring physical file members.

```
fteCreateTransfer -da windows -dm QM1 -sa ibmi -sm QM1 -df C:\FTE\fromibmi\file.bin /qsys.lib/fromibmi.lib/file1.file/file1.mbr
```

MFT supports the transfer of physical file members that are in the QSYS.LIB file system but does not support the transfer of source physical file members that are in the QSYS.LIB file system. File transfers in the QDLS file system are supported using the provided sample user exits. You can use the user exit samples provided in MFT for the following tasks:

- Transfer files in the QDLS file system.
- Automatically transfer physical file members from an IBM i library in the same way as an MFT file monitor.
- Delete an empty file object when the source file member is deleted as part of the transfer.

For more information, see [Sample MFT on IBM i user exits](#).

#### Related reference

[“Guidelines for transferring files” on page 2491](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring files to or from IBM i systems” on page 2511](#)

If you transfer files to or from IBM i systems using Managed File Transfer in text mode and you want to convert the data in the files, consider the information in this topic.

## IBM i

### Transferring save files located in QSYS.LIB on IBM i

Managed File Transfer supports the transfer of save files located in the QSYS.LIB file system between two IBM i systems. Consider the following information when requesting file transfers of save files.

A save file on IBM i is located in a library on IBM i. A library can be one of the standard libraries that ship with the operating system for example QSYS or QGPL or it can be a library that is created by the user. Save files in the QSYS.LIB file system are identified in two different ways on IBM i. When working with CL commands on an IBM i command line, the naming syntax used is as follows:

```
FILE(library name/file name)
```

For example, a save file called MYSAVF is located in a library called SOMELIB is identified as FILE(SOMELIB/MYSAVF).

You can also identify the same save file by specifying a UNIX-like path name that follows the Integrated File System (IFS) naming convention. See [Path names in the QSYS.LIB file system](#) for more information. Using the IFS naming convention, MYSAVF in SOMELIB has the following path name:

```
/QSYS.LIB/SOMELIB.LIB/MYSAVF.FILE
```

Managed File Transfer on IBM i recognizes the IFS naming convention but does not support the syntax used by CL commands. The following examples illustrate valid and invalid path names for Managed File Transfer.

Some examples of valid path names for save file transfers are as follows:

- /QSYS.LIB/SOMELIB.LIB/MYSAVF.FILE (assuming MYSAVF save file is located in the library SOMELIB)
- /QSYS.LIB/MYSAVF.FILE (assuming MYSAVF is located in the library QSYS)

Some examples of invalid path names for save file transfers are as follows:

- SOMELIB.LIB/MYSAVF.FILE (Path name must start with /QSYS.LIB)
- /QSYS.LIB/MYLIB.LIB (Path must end in a save file name, not a library name)
- /QSYS.LIB/MYLIB.LIB/ (Save file name is required)
- /QSYS.LIB/SOMELIB.LIB/MYSAVF (Save file name must have a .FILE extension in name)
- /QSYS.LIB/SOMELIB.LIB/MYSAVF.SAVF (Save file name extension must be .FILE)

### Transferring multiple save files from a library in a single transfer request

Managed File Transfer on IBM i supports the transfer of multiple save files from a library as a single transfer request. You can specify an appropriate path name that includes wildcard characters as shown in the following examples:

- ABCLIB contains many save files. To transfer all these files in a single request, specify the following path name:

```
/QSYS.LIB/ABCLIB.LIB/*.FILE
```

- XYZLIB contains several save files whose names differ by a single character, that is: TEST1.FILE, TEST2.FILE, TEST3.FILE, and so on. To transfer all of these files in a single request, specify the following path name:

```
/QSYS.LIB/XYZLIB.LIB/TEST?.FILE
```

The following types of transfer requests are not supported for transferring multiple save files and result in an error:

- /QSYS.LIB/MYLIB.LIB/\*.\*
- /QSYS.LIB/MYLIB.LIB/\*

Managed File Transfer supports the transfer of save files that are located in the QSYS.LIB file system but the transfer of other types of files that are located in the QSYS.LIB file system is not supported. However, Managed File Transfer provides samples that use the save file support and use predefined fteAnt tasks to demonstrate how a complete library, a source physical file, or database file can be transferred between two IBM i systems. See [Getting started using Ant scripts with MFT](#) for details on how to customize and use these samples.

#### Related reference

[“Guidelines for transferring files” on page 2491](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“Transferring files to or from IBM i systems” on page 2511](#)

If you transfer files to or from IBM i systems using Managed File Transfer in text mode and you want to convert the data in the files, consider the information in this topic.

## Transferring generation data groups (GDGs)

Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

**Note:** When creating a GDG entry in a batch environment using BASEGDG(+n), it cannot be referred to later in the same job by using the same positive generation number. Maintaining the same GDG entry numbers between steps of a job is a function of JCL and is not available to utility functions that update the GDG by using dynamic allocation. Therefore, a job that creates a new generation using BASEGDG(+1) would find the GDG updated as soon as the transfer successfully completes and would then need to refer to the same data set as BASEGDG(0).

### GDG examples

The following examples show the **fteCreateTransfer** command using GDGs. In the examples, the name BASEGDG refers to an existing base GDG name. The name DSET refers to a sequential data set that is to be created. The name /u/user/file.dat refers to the name of a source data file.

This command copies file.dat into a new generation in BASEGDG. The absolute name of the new generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//BASEGDG(+1)" /u/user/file.dat
```

This command copies file.dat into the generation with the absolute name specified in BASEGDG:

```
fteCreateTransfer -sa A1 -da A2 -ds "//BASEGDG.G0009V00" /u/user/file.dat
```

This command copies the most recent generation in BASEGDG to DSET. The absolute name of the generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//DSET" "//BASEGDG(0)"
```

This command copies the next most recent generation in BASEGDG to DSET. The absolute name of the generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//DSET" "//BASEGDG(-1)"
```

### Related reference

[“Guidelines for transferring files” on page 2491](#)

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[“Transferring between data sets on z/OS” on page 2493](#)

You can transfer between z/OS data sets using Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

## Using wildcard characters with MFT

You can use wildcard characters when you specify source file names and source file paths for file transfers. This allows you to select multiple files simultaneously.

### Multiplatforms

You can use the following wildcard characters on [Multiplatforms](#):

?

Use the question mark (?) to represent exactly one character. All of the other characters specified are required in matching file names.

For example, `ab?d.jpg` matches the files `abcd.jpg`, `abed.jpg`, and `abfd.jpg`.


\*

Use the asterisk character (\*) to represent zero or more characters.

For example `*.txt` matches the files `abc.txt` and `x.txt`, but not `newtxt` because the period (.) in the file names is a required character.

The pattern `*txt` matches the files `abc.txt`, `x.txt`, and `newtxt`.

You must enclose the asterisk character (\*) in double quotation marks. If you do not, the character will be interpreted by the command shell and might cause the command to fail.

 On AIX and Linux, using the asterisk character (\*) will not include the pseudo hidden files, for example `.bashrc`.

If the operating system is case-insensitive for file and path names, for example Windows, the pattern match is case-insensitive. You can use wildcard characters to specify file names only: you cannot use wildcards in directory names.

### Protocol bridge agent

If you are using a protocol bridge agent to transfer files from an FTP, FTPS, or SFTP file server, wildcard matching is case sensitive, regardless of the platform that the file server is actually running on.

### Connect:Direct bridge

When the source of a transfer is a Connect:Direct bridge agent that is requesting files from a Connect:Direct node, wildcards are not supported.

### IBM i



You can use the following wildcard characters on IBM i platforms:

?

Use the question mark (?) to represent exactly one character. All of the other characters specified are required in matching file names.

For example, `ab?d.jpg` matches the files `abcd.jpg`, `abed.jpg`, and `abfd.jpg`.

\*

Use the asterisk character (\*) to represent zero or more characters.

For example `*.txt` matches the files `abc.txt` and `x.txt`.

The pattern `*txt` matches the files `abc.txt`, `x.txt`, and `newtxt` because the period (.) in the pattern is a required character.

For additional considerations regarding the use of wildcard characters with save file transfers, see [Transferring save files that reside in QSYS.LIB file system on IBM i systems](#).

## z/OS

**z/OS** For z/OS systems the wildcard character rules for Managed File Transfer follow the standard ISPF wildcard conventions in general. There are specific rules for both sequential and partitioned data sets as follows:

### Sequential data sets

**z/OS** When you reference sequential data sets, you can use data set name qualifiers containing asterisks (\*) and percent signs (%) as follows:

- \***  
Use a single asterisk (\*) to represent at least one qualifier. A single asterisk within a qualifier represents zero or more characters.
- \*\***  
Use double asterisks (\*\*) to represent zero or more qualifiers. You cannot use a double asterisk within a qualifier.
- %**  
Use a single percent sign (%) to represent one single alphanumeric or national language character.
- %%**  
Use between one and eight percent signs to represent zero or more characters.

### Partitioned data sets

**z/OS** When you reference partitioned data sets, you can specify wildcard characters for the member names only. You can use data set name qualifiers containing asterisks (\*), underscores (\_), and question marks (?) as follows:

- \***  
Use the asterisk (\*) character to represent zero or more characters.
- \_**  
Use the underscore (\_) character to represent exactly one character.
- ?**  
Use the question mark (?) character to represent exactly one character. The question mark is an alternative to the underscore character and is provided as an addition to ISPF conventions.

## Directories

By default if you create a file transfer with a wildcard pattern that matches subdirectories, the subdirectories are not transferred. You can specify the **-x** parameter on the `fteCreateTransfer` command to include subdirectories that match the wildcard pattern. When you transfer a subdirectory, the entire contents and structure of the subdirectory are transferred: including all of its files, subdirectories, and hidden files.

For example, if you have a directory called `abc`, there is a difference in behavior between specifying a source file path of `/opt/abc` and `/opt/abc/*`. In the case of `/opt/abc` because the directory is transferred, a directory called `abc` is created at the destination and all of the file contents are transferred. In the case of `/opt/abc/*`, the contents of `abc` are transferred into the destination path.

## Hidden files

Wildcards do not match hidden files except on UNIX-type platforms when the wildcard pattern starts with a dot character (.). For example: `/opt/.*` transfers all hidden files in the `opt` directory.

On Windows if you want to transfer a hidden file, either specify the file name exactly or transfer the directory containing the hidden file.

## Symbolic links

Symbolic links are a type of file that contain a pointer to another file or directory and are known as shortcuts on Windows. You can match symbolic link files with wildcard characters. However, when a destination file is created from a source that is a symbolic link, the destination file becomes a hard link (that is, a regular file). You cannot successfully transfer symbolic links to directories because this could potentially create a recursive path.

## Transferring files with wildcard characters in their file names

You can transfer a file if the file name itself contains a wildcard character. If you specify that file name exactly, only that file is transferred, and not the set of files that match the wildcard.

For example, if you have a file called `/opt/abc*.txt` and you create a file transfer for `/opt/abc*.txt`, the only file transferred is `/opt/abc*.txt`. But if you create a file transfer for `/opt/ab*.txt`, all files matching the pattern `/opt/ab*.txt` are transferred, including the file `/opt/abc*.txt`.

## Transferring directory paths that contain wildcard characters

Enclose any directory path that includes a wildcard character in quotation marks ( " ") or single quotation marks ( ' ') to avoid shell expansion. Shell expansion happens when the operating system expands the wildcard character before the character is passed to the Managed File Transfer command and this might cause unexpected behavior.

For example, if you run the following **fteCreateTransfer** command with the **-gt** parameter on AIX and Linux, where `${..}` is a variable substitution from a resource monitor:

```
fteCreateTransfer -p QM_VENUS -sa AGT.QM_JUPITER -sm QM_JUPITER -da AGT.QM_NEPTUNE -dm QM_NEPTUNE -r -sd
delete
-t binary -de overwrite -jn MONTASK -gt /home/ftadmin/bin/TransferTask.xml -df "${FilePath}" "${
FilePath}"
```

the shell parses `${FilePath}` and does not pass it to the command. The workaround is to enclose `${FilePath}` in double quotation marks, that is, `"${FilePath}"`.

## Transfer is reported as successful even though wildcard matches zero files

If you attempt to transfer a file that does not exist, Managed File Transfer treats this attempt as a failed transfer. If you specify a file name explicitly (for example, `/a/missing/filename.txt`) and MFT is unable to find that file, the following error message is reported in the log:

```
BFGI00001E: File "/a/missing/filename.txt" does not exist
```

As part of this process the source agent, which could not find the file, notifies the destination agent that this file transfer has been canceled (because the source agent cannot find the source file to read). If you had planned to trigger an exit after the transfer at this point, the destination agent triggers its `DestinationTransferEndExit` with a `FileExitResultCode` of `CANCEL_FILE` for that file name.

However, if you attempt to transfer a wildcard (for example, `/a/missing/*.txt`) and the source agent does not find any files that match that wildcard, MFT reports this as a successful transfer. This is because technically the source agent was asked to transfer 0 files. The following error message is reported in the log:

```
The transfer request has successfully completed, although no files were transferred.
```

In this example, because the destination agent was never involved in the transfer, its exit is not called.

## Related reference

[“Guidelines for transferring files” on page 2491](#)



Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

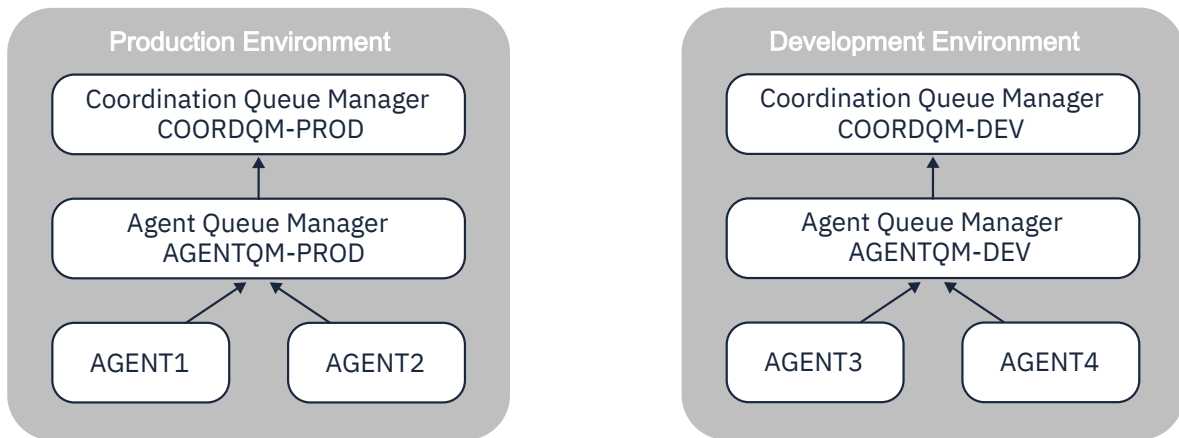
[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

## Transferring between two different MFT topologies

Managed File Transfer (MFT) agents can only perform managed transfers between other agents in the same topology. However, if you have multiple topologies, it can be useful to transfer data between them. The following text provides some high level guidance on how to do this.

Here is a diagram that shows two different topologies:



*Figure 6. AGENT1 and AGENT2 are part of a topology in the Production environment, and AGENT3 and AGENT4 are part of the Development environment topology.*

The Production topology is separate from the Development topology. This means that it is not possible for the agents in Production to directly participate in managed transfers with the agents in the Development environment (for example, AGENT2 cannot perform a managed transfer to AGENT3). To transfer data between the environments, you can use either a shared file system, or file-to-message and message-to-file transfers.

## Transferring data using a shared file system

In this solution, the agents in both topologies have access to the same shared file system.

An agent in one topology acts as the destination agent for a managed transfer and writes a file to a known location on the file system. Another agent in the second topology uses a resource monitor or a scheduled transfer to detect when a file appears in that location, and then processes it.

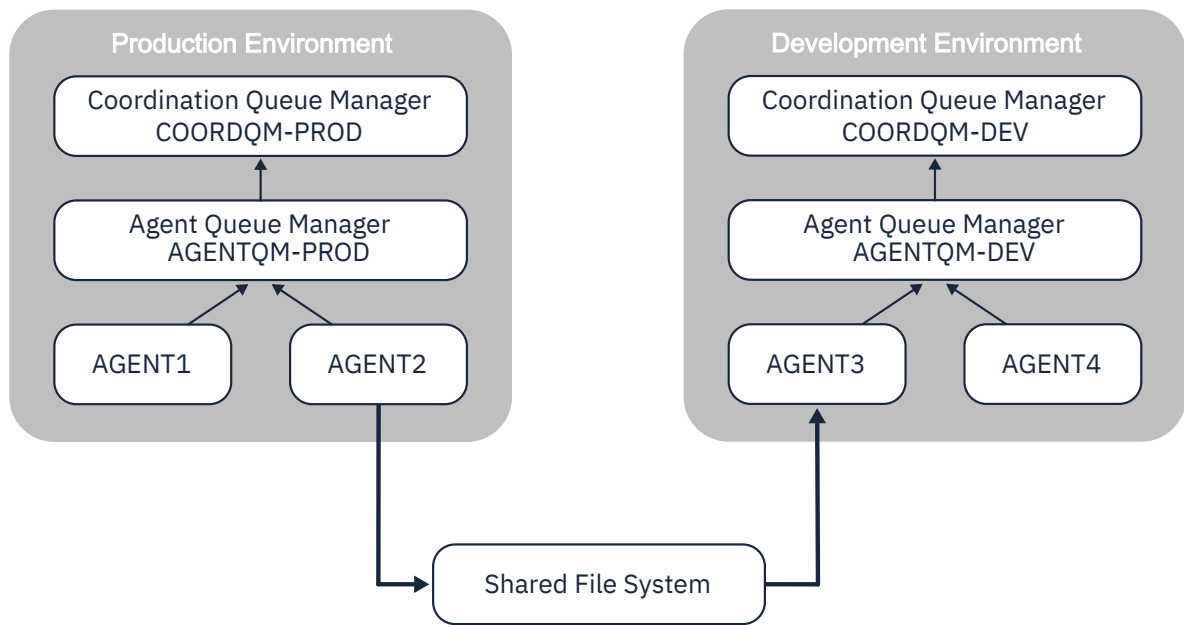


Figure 7. Here, AGENT2 is the destination agent for a managed transfer running in the Production Environment topology, and writes a file to a location on the shared file system. A resource monitor running inside of AGENT3 polls that location. When it detects the file written by AGENT2, it submits a managed transfer request to AGENT3 to process it and bring it into the Development Environment topology.

Note that the shared file system should be reliable, to ensure that data is not lost.

### Transferring data using message-to-file and file-to-message transfers

An alternative approach is to use a gateway queue manager in between the two topologies. This queue manager is connected to agent queue managers in the topologies using sender and receiver channels, to allow data to pass between the two.

An agent in one of the topologies performs a file-to-message transfer, to write data to a remote queue. The message is then routed through the gateway queue manager to a local queue on a queue manager in the other topology. An agent in that topology then performs a message-to-file transfer to get the message and process it.



Figure 8. Here, AGENT2 is connected to its agent queue manager AGENTQM-PROD and performs a file-to-message transfer to write a message to a queue called Q1. Q1 is a remote queue, and so the message gets routed via the Gateway Queue Manager and sender/receiver channels to the local queue Q1 on the queue manager AGENTQM-DEV. AGENT3 then performs a message-to-file transfer to get the message and bring it into the Development Environment topology.

This solution uses standard IBM MQ networking to transfer messages from one topology to another via the gateway queue manager. This means that if a channel between the gateway queue manager and one of the agent queue managers is unavailable for some reason, messages might get stuck and not arrive on the destination queue. In this situation, you should check the channels to ensure that they are all running.

### Related reference

“Guidelines for transferring files” on page 2491

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

## Regular expressions used by MFT

Managed File Transfer uses regular expressions in a number of scenarios. For example, regular expressions are used to match user IDs for Connect:Direct security credentials, or to split a file into multiple messages by creating a new message each time a regular expression is matched. The regular expression syntax used by Managed File Transfer is the syntax supported by the `java.util.regex` API. This regular expression syntax is similar to, but not the same as, the regular expression syntax used by the Perl language.

For more information about Java regular expressions, see the Java tutorial [Regular Expressions](#).

### Examples

To match all patterns, use the following regular expression:

```
.*
```

To match all patterns that begin with the string `fte`, use the following regular expression:

```
fte.*
```

To match all patterns that begin with the string `accounts` followed by a single digit, and end with `.txt`, use the following regular expression:

```
accounts[0-9]\.txt
```

## Substitution variables for use with user-defined Connect:Direct processes

You can define values to substitute in to user-defined Connect:Direct processes by using intrinsic symbolic variables that are specific to Managed File Transfer.

To follow the Connect:Direct naming convention, all intrinsic symbolic variables used by Managed File Transfer have the format `%FTE` followed by five uppercase alphanumeric characters. For more information about intrinsic symbolic variables, see the Connect:Direct product documentation.

When creating a process to transfer files from a Connect:Direct node to the Connect:Direct bridge system, you must use the intrinsic variable `%FTETFILE` as the value of `TO FILE` in the Connect:Direct process.

When creating a process to transfer files to a Connect:Direct node from the Connect:Direct bridge system, you must use the intrinsic variable `%FTEFFILE` as the value of `FROM FILE` in the Connect:Direct process.

These variables contain the temporary file paths that the Connect:Direct bridge agent uses for transfers into and out of the Managed File Transfer network.

Variable name	Description
<code>%FTESAGNT</code>	The name of the Managed File Transfer source agent. This variable is set only for transfers from a Managed File Transfer Agent to a Connect:Direct node.

Table 371. Intrinsic symbolic variables used by Managed File Transfer and Connect:Direct (continued)

Variable name	Description
%FTEDAGNT	The name of the Managed File Transfer destination agent. This variable is set only for transfers from a Connect:Direct node to a Managed File Transfer Agent.
%FTEPNODE	The Connect:Direct primary node name. The value is always the name of the Connect:Direct node that is part of the Connect:Direct bridge.
%FTEPPLAT	The platform that the Connect:Direct primary node runs on. Possible values for this variable are UNIX and WINDOWS. This information is provided by the Connect:Direct bridge agent.
%FTEPUSER	The Connect:Direct primary node user identifier to use in the Connect:Direct process. This information is taken from the ConnectDirectCredentials.xml file.
%FTEPPASS	The password to use with the user name defined by the %FTEPUSER variable. This information is taken from the ConnectDirectCredentials.xml file.
%FTESNODE	The Connect:Direct secondary node name. The value is always the name of the Connect:Direct node that the file is transferred to or from.
%FTESPLAT	The platform that the Connect:Direct secondary node runs on. Possible values for this variable are UNIX, WINDOWS, and ZOS. This information is taken from the ConnectDirectNodeProperties.xml file.
%FTESUSER	The Connect:Direct secondary node user identifier to use in the Connect:Direct process. This information is taken from the ConnectDirectCredentials.xml file.
%FTESPASS	The password to use with the user name defined by the %FTESUSER variable. This information is taken from the ConnectDirectCredentials.xml file.
%FTEFFILE	<p>The source file name. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, the value is the fully qualified location of the file on the same system as the Connect:Direct bridge.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, the value is the name of the file that is specified as the source file in the Managed File Transfer transfer request.</p>
%FTEFDISP	<p>The disposition of the source file when the process is complete. The value of this variable is platform dependent and equivalent to the values for MFT transfer request. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, the action of deleting or not deleting the source file is performed by the Managed File Transfer bridge agent.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, the action of deleting or not deleting the source file must be performed by the Connect:Direct process.</p>

Table 371. Intrinsic symbolic variables used by Managed File Transfer and Connect:Direct (continued)

Variable name	Description
%FTEFCP	<p>The code page to use for the source file. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, this value is UTF-8 or, if the transfer is a binary transfer, the value is not set.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, this value is specified by Connect:Direct or, if the transfer is a binary transfer, the value is not set.</p>
%FTEFSYSO	<p>The Connect:Direct SYSOPTS for the source of the transfer. If the remote Connect:Direct node is on AIX, Linux, and Windows, this value contains information about the code page and data type of the source of the transfer. <b>z/OS</b> If the remote node is on z/OS, this value contains additional information.</p>
%FTEFNODE	<p>Identifies the Connect:Direct node where the source file resides. This will be set to a value of either: PNODE or SNODE.</p>
%FTETFILE	<p>The destination file name. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, the value is the name of the file that is specified as the destination file in the Managed File Transfer transfer request.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, the value is the fully qualified name of the location to write the file to on the same system as the Connect:Direct bridge.</p>
%FTETDISP	<p>The disposition of the destination file. The value of this variable is platform dependent and equivalent to the values for Connect:Direct transfer request. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, the action of creating a file or replacing an existing file must be performed by the Connect:Direct process.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, the action of creating a file or replacing an existing file is performed by the Managed File Transfer bridge agent.</p>
%FTETCP	<p>The code page to use for the destination file. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.</p> <p>When transferring files from a Managed File Transfer Agent to a Connect:Direct node, this value is specified by Connect:Direct or, if the transfer is a binary transfer, the value is not set.</p> <p>When transferring files from a Connect:Direct node to a Managed File Transfer Agent, this value is UTF-8 or, if the transfer is a binary transfer, the value is not set.</p>
%FTETSYSO	<p>The Connect:Direct SYSOPTS for the destination of the transfer. If the remote Connect:Direct node is on AIX, Linux, and Windows, this value contains information about the code page and data type of the destination of the transfer. <b>z/OS</b> If the remote node is on Windows, this value contains additional information.</p>
%FTETNODE	<p>Identifies the Connect:Direct node where the destination file is to reside. This will be set to a value of either: PNODE or SNODE.</p>

Table 371. Intrinsic symbolic variables used by Managed File Transfer and Connect:Direct (continued)

Variable name	Description
%FTEDTYPE	The data type or mode of the transfer. Possible values for this variable are text or binary. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.
%FTETRID	The 48-character hexadecimal transfer ID from the Managed File Transfer transfer.
%FTEJOB	The job name from the Managed File Transfer transfer request. The value of this variable is truncated to 256 characters and can be used in the process accounting data.
%FTEPNAME	The Connect:Direct process name generated by the Managed File Transfer bridge agent. The value of this variable is 8 alphanumeric characters. The value always starts with an alphabetic character.
%FTEMETA(key)	A metadata from the Managed File Transfer transfer request. The value of <i>key</i> is the key of the metadata. The value of <i>key</i> is not case sensitive. A key of ABC is treated the same as a key of abc. If both ABC and abc are defined as metadata keys, the value of the second metadata defined overwrites the value of the first metadata defined.


 The following table contains information about additional intrinsic symbolic variables that are used when the remote Connect:Direct node in the transfer is on a z/OS platform.

Table 372. Additional intrinsic symbolic variables used when the remote Connect:Direct is on z/OS

Variable name	Description
%FTEFDCB	The value of the DCB parameter at the source of the transfer.
%FTEFSPACE	The value of the SPACE parameter at the source of the transfer.
%FTEFLBEL	The value of the LABEL parameter at the source of the transfer.
%FTEFUNIT	The value of the UNIT parameter at the source of the transfer.
%FTEFVOL	The value of the VOL parameter at the source of the transfer.
%FTEFDAACL	The value of the DATACLAS parameter at the source of the transfer.
%FTETDCB	The value of the DCB parameter at the destination of the transfer.
%FTETSPACE	The value of the SPACE parameter at the destination of the transfer.
%FTETLBEL	The value of the LABEL parameter at the destination of the transfer.
%FTETUNIT	The value of the UNIT parameter at the destination of the transfer.
%FTETVOL	The value of the VOL parameter at the destination of the transfer.

Table 372. Additional intrinsic symbolic variables used when the remote Connect:Direct is on z/OS (continued)

Variable name	Description
%FTETDACL	The value of the DATACLAS parameter at the destination of the transfer.
%FTETDSTY	The value of the DSNTYPE parameter at the destination of the transfer.
%FTETLIKE	The value of the LIKE parameter at the destination of the transfer.
%FTETMGCL	The value of the MGMTCLAS parameter at the destination of the transfer.
%FTETSTCL	The value of the STORCLAS parameter at the destination of the transfer.

## Example: A Connect:Direct process file that calls MFT commands

An example Connect:Direct process file that calls the Managed File Transfer **ftetag** command and the **ftecxfer** command.

In this example, the following actions occur:

1. A Connect:Direct COPY statement transfers the file from C:\test\from\sent.txt on the system where the secondary node runs to C:\test\tmp\midpoint.txt on the system where the primary node runs.
2. The Connect:Direct process calls the **ftetag** command to create audit information in MFT.
3. The Connect:Direct process calls the **ftecxfer** command.
4. The **ftecxfer** command transfers the file from C:\test\tmp\midpoint.txt on the system where the primary node runs and the agent CD\_BRIDGE runs to /test/to/arrived.txt on the system where the agent LINUX\_AGENT is located.

```

/*BEGIN_REQUESTER_COMMENTS
 $PNODE$="cd_win01" $PNODE_OS$="Windows"
 $SNODE$="CD_WIN01" $SNODE_OS$="Windows"
 $OPTIONS$="WDOS"
 END_REQUESTER_COMMENTS*/

TESTPRO PROCESS
  SNODE=CD_WIN01

  COPY
    FROM (
      FILE=C:\test\from\sent.txt
      SNODE
    )
    TO (
      FILE=C:\test\tmp\midpoint.txt
      PNODE
      DISP=RPL
    )
    COMPRESS Extended

  RUN TASK PNODE
    SYSOPTS="pgm(C:\wmqfte\bin\ftetag) args(C:\test\tmp\midpoint.txt)"

  RUN TASK PNODE
    SYSOPTS="pgm(C:\wmqfte\bin\ftecxfer) args(-qmgrname QM_CDBA -connname fish.example.com(1441)
    -channelname SYSTEM.DEF.SVRCONN
    -sa CD_BRIDGE -da LINUX_AGENT -sm QM_CDBA -dm QM_LINUX -de overwrite -df /test/to/arrived.txt
    C:\test\tmp\midpoint.txt"

PEND

```


## Related tasks

[Creating and submitting a Connect:Direct process that calls Managed File Transfer by using the Connect:Direct Requester](#)

[Using Connect:Direct processes to submit Managed File Transfer transfer requests](#)

## Restrictions of the Connect:Direct bridge agent

The Connect:Direct bridge agent is configured to transfer files to and from Connect:Direct nodes. There are some functions that the Connect:Direct bridge agent is not capable of performing.

- The Connect:Direct bridge agent cannot read messages from a queue or write messages to a queue. It cannot act as the destination agent in a file-to-message transfer or as the source agent in a message-to-file transfer.
- You cannot define a resource monitor on the Connect:Direct bridge agent.
- You cannot have a Connect:Direct bridge agent as both the source and destination of a transfer. You cannot transfer from Connect:Direct node to Connect:Direct node through the Connect:Direct bridge.
- The Connect:Direct bridge agent does not support user exits that are called before or after the transfer. The Connect:Direct bridge agent does support a credential mapping exit. For more information, see [Mapping credentials for Connect:Direct by using exit classes](#).
- You cannot define presrc or postsrc program invocations for a transfer that has the Connect:Direct bridge agent as the source agent. For more information, see [Program invocation nested elements](#).
- You cannot define predst or postdst program invocations for a transfer that has the Connect:Direct bridge agent as the destination agent. For more information, see [Program invocation nested elements](#).
- You cannot specify a wildcard character in the source specification if the source agent is the Connect:Direct bridge agent.
- If you specify a source disposition (**-sd**) of deLeTe when transferring a file  or data set from a Connect:Direct node, the behavior is different to the usual source disposition behavior. One of the following cases occurs:
  - If Connect:Direct uses a process that is generated by Managed File Transfer to move the file or data set from the source, specifying the deLeTe option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see [Submitting a user-defined Connect:Direct process from a file transfer request](#).
  - If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

## Related reference

[The Connect:Direct bridge](#)

## FTPS server support by the protocol bridge

The protocol bridge supports a subset of the FTPS protocol as defined by RFC-2228, RFC-4217, and the Internet-Draft entitled *Secure FTP over SSL*.

For a list of valid cipher suite values for connections between protocol bridge agents and FTPS servers, see [Cipher suites](#) in the IBM SDK and Runtime Environment Java Technology Edition 7 product documentation.



The following features of the FTPS protocol are supported:

- Implicit and explicit modes of operation.
- Validation of the server certificate.
- Optional mutual authentication using client certificate checks.



- Optional use of a clear control channel after the initial authentication and level of protection for the data channel has been selected.
- SHA-2 cipher suites and FIPS 140-2 compliance are supported. The following versions of Java are required: IBM JREs 6.0 SR13 FP2, 7.0 SR4 FP2, or later.

The following features of the FTPS protocol and runtime environment are not supported:

- Use of the **ADAT** command for additional security data exchange.
- Use of FTPS for channel encryption only that is, where the servers certificate is not validated.
- Selection of the Clear, Secure, or Confidential levels of protection using the **PROT** command.
- Encryption for each command using the **MIC**, **CONF**, and **ENC** commands.
- Fallback to the FTP protocol if the server does not support explicit FTPS. Use the FTP support provided by the protocol bridge to work with such a server.
- Use of the **FEAT** command to determine the available capabilities of the FTPS server.
- Validation of certificates using pattern matching against the DN field.
- Certificate revocation checking.
- Validation of certificates with the issuing trusted certificate authority.
- Explicit selection of the cipher suites available to the SSL negotiation phase of establishing a session.
-  Use of extensions specific to z/OS  or IBM i that integrate cryptography with the operating system. Specifically, the use of the z/OS keyring or non-hierarchical file systems for storing key and trust information, for example, data sets. Cryptographic hardware and offload engines are used if these functions are managed transparently by the JVM and do not require explicit application code.

#### **Related reference**

[The protocol bridge](#)

## **SFTP server support by the protocol bridge**

The protocol bridge supports the SFTP protocol as defined by the IETF Internet Draft entitled SSH File Transfer Protocol, version 6 draft 13.

Protocol bridge agents support the following ciphers when connecting to a file server using the SFTP protocol:

- blowfish-cbc
- 3des-cbc
- aes128-cbc
- aes192-cbc
- aes256-cbc
- aes128-ctr
- aes192-ctr
- aes256-ctr
- 3des-ctr
- arcfour
- arcfour128
- arcfour256

By default, the list of ciphers used by protocol bridge agents is aes128-cbc,aes192-cbc,aes256-cbc. For information on how to configure a protocol bridge agent to use specify different ciphers, see [“Protocol bridge properties file format”](#) on page 2704.

## Methods of authentication

If you have provided the IBM MQ Managed File Transfer (MFT) protocol bridge agent code with a private key and a server password, for a single user within the `ProtocolBridgeCredentials.xml` file, the MFT protocol bridge agent by default, configures the JSch library to use both methods of authentication, if required by the SFTP file server, when establishing a connection.

Should both a private key and a server password be configured for a single user within the `ProtocolBridgeCredentials.xml` file, but the SFTP file server requires only one of these authentication methods, the MFT protocol bridge agent configures the JSch library to use public/private-key authentication in preference to password based authentication.

Should the SFTP file server reject the attempt to use public/private-key authentication, then the MFT protocol bridge agent, using the JSch library, attempts username and password based authentication.

If either of these authentications alone is successful, a connection is established to the SFTP file server.

To configure both private key and a password authentication for the `ProtocolBridgeCredentials.xml` file, associated with the MFT protocol bridge agent, you need to specify:

- The **serverPassword** attribute (with associated value) in the element that maps from an MFT user name to a protocol server user name, and
- The element for the MFT user defined by the parent element.

For example, the syntax could be as follows:

```
-----BEGIN RSA PRIVATE KEY-----  
...  
-----END RSA PRIVATE KEY-----
```

## Keyboard interactive method

The MFT protocol bridge agent uses the JSch, third-party library, to connect to SFTP file servers. You can configure the JSch library so that it can attempt to authenticate with an SFTP file server using the *keyboard-interactive* method when no private key is specified in the `ProtocolBridgeCredentials.xml` file.

Note that authentication using the *keyboard-interactive* method works only if the SFTP file server prompts for the password using the string `password:` (in either upper, lower or mixed case). In the situation where you use the *keyboard-interactive* authentication method, and the SFTP file server responds with a string different from `password:`, the connection attempt fails.

When the SFTP file server responds to the initial connection attempt with this string, the protocol bridge agent, using the JSch library, sends the password configured in the **serverPassword** attribute of the user element within the `ProtocolBridgeCredentials.xml` file.

### Related reference

[The protocol bridge](#)



## FIPS support in MFT

Managed File Transfer supports the use of FIPS-compliant cryptography modules in client connections from agents, commands, and the IBM MQ Explorer to queue managers. All SSL connections to the queue manager use the TLS protocol only. Support is provided for JKS and PKCS#12 keystore types.

**Note:** On AIX, Linux, and Windows, IBM MQ provides FIPS 140-2 compliance through the "IBM Crypto for C" cryptographic module. The certificate for this module has been moved to the Historical status. Customers should view the [IBM Crypto for C certificate](#) and be aware of any advice provided by NIST. A replacement FIPS 140-3 module is currently in progress and its status can be viewed by searching for it in the [NIST CMVP modules in process list](#).

Specify whether you want to enable FIPS support for an agent, a coordination queue manager, or a command queue manager as follows:

- If you want to enable FIPS for a specific agent, set the appropriate `agentSsl` properties in the `agent.properties` file for that agent. For more information, see [SSL/TLS properties for MFT](#).
- If you want to enable FIPS for a specific coordination queue manager, set the appropriate `coordinationSsl` properties in the `coordination.properties` file for that coordination queue manager. For more information, see [SSL/TLS properties for MFT](#).
- If you want to enable FIPS for a specific command queue manager, set the appropriate `connectionSsl` properties in the `command.properties` file for that command queue manager. For more information, see [SSL/TLS properties for MFT](#).

 FIPS is not supported on Managed File Transfer for  IBM i.

FIPS is not supported on connections to or from a protocol bridge or a Connect:Direct bridge.

For more information about IBM MQ and FIPS and the configuration steps required, see [Federal Information Processing Standards \(FIPS\)](#).

If you want to use FIPS, the CipherSuite must be FIPS-compliant or the connection fails. For more information about the CipherSpecs supported by IBM MQ, see [SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for Java](#) and [SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS](#).

## MFT database logger tables

When you have installed and configured the logger, a number of database tables are created.

### MFT Logger database schema updates

From IBM MQ 9.1, certain data types have been modified in the database schema, causing a change to the widths of columns in those tables:

#### Db2 schema

LONG VARCHAR in the following tables has been modified to VARCHAR in the Db2 schema, with a fixed length of 2000 bytes, or 256 characters.

- SCHEDULE\_ACTION
- TRANSFER\_ITEM
- SCHEDULE\_ITEM
- TRIGGER\_CONDITION
- CALL\_ARGUMENT
- CALL
- CALL\_REQUEST
- TRANSFER
- CALL\_RESULT
- MONITOR\_METADATA
- MONITOR\_EXIT\_RESULT
- MONITOR\_ACTION
- AUTH\_EVENT
- FILE\_SPACE\_ENTRY

By default, LONG VARCHAR allowed you to store 32700 bytes, but VARCHAR(*Size*) limits the modified column size to 2000 characters, or 256 characters.

See [“Migrating a Db2 database to the new schema” on page 2543](#) for more information on migrating a Db2 database to the new schema.

## Oracle schema

NCLOB in the following tables has been modified to NVARCHAR(*Size*), where *Size* can be 2000 bytes or 256 bytes:

- SCHEDULE\_ACTION
- TRANSFER\_ITEM
- SCHEDULE\_ITEM
- TRIGGER\_CONDITION
- CALL\_ARGUMENT
- CALL
- CALL\_REQUEST
- TRANSFER
- CALL\_RESULT
- MONITOR\_METADATA
- MONITOR\_EXIT\_RESULT
- MONITOR\_ACTION
- AUTH\_EVENT
- FILE\_SPACE\_ENTRY

By default, NVARCHAR2 allows you to store only 4000 bytes. You must set the MAX\_STRING\_SIZE property to *extended* for the database to extend the storage to 32767 bytes.

See [“Migrating an Oracle database to the new schema” on page 2546](#) for more information on migrating an Oracle database to the new schema.


In the SOURCE\_FILENAME and DESTINATION\_FILENAME columns, in the TRANSFER\_ITEM and SCHEDULE\_ITEM tables, a datatype of 2000 characters, (VARCHAR(2000)) brings commonality in both the Db2 and Oracle schemas.

## AUTH\_EVENT

An event related to authority checking, typically the rejection of a request due to insufficient privileges.

- **ID:** Row ID.
- **ACTION:** The type of action that took place.
- **COMMAND\_ID:** The IBM MQ message ID of the original message that requested the event. In the case of a transfer request, this will also be the transfer ID.
- **TIME:** The time at which the event occurred.
- **ORIGINATOR\_MQ\_USER:** The user ID contained in the IBM MQ message, against which the authority check was performed.
- **AUTHORITY:** The authority that was required for the requested action.
- **ORIGINAL\_XML\_REQUEST:** The payload of the command message, indicating what action was refused.
- **RESULTCODE:** The numeric code identifying the result.
- **RESULT\_TEXT:** A message explaining the result of the authority event.

## CALL

The remote running of an operating system command, or Ant script , or z/OS JCL job, managed by Managed File Transfer. Calls can be embedded in transfers, or referred to by call\_request rows.

A CALL (that is, a row in this table) can either be part of a normal transfer (in which case TRANSFER\_CALLS is used to link it to the relevant entry in TRANSFERS) or it can be a stand-alone managed call on its own (available only from Ant or by directly inserting messages). In the latter case, the CALL\_REQUEST table is used instead of the TRANSFERS table; an equivalent to TRANSFER\_CALLS is not needed because there can be only one call per call request.

- **ID:** Row ID.
- **COMMAND:** The command that was run. This field does not include any arguments passed to the command or the path where the command is located.
- **TYPE:** The type of command, such as Ant or JCL.
- **RETRIES:** The number of retries that were requested.
- **RETRY\_WAIT:** The interval to wait between retries as originally requested, in seconds.
- **SUCCESS\_RC:** The return code that indicates a successful completion of the command. If any other code is received, the run is reported to have failed.
- **EXECUTED\_COMMAND:** The full name of the command that was run, including path.
- **CAPPED\_RETRIES:** The number of retries available; this number might be less than requested if the retry limit of the agent is lower than the number of retries requested.
- **CAPPED\_RETRY\_WAIT:** The interval between retries that is used; this number might be less than requested if the configured limit of the agent is lower than the retry wait requested.
- **OUTCOME:** Whether the call was successful overall. If there were multiple tries the outcome of each one is recorded separately in the CALL\_RESULT table.

## CALL\_ARGUMENT

An argument or parameter supplied to a command that is called.

- **ID:** Row ID.
- **CALL\_ID:** The call that the argument is associated with.
- **KEY:** Where the argument is of a key-value-pair kind, the key, or name.
- **TYPE:** The type of the argument: some are position parameters to operating system commands and others are named properties used with Ant.
- **VALUE:** The value of the argument.

## CALL\_REQUEST

The vehicle for a command call that is not part of a file transfer. You can submit ManagedCall messages using Ant and using direct XML injection.

- **ID:** The hexadecimal ID of the managed call request.
- **CALL\_ID:** The database ID of the row in the CALL table describing this call.
- **ACTION\_TIME:** The time that the action occurred.
- **AGENT:** The agent that the command is run on.
- **AGENT\_QM:** The queue manager used by the agent that the command is run on.
- **ARCHITECTURE:** The machine architecture of the system that the agent runs on.
- **OS\_NAME:** The name of the operating system that the agent is running on.
- **OS\_VERSION:** The version of the operating system.
- **ORIGINATOR\_HOST:** The host name of the machine that the call request was submitted from.
- **ORIGINATOR\_USER:** The name of the user who submitted the call request, as reported in the request XML.
- **ORIGINATOR\_MQ\_USER:** The name of the user who submitted the call request, as contained in the IBM MQ message descriptor of the request.

- **JOB\_NAME:** A user-specified job name.
- **RESULTCODE:** The overall result code for the call.
- **RESULTTEXT:** The overall result message for the call.

## CALL\_RESULT

The detailed result of calling a command. A call can have multiple results if retries were enabled.

- **ID:** Row ID.
- **CALL\_ID:** The database ID of the row in the CALL table that this result applies to.
- **SEQUENCE:** Which attempt this result applies to, where there have been multiple attempts.
- **OUTCOME:** The outcome (for example, success or failure) of the command.
- **RETURN\_CODE:** The command return code.
- **TIME:** The time that the command completed.
- **STDOUT:** The standard output stream from the command, if it was started.
- **STDERR:** The standard error stream from the command, if it was started.
- **ERROR:** If the command could not be started, an error message produced by Managed File Transfer explaining the problem.

## FILE\_SPACE\_ENTRY

Each row represents a file that has been sent to the named file space.

- **ID:** The ID of the file space entry.
- **FILE\_SPACE\_NAME:** The name of the file space. This is the name of the user that the file space belongs to.
- **TRANSFER\_ITEM\_ID:** The ID of the transfer item that this row relates to.
- **ALIAS:** The alias name for this file space entry. Typically this alias name is the name of the source file for the transfer.
- **DELETED:** The time when the file was deleted from the file space. If the file has not been deleted the value is null.

## METADATA

Metadata associated with a transfer.

- **ID:** Row ID.
- **TRANSFER\_EVENT\_ID:** The transfer\_event row that this metadata is associated with, if it relates to a transfer. This field is null if the metadata is associated with a stand-alone managed call.
- **STANDALONE\_CALL\_ID:** If the metadata is associated with a stand-alone managed call, the ID of the managed call request concerned.
- **KEY:** The name of the metadata item.
- **VALUE:** The value of the metadata item.

## MONITOR

Resource monitors that trigger Managed File Transfer operations based on external conditions.

- **AGENT:** The agent that the monitor runs on.
- **ID:** The hexadecimal ID of the monitor.
- **NAME:** The name of the monitor.
- **QMGR:** The queue manager of the agent where the monitor runs.

## MONITOR\_ACTION

Each row represents an action (for example, creation and triggering) occurring in respect of a monitor

- **ID:** Row ID.
- **ACTION:** The type of action that took place.
- **JOB\_NAME:** The name of the submitted job, where applicable.
- **MONITOR:** The monitor that this action occurred on. Might be null if the action failed because it was requested for a monitor that does not exist.
- **ORIGINAL\_XML\_REQUEST:** If this action was a *create* or *triggerSatisfied* action, the XML request that is started when the monitor is triggered.
- **ORIGINATOR\_MQ\_USER:** The user ID contained in the IBM MQ message that initiated the action
- **ORIGINATOR\_USER:** The user name that submitted the request to perform the action.
- **ORIGINATOR\_HOST:** The machine from which the user submitted the request to perform the action.
- **TIME:** The time that the action occurred.
- **UPDATED\_XML\_REQUEST:** If the action is *triggerSatisfied*, the XML request that was started. This request might vary from the XML request that was originally made because of variable substitution.

## MONITOR\_EXIT\_RESULT

The result of running a resource monitor exit.

- **ID:** Row ID.
- **ACTION\_ID:** The monitor action that the result is associated with.
- **EXIT\_NAME:** The name of the exit that produced this result.
- **RESULTCODE:** The value that the exit returned, either cancel or proceed.
- **RESULTTEXT:** The text output from the exit, if provided.

## MONITOR\_METADATA

Items of metadata associated with a resource monitor.

- **ID:** Row ID.
- **ACTION\_ID:** The monitor\_action that the metadata is associated with.
- **KEY:** The name of the metadata item.
- **PHASE:** Whether this metadata item represents the data that was originally submitted or the updated version after variable substitution.
- **VALUE:** The value of the metadata item.

## SCHEDULE

A transfer schedule registered with an agent.

- **AGENT:** The name of the agent that has this schedule.
- **CREATION\_DATE:** The point in time that this schedule was created.
- **ID:** The unique database (not agent) ID for the schedule.
- **ID\_ON\_AGENT:** The ID that the agent uses for the database ID. This ID is not unique across agents and might not even be unique in an agent if the persistent state of the agent is reset.
- **LATEST\_ACTION:** The most recent action that modified the state of this schedule.

## SCHEDULE\_ACTION

When an event occurs that modifies the schedule state, an action is recorded.

- **ACTION\_TYPE:** The action that occurred.
- **ID:** Row ID
- **ORIGINATOR\_HOST:** The machine that the request that caused the change was submitted from.
- **ORIGINATOR\_USER:** The user whose name the request that caused the change was submitted in.
- **SCHEDULE\_ID:** The schedule that this action applies to.
- **SPEC\_AFTERWARDS:** The schedule\_spec that represents the state of this schedule after the action occurred.
- **STATUS\_CODE:** A numeric return code describing the outcome of the action
- **STATUS\_TEXT:** A text description of the outcome of the action. Typically null if the action succeeded.
- **TIME:** The point in time that the action occurred



## SCHEDULE\_SPEC

The details of an individual scheduled transfer.

- **ID:** Row ID.
- **DESTINATION\_AGENT:** The agent that the files are transferred to.
- **DESTINATION\_QM:** The queue manager used by the destination agent.
- **REPEAT\_COUNT:** How many times to repeat if the schedule repeats and is bound by the number of occurrences rather than an end time.
- **REPEAT\_FREQUENCY:** How many repeat\_intervals there are between scheduled transfers.
- **REPEAT\_INTERVAL:** If the transfer repeats, what interval to repeat at (for example, minutes or weeks).
- **SOURCE\_AGENT:** The agent that the files are transferred from.
- **SOURCE\_QM:** The queue manager used by the source agent.
- **START\_TIME:** The time that the first transfer in the schedule will take place.
- **START\_TIMEBASE:** The time base for the times associated with the transfer. For example, whether to operate from the time zone of the agent or the time zone of the administrator.
- **START\_TIMEZONE:** The time zone that the time base corresponds to and which will be used in operating the schedule.

## SCHEDULE\_ITEM

Each file (or pattern to match at transfer time) is represented by a schedule\_item.

- **ID:** Row ID.
- **CHECKSUM\_METHOD:** How the checksum for the file is calculated
- **DESTINATION\_EXISTS\_ACTION:** What action the destination agent takes if the file already exists at the destination.
- **DESTINATION\_FILENAME:** The file or directory that the files are transferred into.
- **DESTINATION\_QUEUE:** The destination queue name for a file-to-message transfer.
-  **DESTINATION\_TYPE:** Whether the destination\_filename column refers to a file or directory.
-  **DESTINATION\_TYPE:** Whether the destination\_filename column refers to a file, directory, or data set.
- **FILE\_MODE:** The mode (for example, *text* or *binary*) that the file is transferred in.
- **RECURSIVE:** When the agent creates the transfer according to the schedule, whether the agent recurses (Y) or not (N) the source directory.
- **SCHEDULE\_SPEC\_ID:** The schedule\_spec that this item is associated with.



- **SOURCE\_DISPOSITION:** What action to perform on source files after the transfer completes.
- **SOURCE\_FILENAME:** The source file, directory name, or pattern.
- **SOURCE\_QUEUE:** The source queue name for a message-to-file transfer

## TRANSFER

A single transfer of one or more files.

- **TRANSFER\_ID:** The hexadecimal ID for the transfer.
- **JOB\_NAME:** A user-specified job name for the transfer.
- **SCHEDULE\_ID:** If this transfer is the result of a schedule, the database row ID of the schedule concerned.
- **START\_ID:** The row ID of the transfer\_event that represents the start of the transfer.
- **COMPLETE\_ID:** The row ID of the transfer\_event that represents the end of the transfer.
- **RESULTCODE:** The overall result code for the transfer. The possible values for this column are listed in the following topic: [Return codes for MFT](#). These codes apply to the transfer as a whole; see [TRANSFER\\_ITEM.RESULTCODE](#) for the status of each individual item.
- **RESULTTEXT:** The overall result text for the transfer, if any.
- **STATUS:** The status of a transfer. The possible values for this column are started, success, partial success, failure, and cancelled.
- **RELATED\_TRANSFER\_ID:** The hexadecimal ID of a previous transfer that is related to this transfer. For example, if the transfer is a file download, this field will refer to the transfer that uploaded the file.

## TRANSFER\_CALLS

Links runnable command calls to transfers

- **ID:** Row ID.
- **POST\_DESTINATION\_CALL:** The call made at the destination after the transfer is complete.
- **POST\_SOURCE\_CALL:** The call made at the source agent after the transfer is complete.
- **PRE\_DESTINATION\_CALL:** The call made at the destination agent before the transfer starts.
- **PRE\_SOURCE\_CALL:** The call made at the source agent before the transfer starts.
- **TRANSFER\_ID:** The transfer that the calls in this row are associated with.

## TRANSFER\_CD\_NODE

Information about Connect:Direct nodes that are used in a transfer.

- **PNODE:** The primary node in the transfer.
- **SNODE:** The secondary node in the transfer.
- **BRIDGE\_IS\_PNODE:** Character indicating which node is the node that is part of the Connect:Direct bridge. If this value is Y, the primary node is the bridge node. If this value is N, the secondary node is the bridge node.
- **ID:** The ID of this row.

## TRANSFER\_CORRELATOR

Each row contains a correlation string and a number associated with a transfer item.

- **CORRELATION\_BOOLEAN:** A boolean correlation value. Represented by a single character of Y for true and N for false.
- **CORRELATION\_STRING:** A string correlation value.
- **CORRELATION\_NUMBER:** A numeric correlation value.

- **ID:** The ID of this row.

## TRANSFER\_EVENT

An event (start or end) related to a transfer.

- **ID:** Row ID.
- **ACTION\_TIME:** The time that the transfer action took place.
- **SOURCE\_AGENT:** The name of the agent that the files are transferred from.
- **SOURCE\_AGENT\_TYPE:** The type of agent that the files are transferred from. The following values are possible: 1 = STANDARD, 2 = BRIDGE, 3 = WEB\_GATEWAY, 4 = EMBEDDED, 5 = CD\_BRIDGE, 6 = SFG.  
**Note:** From IBM MQ 9.0, Managed File Transfer does not support the Web Gateway or web agents.
- **SOURCE\_QM:** The queue manager used by the source agent.
- **SOURCE\_ARCHITECTURE:** The machine architecture of the system hosting the source agent.
- **SOURCE\_OS\_NAME:** The operating system of the source agent machine.
- **SOURCE\_OS\_VERSION:** The version of operating system of the source agent machine.
- **SOURCE\_BRIDGE\_URL:** If the source agent is a protocol bridge agent, the URL of the data source to which it forms a bridge.
- **SOURCE\_CD\_NODE\_ID:** The Connect:Direct node that is the source of the transfer.
- **DESTINATION\_AGENT:** The name of the agent that the files are transferred to.
- **DESTINATION\_AGENT\_TYPE:** The type of agent that the files are transferred to. The following values are possible: 1 = STANDARD, 2 = BRIDGE, 3 = WEB\_GATEWAY, 4 = EMBEDDED, 5 = CD\_BRIDGE, 6 = SFG.  
**Note:** From IBM MQ 9.0, Managed File Transfer does not support the Web Gateway or web agents.
- **DESTINATION\_QM:** The queue manager used by the destination agent.
- **DESTINATION\_BRIDGE\_URL:** If the destination agent is a bridge agent, the URL of the data source to which it forms a bridge.
- **DESTINATION\_CD\_NODE\_ID:** The Connect:Direct node that is the destination of the transfer.
- **ORIGINATOR\_HOST:** The host name of the machine that the transfer request was submitted from.
- **ORIGINATOR\_USER:** The name of the user who submitted the transfer request, as reported by the `fteCreateTransfer` command.
- **ORIGINATOR\_MQ\_USER:** The name of the user who submitted the transfer request, as contained in the IBM MQ message descriptor of the request.
- **TRANSFERSET\_TIME:** The time that the transfer set was created.
- **TRANSFERSET\_SIZE:** The number of items being transferred.
- **TRIGGER\_LOG:** For transfer definitions involving a trigger, whether to log trigger evaluations that did not result in a transfer.

## TRANSFER\_EXIT

Each row represents a transfer exit which was executed as part of a file transfer.

- **ID:** Row ID.
- **EXIT\_NAME:** The name of the exit.
- **TRANSFER\_ID:** The ID of the completed or canceled transfer that this exit applies to.
- **TYPE:** The type of exit. This can be one of the following values: *SourceStart*, *SourceEnd*, *DestinationStart* or *DestinationEnd*.
- **STATUS:** The value that the exit returned. This can be *cancel* or *proceed*.
- **SUPPLEMENT:** An optional message explaining the status of the exit.

## TRANSFER\_ITEM

Each row represents a file that is sent as part of the transfer.

- **DESTINATION\_CHECKSUM\_METHOD:** The algorithm used to calculate a checksum of the destination file. Might be null if no checksum was calculated because the transfer did not complete successfully.
- **DESTINATION\_CHECKSUM\_VALUE:** The checksum value of the destination file. The value might be null if checksumming was disabled.
- **DESTINATION\_ENCODING:** The character encoding used on the destination file, if the destination file is transferred as text.
- **DESTINATION\_EXISTS\_ACTION:** The action to perform if the file exists at the destination.
- **DESTINATION\_FILE\_SIZE:** The size of the file name `z/OS` or data set name to use at the destination.
- **DESTINATION\_FILENAME:** The file name `z/OS` or data set name to use at the destination.
- **DESTINATION\_LINEEND:** The line-end format used in the destination file, if the destination file is transferred as text.
- **DESTINATION\_MESSAGE\_QUEUE\_NAME:** The destination queue for the messages that are produced from the source file during a file to message transfer.
- **DESTINATION\_MESSAGE\_GROUP\_ID:** If more than one message is produced, the group ID used for the messages that are produced from the source file during a file to message transfer.
- **DESTINATION\_MESSAGE\_MESSAGE\_ID:** If only one message is produced, The message ID of the message that is produced from the source file during a file to message transfer.
- **DESTINATION\_MESSAGE\_COUNT:** The number of messages that the source file was split into during a file to message transfer.
- **DESTINATION\_MESSAGE\_LENGTH:** The length of the message that is produced from the source file during a file to message transfer, in bytes. This value is only set if you specify a length for the output messages, for example by using the `-qs` option of the **fteCreateTransfer** command. If you specify `-qs 20K` and the size of your source file is 50 KB, the resulting three messages are 20 KB, 20 KB, and 10 KB in size. In this case the value of `DESTINATION_MESSAGE_LENGTH` is set to 20480.
- **DESTINATION\_CORRELATOR\_ID:** The ID of the correlator information for the destination.
- **FILE\_MODE:** The file transfer mode, for example *text* or *binary*.
- **ID:** Row ID
- **RESULTCODE:** A numeric code indicating the outcome of the transfer of this item. The possible values for this column are listed in the following topic: [Return codes for files in a transfer](#). These codes apply to the individual items in the transfer; see [TRANSFER.RESULTCODE](#) for the result of the transfer as a whole.
- **RESULT\_TEXT:** A textual explanation of the result of the transfer. Typically null if the transfer was successful.
- **SOURCE\_CHECKSUM\_METHOD:** The algorithm used to calculate a checksum of the source file.
- **SOURCE\_CHECKSUM\_VALUE:** The checksum value of the source file. The value might be null if checksumming was disabled.
- **SOURCE\_DISPOSITION:** The action to perform on the source file when the transfer is complete.
- **SOURCE\_ENCODING:** The character encoding used on the source file, if the source file is transferred as text.
- **SOURCE\_FILE\_SIZE:** The size of the file name `z/OS` or data set name to use at the source.
- **SOURCE\_FILENAME:** The source file name `z/OS` or data set name .
- **SOURCE\_LINEEND:** The line-end format used in the source file, if the source file is transferred as text.
- **SOURCE\_MESSAGE\_QUEUE\_NAME:** The source queue for the messages that are included in the destination file for a message to file transfer.

- **SOURCE\_MESSAGE\_GROUP\_ID:** The group ID of the messages that are included in the destination file for a message to file transfer.
- **SOURCE\_MESSAGE\_COUNT:** The number of messages that are included in the destination file for a message to file transfer.
- **SOURCE\_CORRELATOR\_ID:** The ID of the correlator information for the source.
- **TRANSFER\_ID:** The transfer that this item is part of.
- **TRUNCATE\_RECORDS:** Indicates whether over length data set records are to be truncated or wrapped.

## TRANSFER\_STATS

A set of statistics generated at the end of a transfer.

- **ID:** Row ID.
- **TRANSFER\_ID:** The transfer to which the statistics refer.
- **START\_TIME:** The time at which the transfer started. In a system that is busy or has intermittent connectivity, this time might be later than the time reported in the Started message, as that time represents the point at which initial processing began rather than the point at which the successful transfer of data began.
- **RETRY\_COUNT:** The number of times that the transfer had to be retried because of load or availability issues.
- **FILE\_FAILURES:** The number of files that failed to be transferred.
- **FILE\_WARNINGS:** The number of files that had warnings reported for them when they were transferred.

## TRIGGER\_CONDITION

One condition in a basic Managed File Transfer conditional transfer. For example, "file example.file exists".

- **ID:** Row ID.
- **TRANSFER\_EVENT\_ID:** The transfer event that the trigger is related to.
- **CONDITION\_TYPE:** The type of check used in the trigger. For example, the existence of a file or the size of a file.
- **COMPARISON:** The specific comparison to make. For example "greater than or equal to".
- **VALUE:** The value to compare against.
- **FILENAME:** The file name to examine.

### Related tasks

[Configuring an MFT logger](#)

### Related reference

[“fteStartLogger \(start an MFT logger\)” on page 2142](#)

The **fteStartLogger** command starts a Managed File Transfer logging application.

[“fteModifyLogger \(run an MFT logger as a Windows service\)” on page 2100](#)

Use the **fteModifyLogger** command to modify a Managed File Transfer logger so that it can be run as a Windows service. You can use this command only on Windows platforms, must be run by a user who is an IBM MQ administrator and a member of the mqm group, and you must first stop the logger by using the **fteStopLogger** command.

[“fteStopLogger \(stop an MFT logger\)” on page 2148](#)

The **fteStopLogger** command stops a Managed File Transfer logger.

## **Db2 entity relationship diagram**

A diagram showing the relationship of the entities in a Db2 database.

In the ERD, the #, \*, and o symbols each has a specific meaning:

- # means a primary key
- \* means that a value cannot be null
- o means that a value can be null

*Figure 9. Db2 entity relationship diagram (ERD)*



## Related tasks

[“Migrating a Db2 database to the new schema” on page 2543](#)

How you migrate a database with the existing schema to the new schema, by using the sample SQL script file.

## Migrating a Db2 database to the new schema

How you migrate a database with the existing schema to the new schema, by using the sample SQL script file.

## Before you begin

Take a backup of the database, and its relevant configuration information, that you are going to migrate and refer to the [“Db2 entity relationship diagram” on page 2541](#).



### Attention:

In the Db2 database, the LongVarchar data type now has a limit of :

- 2000 bytes in the SOURCE\_FILENAME and DESTINATION\_FILENAME columns, in the TRANSFER\_ITEM and SCHEDULE\_ITEM tables
- 4000 bytes, or 256 bytes for all the remaining columns, depending on the purpose of each column

If, for any reason, you want to increase the size of these database columns, you can change the script file and increase the size of the corresponding column.

## About this task

The following four sample SQL script files are located in `<MQ_Installation_Directory>/mqft/sql`:

- `db2_varchar_migration_step_1.sql`
- `db2_varchar_migration_step_2.sql`
- `db2_varchar_migration_step_3.sql`
- `db2_varchar_migration_step_4.sql`

## Procedure

1. Carry out the following tasks in order:

- a) Run **db2\_varchar\_migration\_step\_1.sql**
- b) Run **db2\_varchar\_migration\_step\_2.sql**
- c) Run **db2\_varchar\_migration\_step\_3.sql**
- d) Run **db2\_varchar\_migration\_step\_4.sql**

**Important:** Before running step [“1.c” on page 2543](#), ensure that steps [“1.a” on page 2543](#) and [“1.b” on page 2543](#) have run successfully.

2. Issue the command **cd <MQ\_Installation\_Directory>/mqft/sql**

3. Process the SQL script files, using the following commands in order:

- a) Run **db2 -tvvf db2\_varchar\_migration\_step\_1.sql**
- b) Run **db2 -tvvf db2\_varchar\_migration\_step\_2.sql**
- c) Run **db2 -tvvf db2\_varchar\_migration\_step\_3.sql**
- d) Run **db2 -tvvf db2\_varchar\_migration\_step\_4.sql**

## What to do next

If you receive some errors while creating new tables or new columns, caused by temporary table spaces, you can resolve these problems as follows:

### Error:

```
SQL State [54048], Error Code [-1585], Message [DB2 SQL Error: SQLCODE=1585 ,
SQLSTATE=54048, SQLERRMC=null in the trace file of logger
```

Explanation:

One of the following conditions could have occurred:

1. The row length of the system temporary table exceeded the limit that can be accommodated in the largest system temporary table space in the database.
2. The number of columns required in a system temporary table exceeded the limit that can be accommodated in the largest system temporary table space in the database.

### Link:

Message [SQL1585N](#).

### Solution:

Create a system temporary tablespace for each page as SMS (System Managed). In that case, your query always finds a tablespace with the appropriate page size.

### Example:

The following SQL commands resolve the preceding issue:

```
CREATE BUFFERPOOL BP4K pagesize 4K
CREATE SYSTEM TEMPORARY TABLESPACE STB_4 PAGESIZE 4K BUFFERPOOL BP4K
CREATE BUFFERPOOL BP8K pagesize 8K
CREATE SYSTEM TEMPORARY TABLESPACE STB_8 PAGESIZE 8K BUFFERPOOL BP8K
CREATE BUFFERPOOL BP16K pagesize 16K
CREATE SYSTEM TEMPORARY TABLESPACE STB_16 PAGESIZE 16K BUFFERPOOL BP16K
CREATE BUFFERPOOL BP32K pagesize 32K
CREATE SYSTEM TEMPORARY TABLESPACE STB_32 PAGESIZE 32K BUFFERPOOL BP32K
```

## Oracle entity relationship diagram

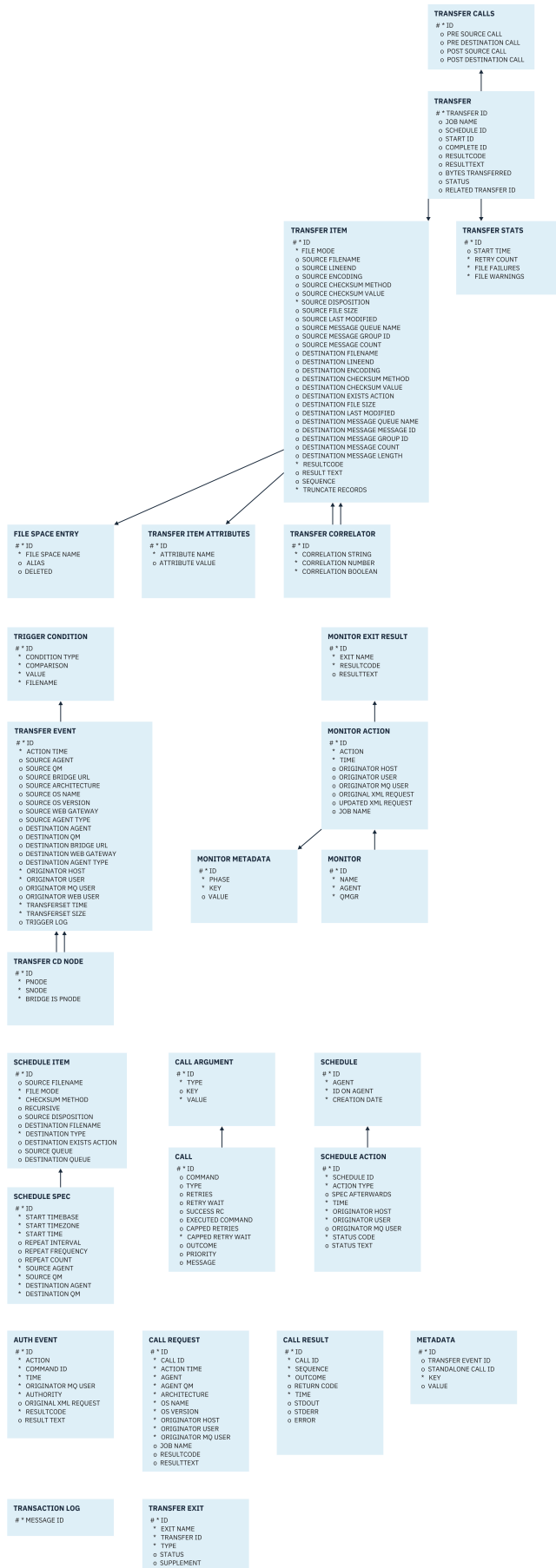
A diagram showing the relationship of the entities in an Oracle database.

In the ERD, the #, \*, and o symbols each has a specific meaning:

- # means a primary key
- \* means that a value cannot be null
- o means that a value can be null

*Figure 10. Oracle Entity Relationship Diagram (ERD)*





## Related tasks

[“Migrating an Oracle database to the new schema” on page 2546](#)

How you migrate a database with the existing schema to the new schema, by using the sample SQL script file.

## Migrating an Oracle database to the new schema

How you migrate a database with the existing schema to the new schema, by using the sample SQL script file.

### Before you begin

Take a backup of the database, and its relevant configuration information, that you are going to migrate and refer to the [“Oracle entity relationship diagram” on page 2544](#).



**Attention:** The NCLOB data type has no limit on the length of the data that can be stored. However, VARCHAR2 has a limit of 4000 bytes, so there could be some data loss while migrating to a new schema if the existing database contains file names that are longer than 4000 bytes (or 32767 bytes for an extended string).

In this situation, only the last 2000 characters of the filename will be migrated, therefore, you should ensure that your file names do not exceed 2000 characters.

### About this task

The following four sample SQL script files are located in `<MQ_Installation_Directory>/mqft/sql`:

- `oracle_nvarchar_migration_step_1.sql`
- `oracle_nvarchar_migration_step_2.sql`
- `oracle_nvarchar_migration_step_3.sql`
- `oracle_nvarchar_migration_step_4.sql`

### Procedure

1. Carry out the following tasks in order:

- a) Run **`oracle_nvarchar_migration_step_1.sql`**
- b) Run **`oracle_nvarchar_migration_step_2.sql`**
- c) Run **`oracle_nvarchar_migration_step_3.sql`**
- d) Run **`oracle_nvarchar_migration_step_4.sql`**

**Important:** Before running step [“1.c” on page 2546](#), ensure that steps [“1.a” on page 2546](#) and [“1.b” on page 2546](#) have run successfully.

2. Issue the command `cd <MQ_Installation_Directory>/mqft/sql`

3. Process the SQL script files, using the following commands in order:

- a) Run **`sqlplus USERNAME/PASSWORD < oracle_nvarchar_migration_step1.sql`**
- b) Run **`sqlplus USERNAME/PASSWORD < oracle_nvarchar_migration_step2.sql`**
- c) Run **`sqlplus USERNAME/PASSWORD < oracle_nvarchar_migration_step3.sql`**
- d) Run **`sqlplus USERNAME/PASSWORD < oracle_nvarchar_migration_step4.sql`**

where USERNAME/PASSWORD refers to the user Id and password of a particular user.

## Authorities for the MFT logger

The operating system user who runs the logger requires certain IBM MQ authorities on the logger queues and the SYSTEM.FTE topic.

The operating system user who runs the logger requires the following IBM MQ authorities:

- CONNECT and INQUIRE on the coordination queue manager.
- SUBSCRIBE permission on the SYSTEM.FTE topic.
- PUT permission on the SYSTEM.FTE.LOG.RJCT.*logger\_name* queue.
- GET permission on the SYSTEM.FTE.LOG.CMD.*logger\_name* queue.

#### Related tasks

[Restricting group authorities for MFT-specific resources](#)

[Restricting user authorities on MFT agent actions](#)

## File permissions for destination files

The file permissions for destination files written by Managed File Transfer destination agents are determined by the platform that the agent is running on.

### Destination agents on z/OS, AIX and Linux platforms



You need to alter the value of **umask** on your system.

For example, assume that the default **umask** value for your user ID on your z/OS system is *0022*.

When an MFT agent is running as this user, and writes a destination file, the file has the following permissions:

```
-IW-I--I--
```

If you change the **umask** value to, for example, *0006*, by running the command

```
umask 0006
```

and the agent restarted, then any destination files that the agent writes has the permissions:

```
-IW-IW----
```

Note, that you must restart the agent after you have run the `umask` command in order for the agent to pick up the new value.

Although z/OS is used as an example here, the same information applies to AIX and Linux platforms.

### Destination agents on Windows



By default, permissions are inherited from a root folder to the files and sub-folders beneath it, though this inheritance can be turned off.

Your Windows administrator or domain administrator should review and manage the permissions and change them if necessary. They can use the `icacls` command to view, add, update, and remove permissions.

#### Related tasks

[Restricting group authorities for MFT-specific resources](#)

[Restricting user authorities on MFT agent actions](#)

## MQ message properties set by MFT on messages written to destination queues

When transferring from file to message, Managed File Transfer can set IBM MQ message properties on the first message written to the destination queue. Additional IBM MQ message properties are set when a file to message transfer has failed.

IBM MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing MQ Message Descriptor (MQMD) or MQRFH2 headers. See [Message properties](#).

This topic describes the parameter used in the **fteCreateTransfer** and **fteCreateTemplate** commands to indicate that message properties should be added to the first message written to the destination queue. You can also specify that message properties should be added to the first message written to the destination queue using the *dstmsgprop* value of the **fte:filespec** parameter.

### Standard properties

You can use the **-qmp** parameter on the **fteCreateTransfer** command or the **fteCreateTemplate** command to specify whether IBM MQ message properties are set on the first message written to the destination queue by the transfer. For an example of how to use this parameter, see the topic [Example: Setting IBM MQ message properties on a file-to-message transfer](#)

The IBM MQ message properties contain transfer metadata. The message property names are prefixed with **usr.WMQFTE**. The **usr.** prefix makes these message properties available to JMS applications.

#### **usr.WMQFTETransferId**

The unique hexadecimal transfer ID.

#### **usr.WMQFTETransferMode**

The type of file transfer: binary mode or text mode.

#### **usr.WMQFTESourceAgent**

The name of the source agent.

#### **usr.WMQFTEDestinationAgent**

The name of the destination agent.

#### **usr.WMQFTEFileName**

The name of the source file.

#### **usr.WMQFTEFileSize**

The size of the source file in bytes.

#### **usr.WMQFTEFileLastModified**

The last modified time of the source file. This value is in units of milliseconds, measured from 00:00:00 UTC, January 1, 1970.

#### **usr.WMQFTEFileIndex**

The index of the current file in the list of files that are being transferred. The first file in the list has index 0.

#### **usr.WMQFTEMqmdUser**

The MQMD user ID of the user that submitted the transfer request.

### Failure properties

When a file to message transfer fails after the destination agent has written at least one message to the destination queue, Managed File Transfer writes a blank message to the destination queue. If the **-qmp** parameter is set to true, this blank message has two IBM MQ message properties set. For an example of a file to message transfer failure, see [Failure of a file-to-message transfer](#).

When a file to message transfer fails completely, Managed File Transfer writes a blank message to the destination queue. If the **-qmp** parameter is set to true, and the length of the message data is greater than the `maxInputOutputMessageLength` value, the following error message is displayed at the command line.

```
Name WMQFTEResultCode
Value 40
Name WMQFTESupplement
Value BFGTR0072E: The transfer failed to complete due to the exception BFGI00205E:The message
data length 1290843 being written
to the output queue "M2F@q2" is greater than the maximum allowed 1048576.
```

The IBM MQ message properties contain information about the failure. As with the standard message properties, the message property names are prefixed with **usr.WMQFTE** and are available to JMS applications.

#### **usr.WMQFTEReturnCode**

The return code of the transfer. For a list of possible values for this return code, see the topic [Return codes for MFT](#).

#### **usr.WMQFTESupplement**

A supplementary message describing in more detail why the transfer failed.

### **User-defined properties**

Metadata specified using the **-md** parameter with the **fteCreateTransfer** command can be set as IBM MQ message properties. If the **-qmp** parameter is set to true, any metadata specified by the user will be added to the message header of the first message.

The metadata name is prefixed by **usr.**. For example, if the metadata is `department=accounts`, the IBM MQ message header is set to `usr.department=accounts`.

You cannot use metadata to specify headers that begin with `usr.WMQFTE` or `usr.com.ibm.wmqfte`. If you specify metadata with a name beginning with `WMQFTE` or `com.ibm.wmqfte` this metadata is not used in the message properties and is ignored.

#### **Related concepts**

[Failure of a file-to-message transfer](#)

#### **Related tasks**

[Transferring data from files to messages](#)

#### **Related reference**

[Example: Setting IBM MQ message properties on a file-to-message transfer](#)

[“IBM MQ message properties read by MFT from messages on source queues” on page 2550](#)

The agent reading messages from a source queue in a message to file transfer reads the IBM MQ message properties from the message. The value of these properties can be used to determine the behavior of a transfer.

[Return codes for MFT](#)

[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

[fte:filespec](#)

## IBM MQ message properties read by MFT from messages on source queues

The agent reading messages from a source queue in a message to file transfer reads the IBM MQ message properties from the message. The value of these properties can be used to determine the behavior of a transfer.

### Headers used to cancel message to file transfers

Set the following IBM MQ message properties on the last message in a group to cancel the message to file transfer of that group:

#### **usr.UserReturnCode**

Required. The return code of the transfer. Set this header as a non-zero value to indicate that the transfer is to be canceled.

#### **usr.UserSupplement**

Optional. Text describing why the transfer was canceled.

If the source agent of a message to file transfer reads a message from the source queue that has the **usr.UserReturnCode** message property set to a non-zero value, it stops reading messages from the queue and reports that the transfer failed in the transfer log XML. The transfer log XML contains the return code and supplementary text that is set in the message headers. If the destination agent has already written data to a temporary file this file is deleted from the destination.

### Headers used by variable substitution

The value of any IBM MQ message property in the first message to be read from the monitored queue can be substituted into the task XML definition. User-defined message properties are prefixed with `usr.`, but do not include this prefix in the variable name. Variable names must be preceded by a dollar sign (\$) character and enclosed in braces ({}). For example, `${destFileName}` is replaced with the value of the `usr.destFileName` message property of the first message to be read from the source queue.

For example, the user or program putting messages to a monitored queue can set IBM MQ message properties on the first message in a group specifying which agent is to be used as the destination of the file transfer and what file name to transfer the data to.

For more information, see [Monitoring a queue and using variable substitution](#).

### Related tasks

[Transferring data from messages to files](#)

[Configuring an agent to perform message-to-file transfers](#)

### Related reference

Example: [Failing a message-to-file transfer using IBM MQ message properties](#)

“MQ message properties set by MFT on messages written to destination queues” on page 2548

When transferring from file to message, Managed File Transfer can set IBM MQ message properties on the first message written to the destination queue. Additional IBM MQ message properties are set when a file to message transfer has failed.

“`fteCreateTransfer` (start a new file transfer)” on page 2048

The **`fteCreateTransfer`** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

## Guidance for setting MQ attributes and MFT properties associated with message size

You can change IBM MQ attributes and Managed File Transfer properties to affect the behavior of Managed File Transfer when reading or writing messages of various sizes.

If the size of messages being read from a source queue or written to a destination queue exceeds 1048576 bytes (1 MB), you must increase the value of the Managed File Transfer Agent property

**maxInOutPutMessageLength** to a value that is greater than or equal to the maximum message size to be read or written.

If the messages on the source queue are greater than 1048576 bytes, you must set the **maxInOutPutMessageLength** property on the source agent. If the messages on the destination queue are greater than 1048576 bytes you must set the **maxInOutPutMessageLength** property on the destination agent. For more information about the **maxInOutPutMessageLength** property, see [Advanced agent properties: File to message and message to file agent](#).

- If the queue that the agent is writing to or reading from is local to the agent queue manager, you might have to change the IBM MQ queue manager, queue, and channel **MAXMSGL** attributes.

Ensure that the value of the maximum message size of the source or destination queue is greater than or equal to the value of the **maxInOutPutMessageLength** agent property.

Ensure that the value of each of the following IBM MQ attributes, in bytes:

- The maximum message size of the agent queue manager
- The maximum message size of the `SYSTEM.FTE.STATE.agent_name` queue
- The client channel maximum message size, if your agent connects to the queue manager in client mode

is greater than or equal to the result of the following calculation:

The value of  $(3 * \text{maxInOutPutMessageLength}) + 1048576$

(This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

- If the queue that the agent is writing to is a remote queue, you might have to change the IBM MQ queue manager, queue, and channel **MAXMSGL** attributes.

Ensure that the value of each of the following IBM MQ attributes is greater than or equal to the value of the **maxInOutPutMessageLength** agent property:

- The maximum message size of the remote queue manager transmission queue on the agent queue manager
- The maximum message size of the channel from the agent queue manager to the remote queue manager
- The maximum message size of the destination queue on the remote queue manager
- The maximum message size of the remote queue manager

Ensure that the value of each of the following IBM MQ attributes, in bytes:

- The maximum message size of the agent queue manager
- The maximum message size of the `SYSTEM.FTE.STATE.agent_name` queue
- The client channel maximum message size, if your agent connects to the queue manager in client mode

is greater than or equal to the result of the following calculation:

The value of  $(3 * \text{maxInOutPutMessageLength}) + 1048576$

This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.

If you exceed the value of one of these properties, the agent stops with the following error in the agent event log:

```
BFGUT0002E: An internal error has occurred. Product failure data was captured in file
"FFDC.FTE.20100928170828514.8172766022149157013.log".
BFGSS0025E: An internal error has occurred. The exception is: cc=2 rc=2010 op=put - MQPUT to
SYSTEM.FTE.STATE.agent_name
BFGAG0061E: The agent ended abnormally
```

The following IBM MQ reason codes might be included in this message in the agent event log:

- `rc=2010` This reason code maps to `MQRC_DATA_LENGTH_ERROR` and indicates that the value of the client channel maximum message size was exceeded. To resolve this problem ensure that the client channel maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

$$3 * (\text{maxInputOutputMessageLength}) + 1048576$$

- `rc=2030` This reason code maps to `MQRC_MSG_TOO_BIG_FOR_Q` and indicates that the value of the maximum message size of the `SYSTEM.FTE.STATE.agent_name` queue was exceeded. To resolve this problem ensure that the maximum message size of the `SYSTEM.FTE.STATE.agent_name` queue is greater than or equal to the result of the following calculation:

$$3 * (\text{maxInputOutputMessageLength}) + 1048576$$

- `rc=2031` This reason code maps to `MQRC_MSG_TOO_BIG_FOR_Q_MGR` and indicates that the value of the maximum message size of the agent queue manager was exceeded. To resolve this problem ensure that the maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

$$3 * (\text{maxInputOutputMessageLength}) + 1048576$$

## If you are transferring many small messages

If the average size of the messages that the agent is reading from or writing to a queue is less than 1310 bytes and the agent is reading or writing more than 10000 messages, you must increase the maximum number of uncommitted messages attribute on the queue manager or reduce the amount of data in a checkpoint interval.

When the agent is reading messages from or writing messages to a queue the corresponding **GETs** or **PUTs** are grouped together into transactions. The number of **GETs** or **PUTs** in a transaction is determined by the number required to process all of the data within a checkpoint interval. The approximate amount of the data in a checkpoint interval is determined from agent properties using the following calculation:

$$\text{Checkpoint interval data size (in bytes)} = \text{agentCheckpointInterval} * \text{agentFrameSize} * \text{agentWindowSize} * \text{agentChunkSize}.$$

The default checkpoint data size is  $1 * 5 * 10 * 262144$  bytes = 13107200 bytes (12.5MB). The maximum number of uncommitted messages in a transaction that a queue manager supports is controlled by the **MaxUncommittedMsgs** queue manager attribute. The default value of this attribute is 10000 messages. If the average message size is less than approximately 1310 bytes the default maximum number of uncommitted messages is exceeded if there are more than 10000 messages to be written.

If you exceed the **MaxUncommittedMsgs** limit, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2024' from the message queue interface (MQI).
The agent cannot continue processing and will now end.
BFGAG0139I: The agent has suspended its current transfers and is now stopping.
```

The reason code 2024 maps to: `MQRC_SYNCPOINT_LIMIT_REACHED`.

To resolve this problem perform one of the following actions

- Increase the value of the **MaxUncommittedMsgs** queue manager attribute of the queue manager that the agent reading from or writing to a queue connects to. See [MaxUncommittedMsgs \(MQLONG\)](#).
- Reduce the amount of data in a checkpoint interval. To do this, decrease the value of one or more of the following agent properties:
  - `agentCheckpointInterval`



- agentFrameSize
- agentWindowSize
- agentChunkSize

For information about these agent properties, see [Advanced agent properties](#).

## If you are writing messages to a queue persistently

If you are transferring to a queue and writing the messages to the queue persistently, you might have to increase the size of the queue manager log file space to be able to log all of the data in a checkpoint interval.

If you exceed the queue manager log file space, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2102' from the message queue interface (MQI).
The agent cannot continue processing and will now end.
BFGAG0062E: The agent has received MQI reason code '2102'. The agent cannot continue processing and
will now end.
BFGAG0061E: The agent ended abnormally
```

The reason code '2102' maps to: MQRC\_RESOURCE\_PROBLEM.

To resolve this problem increase the size of the destination agent queue manager log file space.

### Related tasks

[Transferring data from messages to files](#)

[Transferring data from files to messages](#)

### Related reference

[The MFT agent.properties file](#)

## Guidance for specifying a wait time on a message-to-file transfer

When specifying a message-to-file transfer you can optionally specify a wait time on the transfer using the **-sqwt** parameter. The value of **-sqwt** is the amount of time that the source agent waits either for a message to appear on the source queue if the source queue is empty or becomes empty, or for a complete group to appear on the source queue if the **-sqgi** attribute is specified.

This topic describes the parameters used in the **fteCreateTransfer** command for specifying a wait time. You can also specify the wait time using the [srcqueuetimeout](#) value of the **fte:filespec** parameter.

If the value of the **-sqwt** parameter is greater than or equal to the amount of time the destination agent waits for the transfer to be completed by the source agent, the transfer does not complete. The amount of time the destination agent waits for the transfer to complete is given by the following calculation:

```
transferAckTimeout * transferAckTimeoutRetries
```

The properties `transferAckTimeout` and `transferAckTimeoutRetries` are set in the destination agent `agent.properties` file. For more information about these agent properties, see [The agent.properties file](#).

To prevent transfers from failing to complete, you must perform one of the following steps:

- Reduce the value of the **-sqwt** parameter so that it is less than the value of the destination agent `transferAckTimeout` property.

**Note:** The default value of the `transferAckTimeout` property is 60,000 milliseconds. The value of the **-sqwt** parameter is given in seconds, set the value to 59 or less.

- Increase the value of the destination agent `transferAckTimeout` property so that it is greater than the value of the **-sqwt** parameter.

**Note:** The value of the `transferAckTimeout` property is given in milliseconds. The value of the `-sqwt` parameter is given in seconds.

### Related reference

“`fteCreateTransfer (start a new file transfer)`” on page 2048

The **`fteCreateTransfer`** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

The `agent.properties` file

`fte:filespec`

## Available code pages for MFT

This reference topic lists all character encoding formats available for text file conversion on the various platforms supported by Managed File Transfer.

### Common encodings

These character encoding formats are available on all supported platforms. If your source file is encoded using one of the formats in this table, and you want to use another of the formats in this table to write the destination file, you can do so without any consideration of platform. You can use either the canonical name or any of the aliases to specify an encoding format.

Canonical name	Aliases
<b>windows-1256</b>	ibm-1256, Cp1256
<b>windows-1255</b>	ibm-1255, Cp1255
<b>windows-1254</b>	Cp1254, ibm-1254
<b>windows-1253</b>	Cp1253, ibm-1253
<b>windows-1252</b>	ibm-1252, Cp1252
<b>windows-1251</b>	ibm-1251, Cp1251
<b>windows-1250</b>	Cp1250, ibm-1250
<b>UTF-8</b>	UTF_8, UTF8
<b>UTF-16LE</b>	X-UTF-16LE, UTF16LE, UTF_16LE, UnicodeLittleUnmarked
<b>UTF-16BE</b>	UTF16BE, UnicodeBigUnmarked, ISO-10646-UCS-2, UTF_16BE, X-UTF-16BE
<b>US-ASCII</b>	Cp367, iso-ir-6, ANSI_X3.4-1968, ANSI_X3.4-1986, default, ASCII, us, iso-646.irv:1983, csASCII, 646, ascii7, ISO646-US, ibm-367, ISO-646.irv:1991, direct
<b>TIS-620</b>	tis620, tis620.2533
<b>IBM-1122</b>	Cp1122, ibm1122
<b>IBM-1006</b>	Cp1006, ibm1006
<b>IBM-037</b>	ibm-37
<b>GB18030</b>	windows-54936, gb18030-2000, ibm-1392
<b>EUC-TW</b>	x-euc-tw, euctw, cns11643, euc_tw
<b>EUC-KR</b>	ibm-euckr, euc_kr, ksc_5601, ks_c_5601-1987, ksc5601_1987, euckr, ksc5601-1987, ibm-970, Cp970, 5601
<b>EUC-JP</b>	x-euc-jp, euc_jp, eucjp, x-eucjp, euc_jp_linux, euc-jp-linux
<b>EUC-CN</b>	x-euc-cn, ibm-euccn, euc_cn, euccn
<b>Big5</b>	big5-0, big5, Big5-HKSCS
<b>IBM-1025</b>	Cp1025, ibm1025

Table 373. Character encoding formats available on all supported platforms (continued)

Canonical name	Aliases
<b>IBM-1026</b>	ibm1026, Cp1026
<b>IBM-1046</b>	Cp1046, ibm1046
<b>IBM-1097</b>	Cp1097, ibm1097
<b>IBM-1098</b>	Cp1098, ibm1098
<b>IBM-1112</b>	ibm1112, Cp1112
<b>IBM-1383</b>	Cp1383, ibm1383
<b>IBM-273</b>	Cp273, ibm273
<b>IBM-277</b>	Cp277, ibm277
<b>IBM-278</b>	Cp278, ibm278
<b>IBM-280</b>	ibm280, Cp280
<b>IBM-284</b>	ibm284, Cp284
<b>IBM-285</b>	Cp285, ibm285
<b>IBM-297</b>	ibm297, Cp297
<b>IBM-420</b>	Cp420, ibm420
<b>IBM-860</b>	Cp860, ibm860
<b>IBM-861</b>	ibm861, Cp861
<b>IBM-862</b>	Cp862, ibm862
<b>IBM-863</b>	Cp863, ibm863
<b>IBM-864</b>	Cp864, ibm864
<b>IBM-865</b>	ibm865, Cp865
<b>windows-1257</b>	Cp1257, ibm-1257
<b>windows-1258</b>	Cp1258, ibm-1129, ibm-1258
<b>windows-31j</b>	ms_kanji, cswindows31j, MS932, windows-932
<b>windows-874</b>	MS874
<b>windows-936</b>	MS936, x-mswin-936, 936
<b>windows-949</b>	MS949, Cp1361, ibm-1361, ibm1361, ms1361, ksc5601-1992, x-windows-949
<b>windows-950</b>	MS950, x-windows-950
<b>IBM-857</b>	ibm857, Cp857, csibm857
<b>IBM-856</b>	Cp856, ibm856
<b>IBM-855</b>	Cp855, ibm855
<b>IBM-852</b>	cspcp852, ibm852, Cp852
<b>IBM-850</b>	Cp850, ibm850, cspc850multilingual
<b>IBM-838</b>	Cp838, ibm838
<b>IBM-834</b>	Cp834, ibm834
<b>IBM-775</b>	ibm775, Cp775
<b>IBM-737</b>	Cp737, ibm737
<b>IBM-500</b>	Cp500, ibm500
<b>IBM-437</b>	ibm437, Cp437, cspc8codepage437
<b>IBM-424</b>	ibm424, Cp424

Table 373. Character encoding formats available on all supported platforms (continued)

Canonical name	Aliases
<b>IBM-1123</b>	Cp1123, ibm1123
<b>IBM-1124</b>	Cp1124, ibm1124
<b>IBM-1381</b>	Cp1381, ibm1381
<b>IBM-866</b>	Cp866, ibm866
<b>IBM-868</b>	Cp868, ibm868
<b>IBM-869</b>	ibm869, Cp869
<b>IBM-870</b>	Cp870, ibm870
<b>IBM-871</b>	ibm871, Cp871
<b>IBM-874</b>	ibm874, Cp874
<b>IBM-875</b>	Cp875, ibm875
<b>IBM-921</b>	Cp921, ibm921
<b>IBM-922</b>	Cp922, ibm922
<b>IBM-933</b>	Cp933, ibm933
<b>IBM-935</b>	Cp935, ibm935
<b>IBM-937</b>	Cp937, ibm937
<b>IBM-942</b>	Cp942, ibm942
<b>IBM-943</b>	Cp943, ibm943
<b>IBM-948</b>	ibm948, Cp948
<b>IBM-949</b>	ibm949, Cp949
<b>IBM-950</b>	ibm950, Cp950
<b>ISCI91</b>	iscii
<b>ISO-2022-CN</b>	iso2022-cn-cns, iso2022cn-cns, iso-2022-cn-cns, iso2022cn, iso2022-cn
<b>ISO-2022-CN-GB</b>	iso2022-cn-gb, iso2022cn-gb
<b>ISO-2022-JP</b>	iso2022jp, jis, iso2022-jp, iso-2022-jp2, csiso2022jp2, csjisencoding, jis-encoding
<b>ISO-2022-KR</b>	csiso2022kr, iso2022-kr, iso2022kr
<b>ISO-8859-1</b>	iso8859_1, iso8859-1, ibm819, l1, csisolatin1, Cp819, iso-ir-100, iso-8859-1:1987, ibm-819, latin1, 8859-1
<b>ISO-8859-13</b>	iso8859-13, 8859-13, iso8859_13
<b>ISO-8859-15</b>	csisolatin9, iso8859-15, ibm923, latin9, ibm-923, l9, iso8859_15, iso8859_15_fdis, Cp923, latin0
<b>ISO-8859-2</b>	Cp912, ibm912, iso8859-2, iso-8859-2:1987, l2, iso8859_2, csisolatin2, latin2, ibm-912, 8859-2, iso-ir-101
<b>ISO-8859-3</b>	iso8859-3, Cp913, l3, iso8859_3, iso-ir-109, iso-8859-3:1988, latin3, ibm-913, 8859-3, csisolatin3
<b>ISO-8859-4</b>	Cp914, latin4, iso8859_4, l4, iso-8859-4:1988, ibm-914, iso8859-4, 8859-4, csisolatin4, iso-ir-110
<b>ISO-8859-5</b>	csisolatincyrillic, iso-ir-144, cyrillic, iso8859_5, iso-8859-5:1988, ibm-915, 8859-5, Cp915, ibm915, iso8859-5
<b>ISO-8859-6</b>	csisolatinarabic, Cp1089, iso-8859-6:1987, ecma-114, iso-ir-127, asmo-708, iso8859_6, 8859-6, ibm1089, arabic, iso8859-6, ibm-1089
<b>ISO-8859-7</b>	ecma-118, ibm813, csisolatingreek, elot-928, iso-ir-126, Cp813, 8859-7, iso-8859-7:1987, iso8859_7, greek, greek8, ibm-813, iso8859-7

Table 373. Character encoding formats available on all supported platforms (continued)









Canonical name	Aliases
<b>ISO-8859-8</b>	iso-ir-138, iso-8859-8:1988, csisolatinhebrew, hebrew, iso8859-8, 8859-8, ibm-916, iso8859_8, Cp916, ibm916
<b>ISO-8859-9</b>	ibm-920, ibm920, latin5, 8859-9, Cp920, l5, iso8859-9, iso8859_9, csisolatin5, iso-ir-148
<b>JIS0212</b>	
<b>KOI8-R</b>	koi8, ibm-878, cskoi8r, koi8_r
<b>MacArabic</b>	
<b>MacCentralEurope</b>	ibm-1282
<b>MacCroatian</b>	ibm-1284
<b>MacCyrillic</b>	ibm-1283
<b>MacGreek</b>	ibm-1280
<b>MacIceland</b>	ibm-1286
<b>MacRoman</b>	ibm-1275
<b>MacRomania</b>	ibm-1285
<b>MacSymbol</b>	Adobe-Symbol-Encoding, ibm-1038
<b>MacTurkish</b>	ibm-1281

## Source platform default encodings

If you do not specify an encoding for the source file or for the destination file, the default encoding for that platform will be used. The conversion is performed by the destination agent, and both source and destination encodings must be supported on the destination agent's platform for the conversion to take place. The destination default encoding will always be supported on the destination agent, so it is always safe to leave this unspecified. However, it might not be safe to use a default source encoding, because the destination agent might not support the source's default.

If you are using default source encodings, use the tables in this topic to make sure that the combination will be supported.

Table 374. Default encodings

Platform	Default encoding
 Linux SUSE Linux Enterprise Server on x86-64	UTF-8
 IBM i IBM i	ISO-8859-1
 Linux Linux for IBM Z	UTF-8
 AIX AIX	ISO-8859-1
 Windows Windows	windows-1252
 Linux Red Hat® Enterprise Linux on x86-64	UTF-8
 z/OS z/OS	IBM-1047
 Linux Linux on POWER Systems - Big Endian	UTF-8
HP (PA-RISC)	ISO-8859-1

## **Platform-specific encodings**

**Note:** The following two tables contain the same information. It is organized in two different ways to help you find the correct information, depending whether you are looking up by platform or by encoding.

### **Encodings by Platform**

Canonical names are listed in bold, followed by aliases in parentheses.

Platforms that support only encodings already listed in the Common Encodings table are not listed here.

Table 375. Platform-specific encodings by platform

Platform	Supported encodings (not in common encodings table)
<p><b>Linux</b> SUSE Linux Enterprise Server on x86-64</p>	<p><b>windows-1256S</b> (Cp1256s, ibm-1256s)  <b>UTF-8J</b> (UTF8J)  <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>UTF-32</b> (UCS-4, UTF32, ISO-10646-UCS-4)  <b>PTCP154</b> (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)  <b>KOI8-RU</b> (ibm-1168, koi8_ru)  <b>ISO-8859-16</b> (8859-16, iso8859_16, iso8859-16)  <b>ISO-8859-14</b> (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)  <b>IBM01141</b> (cp1141, ccsid01141, cp01141, 1141)  <b>IBM01142</b> (cp01142, cp1142, 1142, ccsid01142)  <b>IBM01143</b> (cp01143, 1143, ccsid01143, cp1143)  <b>IBM01144</b> (cp01144, cp1144, ccsid01144, 1144)  <b>IBM01145</b> (cp1145, cp01145, ccsid01145, 1145)  <b>IBM01146</b> (ccsid01146, cp01146, cp1146, 1146)  <b>IBM01147</b> (ccsid01147, cp1147, 1147, cp01147)  <b>IBM01148</b> (cp1148, ccsid01148, 1148, cp01148)  <b>IBM01149</b> (cp1149, cp01149, ccsid01149, 1149)  <b>IBM1047</b> (cp1047, 1047, ibm-1047)  <b>IBM918</b> (cp918, ebcdic-cp-ar2, ibm-918, 918)  <b>ISO-2022-JP-2</b> (csISO2022JP2, iso2022jp2)  <b>x-Big5-Solaris</b> (Big5_Solaris)  <b>x-eucJP-Open</b> (EUC_JP_Solaris, eucJP-open)  <b>x-IBM33722</b> (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722)  <b>x-IBM930</b> (cp930, ibm930, ibm-930, 930)  <b>x-IBM939</b> (ibm-939, ibm939, cp939, 939)  <b>x-IBM964</b> (964, cp964, ibm-964, ibm964)  <b>x-ISO-2022-CN-CNS</b> (ISO-2022-CN-CNS, ISO2022CN_CNS)  <b>x-iso-8859-11</b> (iso-8859-11, iso8859_11)  <b>x-JISAutoDetect</b> (JISAutoDetect)  <b>x-MS932_0213</b> ()  <b>x-MS950-HKSCS</b> (MS950_HKSCS)  <b>x-PCK</b> (pck)  <b>x-IBM1363C</b> (ibm1363c, cp1363c, ibm-1363c)  <b>x-IBM420S</b> (420s, ibm-420s, csibm420s, ibm420s, cp420s)  <b>x-IBM864S</b> (csibm864s, ibm864s, cp864s, 864s, ibm-864s)  <b>x-IBM943C</b> (cp943c, 943c, ibm-943c, ibm943c)  <b>x-IBM949C</b> (ibm949c, cp949c, 949c, ibm-949c)  <b>x-IBM954C</b> (cp954c, 954c, ibm-954c, ibm954c)  <b>x-ISO-8859-6S</b> (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s)  <b>x-JIS0208</b> (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208)  <b>x-KSC5601</b> (ksc5601)  <b>x-MacDingbat</b> (macdingbat)  <b>x-MacHebrew</b> (machebrew)  <b>x-MacThai</b> (macthai)  <b>x-MacUkraine</b> (macukraine)  <b>x-IBM1046S</b> (ibm-1046s, 1046s, cp1046s, ibm1046s)  <b>x-IBM-udcJP</b> (IBM-udcJP)  <b>JIS_X0201</b> (jis_x0201, x0201, cshalfwidthkatakana, jis0201)  <b>IBM-939A</b> (Cp939A, ibm939A)  <b>IBM-930A</b> (ibm930A, Cp930A)  <b>IBM-33722A</b> (Cp33722A, ibm33722A)  <b>x-windows-iso2022jp</b> (windows-iso2022jp)  <b>x-windows-50221</b> (ms50221, cp50221)  <b>x-windows-50220</b> (cp50220, ms50220)  <b>X-UTF-32LE-BOM</b> (UTF_32LE_BOM, UTF-32LE-BOM)  <b>X-UTF-32BE-BOM</b> (UTF_32BE_BOM, UTF-32BE-BOM)  <b>x-SJIS_0213</b> ()  <b>IBM01140</b> (ccsid01140, cp01140, 1140, cp1140)  <b>IBM00858</b> (cp858, ccsid00858, 858, cp00858)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>IBM-859</b> (Cp859, ibm859)</p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>Linux</b> SUSE Linux Enterprise Server on x86-64</p>	<p> <b>IBM-837</b> (ibm837, Cp837)  <b>IBM-836</b> (ibm836, Cp836)  <b>IBM-835</b> (ibm835, Cp835)  <b>IBM-833</b> (ibm833, Cp833)  <b>IBM-808</b> (Cp808, ibm808)  <b>IBM-720</b> (Cp720, ibm720)  <b>IBM-33722C</b> (ibm-eucjp, Cp33722c)  <b>IBM-301</b> (Cp301, ibm301)  <b>IBM-300</b> (Cp300, ibm300)  <b>IBM-290</b> (ibm290, Cp290)  <b>IBM-1399</b> (ibm1399, Cp1399)  <b>IBM-1390</b> (Cp1390, ibm1390)  <b>IBM-1388</b> (Cp1388, ibm1388)  <b>IBM-1385</b> (Cp1385, ibm1385)  <b>IBM-1382</b> (ibm1382, Cp1382)  <b>IBM-1088</b> (Cp1088, ibm1088)  <b>IBM-1043</b> (Cp1043, ibm1043)  <b>IBM-1041</b> (Cp1041, ibm1041)  <b>IBM-1027</b> (Cp1027, ibm1027)  <b>CESU-8</b> (CESU8)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>GB2312</b> (gb2312-1980, gb2312-80)  <b>GBK</b> (GBK)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>IBM-1114</b> (Cp1114, ibm1114)  <b>IBM-1115</b> (Cp1115, ibm1115)  <b>IBM-1351</b> (Cp1351, ibm1351)  <b>IBM-1362</b> (Cp1362, ibm1362)  <b>IBM-1363</b> (ibm1363, Cp1363)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-1370</b> (Cp1370, ibm1370)  <b>IBM-1371</b> (Cp1371, ibm1371)  <b>IBM-1380</b> (Cp1380, ibm1380)  <b>IBM-867</b> (Cp867, ibm867)  <b>IBM-897</b> (Cp897, ibm897)  <b>IBM-924</b> (Cp924, ibm924)  <b>IBM-927</b> (ibm927, Cp927)  <b>IBM-932</b> (ibm932, Cp932)  <b>IBM-947</b> (Cp947, ibm947)  <b>IBM-951</b> (Cp951, ibm951)  <b>IBM-954</b> (ibm954, Cp954)  <b>IBM-971</b> (Cp971, ibm971)  <b>ISO-8859-10</b> (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)                 </p>



Table 375. Platform-specific encodings by platform (continued)


Platform	Supported encodings (not in common encodings table)
 IBM i	<p> <b>windows-1256S</b> (Cp1256s, ibm-1256s)  <b>UTF-8J</b> (UTF8J)  <b>IBM-1146</b> (Cp1146, ibm1146)  <b>IBM-1145</b> (Cp1145, ibm1145)  <b>IBM-1144</b> (ibm1144, Cp1144)  <b>IBM-1143</b> (Cp1143, ibm1143)  <b>IBM-1142</b> (Cp1142, ibm1142)  <b>IBM-1141</b> (Cp1141, ibm1141)  <b>IBM-1140</b> (ibm1140, Cp1140)  <b>IBM-1115</b> (Cp1115, ibm1115)  <b>IBM-1114</b> (Cp1114, ibm1114)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>GBK</b> (GBK)  <b>GB2312</b> (gb2312-1980, gb2312-80)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>CESU-8</b> (CESU8)  <b>IBM-1027</b> (Cp1027, ibm1027)  <b>IBM-1041</b> (Cp1041, ibm1041)  <b>IBM-1043</b> (Cp1043, ibm1043)  <b>IBM-1046S</b> (ibm1046S, Cp1046S)  <b>IBM-1047</b> (Cp1047, ibm1047)  <b>IBM-1088</b> (Cp1088, ibm1088)  <b>IBM-1382</b> (ibm1382, Cp1382)  <b>IBM-1385</b> (Cp1385, ibm1385)  <b>IBM-1386</b> (ibm1386, Cp1386)  <b>IBM-1388</b> (Cp1388, ibm1388)  <b>IBM-836</b> (ibm836, Cp836)  <b>IBM-837</b> (ibm837, Cp837)  <b>IBM-858</b> (Cp858, ibm858)  <b>IBM-859</b> (Cp859, ibm859)  <b>IBM-864S</b> (ibm864S, Cp864S)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>IBM-1047_LF</b> (Cp1047_LF, ibm1047_LF)  <b>IBM-1141_LF</b> (Cp1141_LF, ibm1141_LF)  <b>IBM-33722A</b> (Cp33722A, ibm33722A)  <b>IBM-924_LF</b> (Cp924_LF, ibm924_LF)  <b>IBM-930A</b> (ibm930A, Cp930A)  <b>IBM-939A</b> (Cp939A, ibm939A)  <b>IBM-835</b> (ibm835, Cp835)  <b>IBM-833</b> (ibm833, Cp833)  <b>IBM-808</b> (Cp808, ibm808)  <b>IBM-720</b> (Cp720, ibm720)  <b>IBM-420S</b> (Cp420S, ibm420S)  <b>IBM-33722C</b> (ibm-eucjp, Cp33722c)  <b>IBM-33722</b> (5050, Cp5050)  <b>IBM-301</b> (Cp301, ibm301)  <b>IBM-300</b> (Cp300, ibm300)  <b>IBM-290</b> (ibm290, Cp290)  <b>IBM-1399</b> (ibm1399, Cp1399)  <b>IBM-1390</b> (Cp1390, ibm1390)  <b>IBM-1147</b> (Cp1147, ibm1147)  <b>IBM-1148</b> (ibm1148, Cp1148)  <b>IBM-1149</b> (Cp1149, ibm1149)  <b>IBM-1351</b> (Cp1351, ibm1351)  <b>IBM-1362</b> (Cp1362, ibm1362)  <b>IBM-1363</b> (ibm1363, Cp1363)  <b>IBM-1363C</b> (ibm1363C, Cp1363C)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-1370</b> (Cp1370, ibm1370)  <b>IBM-1371</b> (Cp1371, ibm1371)  <b>IBM-1380</b> (Cp1380, ibm1380)  <b>IBM-867</b> (Cp867, ibm867)  <b>IBM-897</b> (Cp897, ibm897)         </p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>IBM i</b> IBM i</p>	<p> <b>IBM-918</b> (ibm918, Cp918)  <b>IBM-924</b> (Cp924, ibm924)  <b>IBM-927</b> (ibm927, Cp927)  <b>IBM-930</b> (Cp5026, 5026)  <b>IBM-932</b> (ibm932, Cp932)  <b>IBM-939</b> (Cp5035, 5035)  <b>IBM-942C</b> (Cp942C, ibm942C)  <b>IBM-943C</b> (ibm943C, Cp943C)  <b>IBM-947</b> (Cp947, ibm947)  <b>IBM-949C</b> (Cp949C, ibm949C)  <b>IBM-951</b> (Cp951, ibm951)  <b>IBM-954</b> (ibm954, Cp954)  <b>IBM-954C</b> (Cp954c)  <b>IBM-964</b> (ibm-euctw, Cp964)  <b>IBM-971</b> (Cp971, ibm971)  <b>ISO-8859-10</b> (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)  <b>ISO-8859-14</b> (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)  <b>ISO-8859-16</b> (8859-16, iso8859_16, iso8859-16)  <b>ISO-8859-6S</b> (iso8859-6S, iso8859_6S)  <b>JIS0201</b> ()  <b>JIS0208</b> ()  <b>Johab</b> (x-johab)  <b>KOI8-RU</b> (ibm-1168, koi8_ru)  <b>KOI8-U</b> (koi8_u, ibm-1167)  <b>KSC5601</b> ()  <b>MacDingbat</b> ()  <b>MacHebrew</b> ()  <b>MacThai</b> ()  <b>MacUkraine</b> ()  <b>PTCP154</b> (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)  <b>Shift_JIS</b> ()  <b>UTF-16</b> (UTF16, Unicode, UTF_16, UCS-2)  <b>UTF-32</b> (UCS-4, UTF32, ISO-10646-UCS-4)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)                 </p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>Linux</b> Linux for IBM Z</p>	<p><b>windows-1256S</b> (Cp1256s, ibm-1256s)  <b>UTF-8J</b> (UTF8J)  <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>UTF-32</b> (UCS-4, UTF32, ISO-10646-UCS-4)  <b>PTCP154</b> (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)  <b>KOI8-RU</b> (ibm-1168, koi8_ru)  <b>ISO-8859-16</b> (8859-16, iso8859_16, iso8859-16)  <b>ISO-8859-14</b> (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)  <b>IBM01141</b> (cp1141, ccsid01141, cp01141, 1141)  <b>IBM01142</b> (cp01142, cp1142, 1142, ccsid01142)  <b>IBM01143</b> (cp01143, 1143, ccsid01143, cp1143)  <b>IBM01144</b> (cp01144, cp1144, ccsid01144, 1144)  <b>IBM01145</b> (cp1145, cp01145, ccsid01145, 1145)  <b>IBM01146</b> (ccsid01146, cp01146, cp1146, 1146)  <b>IBM01147</b> (ccsid01147, cp1147, 1147, cp01147)  <b>IBM01148</b> (cp1148, ccsid01148, 1148, cp01148)  <b>IBM01149</b> (cp1149, cp01149, ccsid01149, 1149)  <b>IBM1047</b> (cp1047, 1047, ibm-1047)  <b>IBM918</b> (cp918, ebcdic-cp-ar2, ibm-918, 918)  <b>ISO-2022-JP-2</b> (csISO2022JP2, iso2022jp2)  <b>x-Big5-Solaris</b> (Big5_Solaris)  <b>x-eucJP-Open</b> (EUC_JP_Solaris, eucJP-open)  <b>x-IBM33722</b> (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722)  <b>x-IBM930</b> (cp930, ibm930, ibm-930, 930)  <b>x-IBM939</b> (ibm-939, ibm939, cp939, 939)  <b>x-IBM964</b> (964, cp964, ibm-964, ibm964)  <b>x-ISO-2022-CN-CNS</b> (ISO-2022-CN-CNS, ISO2022CN_CNS)  <b>x-iso-8859-11</b> (iso-8859-11, iso8859_11)  <b>x-JISAutoDetect</b> (JISAutoDetect)  <b>x-MS932_0213</b> ()  <b>x-MS950-HKSCS</b> (MS950_HKSCS)  <b>x-PCK</b> (pck)  <b>x-IBM1363C</b> (ibm1363c, cp1363c, ibm-1363c)  <b>x-IBM420S</b> (420s, ibm-420s, csibm420s, ibm420s, cp420s)  <b>x-IBM864S</b> (csibm864s, ibm864s, cp864s, 864s, ibm-864s)  <b>x-IBM943C</b> (cp943c, 943c, ibm-943c, ibm943c)  <b>x-IBM949C</b> (ibm949c, cp949c, 949c, ibm-949c)  <b>x-IBM954C</b> (cp954c, 954c, ibm-954c, ibm954c)  <b>x-ISO-8859-6S</b> (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s)  <b>x-JIS0208</b> (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208)  <b>x-KSC5601</b> (ksc5601)  <b>x-MacDingbat</b> (macdingbat)  <b>x-MacHebrew</b> (machebrew)  <b>x-MacThai</b> (macthai)  <b>x-MacUkraine</b> (macukraine)  <b>x-IBM1046S</b> (ibm-1046s, 1046s, cp1046s, ibm1046s)  <b>x-IBM-udcJP</b> (IBM-udcJP)  <b>JIS_X0201</b> (jis_x0201, x0201, cshalfwidthkatakana, jis0201)  <b>IBM-939A</b> (Cp939A, ibm939A)  <b>IBM-930A</b> (ibm930A, Cp930A)  <b>IBM-33722A</b> (Cp33722A, ibm33722A)  <b>x-windows-iso2022jp</b> (windows-iso2022jp)  <b>x-windows-50221</b> (ms50221, cp50221)  <b>x-windows-50220</b> (cp50220, ms50220)  <b>X-UTF-32LE-BOM</b> (UTF_32LE_BOM, UTF-32LE-BOM)  <b>X-UTF-32BE-BOM</b> (UTF_32BE_BOM, UTF-32BE-BOM)  <b>x-SJIS_0213</b> ()  <b>IBM01140</b> (ccsid01140, cp01140, 1140, cp1140)  <b>IBM00858</b> (cp858, ccsid00858, 858, cp00858)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>IBM-859</b> (Cp859, ibm859)</p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>Linux</b> Linux for IBM Z</p>	<p> <b>IBM-837</b> (ibm837, Cp837)  <b>IBM-836</b> (ibm836, Cp836)  <b>IBM-835</b> (ibm835, Cp835)  <b>IBM-833</b> (ibm833, Cp833)  <b>IBM-808</b> (Cp808, ibm808)  <b>IBM-720</b> (Cp720, ibm720)  <b>IBM-33722C</b> (ibm-eucjp, Cp33722c)  <b>IBM-301</b> (Cp301, ibm301)  <b>IBM-300</b> (Cp300, ibm300)  <b>IBM-290</b> (ibm290, Cp290)  <b>IBM-1399</b> (ibm1399, Cp1399)  <b>IBM-1390</b> (Cp1390, ibm1390)  <b>IBM-1388</b> (Cp1388, ibm1388)  <b>IBM-1385</b> (Cp1385, ibm1385)  <b>IBM-1382</b> (ibm1382, Cp1382)  <b>IBM-1088</b> (Cp1088, ibm1088)  <b>IBM-1043</b> (Cp1043, ibm1043)  <b>IBM-1041</b> (Cp1041, ibm1041)  <b>IBM-1027</b> (Cp1027, ibm1027)  <b>CESU-8</b> (CESU8)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>GB2312</b> (gb2312-1980, gb2312-80)  <b>GBK</b> (GBK)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>IBM-1114</b> (Cp1114, ibm1114)  <b>IBM-1115</b> (Cp1115, ibm1115)  <b>IBM-1351</b> (Cp1351, ibm1351)  <b>IBM-1362</b> (Cp1362, ibm1362)  <b>IBM-1363</b> (ibm1363, Cp1363)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-1370</b> (Cp1370, ibm1370)  <b>IBM-1371</b> (Cp1371, ibm1371)  <b>IBM-1380</b> (Cp1380, ibm1380)  <b>IBM-867</b> (Cp867, ibm867)  <b>IBM-897</b> (Cp897, ibm897)  <b>IBM-924</b> (Cp924, ibm924)  <b>IBM-927</b> (ibm927, Cp927)  <b>IBM-932</b> (ibm932, Cp932)  <b>IBM-947</b> (Cp947, ibm947)  <b>IBM-951</b> (Cp951, ibm951)  <b>IBM-954</b> (ibm954, Cp954)  <b>IBM-971</b> (Cp971, ibm971)  <b>ISO-8859-10</b> (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)                 </p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>AIX</b> AIX</p>	<p><b>windows-1256S</b> (Cp1256s, ibm-1256s)  <b>UTF-8J</b> (UTF8J)  <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>UTF-32</b> (UCS-4, UTF32, ISO-10646-UCS-4)  <b>UTF-16</b> (UTF16, Unicode, UTF_16, UCS-2)  <b>Shift_JIS</b> ()  <b>PTCP154</b> (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)  <b>MacUkraine</b> ()  <b>MacThai</b> ()  <b>MacHebrew</b> ()  <b>MacDingbat</b> ()  <b>KSC5601</b> ()  <b>KOI8-U</b> (koi8_u, ibm-1167)  <b>KOI8-RU</b> (ibm-1168, koi8_ru)  <b>Johab</b> (x-johab)  <b>JIS0208</b> ()  <b>JIS0201</b> ()  <b>ISO-8859-6S</b> (iso8859-6S, iso8859_6S)  <b>ISO-8859-16</b> (8859-16, iso8859_16, iso8859-16)  <b>ISO-8859-14</b> (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)  <b>ISO-8859-10</b> (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)  <b>IBM-971</b> (Cp971, ibm971)  <b>IBM-964</b> (ibm-euctw, Cp964)  <b>IBM-954C</b> (Cp954c)  <b>IBM-954</b> (ibm954, Cp954)  <b>IBM-951</b> (Cp951, ibm951)  <b>IBM-949C</b> (Cp949C, ibm949C)  <b>IBM-947</b> (Cp947, ibm947)  <b>IBM-943C</b> (ibm943C, Cp943C)  <b>IBM-942C</b> (Cp942C, ibm942C)  <b>IBM-939</b> (Cp5035, 5035)  <b>IBM-932</b> (ibm932, Cp932)  <b>IBM-930</b> (Cp5026, 5026)  <b>IBM-927</b> (ibm927, Cp927)  <b>IBM-924</b> (Cp924, ibm924)  <b>IBM-918</b> (ibm918, Cp918)  <b>IBM-897</b> (Cp897, ibm897)  <b>IBM-867</b> (Cp867, ibm867)  <b>IBM-1380</b> (Cp1380, ibm1380)  <b>IBM-1371</b> (Cp1371, ibm1371)  <b>IBM-1370</b> (Cp1370, ibm1370)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-1363C</b> (ibm1363C, Cp1363C)  <b>IBM-1047</b> (Cp1047, ibm1047)  <b>IBM-1088</b> (Cp1088, ibm1088)  <b>IBM-1382</b> (ibm1382, Cp1382)  <b>IBM-1385</b> (Cp1385, ibm1385)  <b>IBM-1386</b> (ibm1386, Cp1386)  <b>IBM-1388</b> (Cp1388, ibm1388)  <b>IBM-1390</b> (Cp1390, ibm1390)  <b>IBM-1399</b> (ibm1399, Cp1399)  <b>IBM-290</b> (ibm290, Cp290)  <b>IBM-300</b> (Cp300, ibm300)  <b>IBM-301</b> (Cp301, ibm301)  <b>IBM-33722</b> (5050, Cp5050)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>IBM-864S</b> (ibm864S, Cp864S)  <b>IBM-859</b> (Cp859, ibm859)  <b>IBM-858</b> (Cp858, ibm858)</p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>AIX</b> AIX</p>	<p> <b>IBM-837</b> (ibm837, Cp837)  <b>IBM-836</b> (ibm836, Cp836)  <b>IBM-835</b> (ibm835, Cp835)  <b>IBM-833</b> (ibm833, Cp833)  <b>IBM-808</b> (Cp808, ibm808)  <b>IBM-720</b> (Cp720, ibm720)  <b>IBM-420S</b> (Cp420S, ibm420S)  <b>IBM-33722C</b> (ibm-eucjp, Cp33722c)  <b>IBM-1046S</b> (ibm1046S, Cp1046S)  <b>IBM-1043</b> (Cp1043, ibm1043)  <b>IBM-1041</b> (Cp1041, ibm1041)  <b>IBM-1027</b> (Cp1027, ibm1027)  <b>CESU-8</b> (CESU8)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>GB2312</b> (gb2312-1980, gb2312-80)  <b>GBK</b> (GBK)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>IBM-1114</b> (Cp1114, ibm1114)  <b>IBM-1115</b> (Cp1115, ibm1115)  <b>IBM-1140</b> (ibm1140, Cp1140)  <b>IBM-1141</b> (Cp1141, ibm1141)  <b>IBM-1142</b> (Cp1142, ibm1142)  <b>IBM-1143</b> (Cp1143, ibm1143)  <b>IBM-1144</b> (ibm1144, Cp1144)  <b>IBM-1145</b> (Cp1145, ibm1145)  <b>IBM-1146</b> (Cp1146, ibm1146)  <b>IBM-1147</b> (Cp1147, ibm1147)  <b>IBM-1148</b> (ibm1148, Cp1148)  <b>IBM-1149</b> (Cp1149, ibm1149)  <b>IBM-1351</b> (Cp1351, ibm1351)  <b>IBM-1362</b> (Cp1362, ibm1362)  <b>IBM-1363</b> (ibm1363, Cp1363)                 </p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<b>Windows</b> Windows	<p> <b>windows-1256S</b> (Cp1256s, ibm-1256s)  <b>UTF-8J</b> (UTF8J)  <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>PTCP154</b> (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)  <b>KOI8-RU</b> (ibm-1168, koi8_ru)  <b>ISO-8859-16</b> (8859-16, iso8859_16, iso8859-16)  <b>ISO-8859-14</b> (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)  <b>IBM01141</b> (cp1141, ccsid01141, cp01141, 1141)  <b>IBM01142</b> (cp01142, cp1142, 1142, ccsid01142)  <b>IBM01143</b> (cp01143, 1143, ccsid01143, cp1143)  <b>IBM01144</b> (cp01144, cp1144, ccsid01144, 1144)  <b>IBM01145</b> (cp1145, cp01145, ccsid01145, 1145)  <b>IBM01146</b> (ccsid01146, cp01146, cp1146, 1146)  <b>IBM01147</b> (ccsid01147, cp1147, 1147, cp01147)  <b>IBM01148</b> (cp1148, ccsid01148, 1148, cp01148)  <b>IBM01149</b> (cp1149, cp01149, ccsid01149, 1149)  <b>IBM1047</b> (cp1047, 1047, ibm-1047)  <b>ISO-2022-JP-2</b> (csISO2022JP2, iso2022jp2)  <b>x-Big5-Solaris</b> (Big5_Solaris)  <b>x-eucJP-Open</b> (EUC_JP_Solaris, eucJP-open)  <b>x-IBM33722</b> (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722)  <b>x-IBM930</b> (cp930, ibm930, ibm-930, 930)  <b>x-IBM939</b> (ibm-939, ibm939, cp939, 939)  <b>x-IBM964</b> (964, cp964, ibm-964, ibm964)  <b>x-ISO-2022-CN-CNS</b> (ISO-2022-CN-CNS, ISO2022CN_CNS)  <b>x-iso-8859-11</b> (iso-8859-11, iso8859_11)  <b>x-JISAutoDetect</b> (JISAutoDetect)  <b>x-MS932_0213</b> ()  <b>x-MS950-HKSCS</b> (MS950_HKSCS)  <b>x-PCK</b> (pck)  <b>x-IBM1363C</b> (ibm1363c, cp1363c, ibm-1363c)  <b>x-IBM420S</b> (420s, ibm-420s, csibm420s, ibm420s, cp420s)  <b>x-IBM864S</b> (csibm864s, ibm864s, cp864s, 864s, ibm-864s)  <b>x-IBM943C</b> (cp943c, 943c, ibm-943c, ibm943c)  <b>x-IBM949C</b> (ibm949c, cp949c, 949c, ibm-949c)  <b>x-IBM954C</b> (cp954c, 954c, ibm-954c, ibm954c)  <b>x-ISO-8859-6S</b> (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s)  <b>x-JIS0208</b> (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208)  <b>x-KSC5601</b> (ksc5601)  <b>x-MacDingbat</b> (macdingbat)  <b>x-MacHebrew</b> (machebrew)  <b>x-MacThai</b> (macthai)  <b>x-MacUkraine</b> (macukraine)  <b>x-IBM1046S</b> (ibm-1046s, 1046s, cp1046s, ibm1046s)  <b>x-IBM-udcJP</b> (IBM-udcJP)  <b>JIS_X0201</b> (jis_x0201, x0201, cshalfwidthkatakana, jis0201)  <b>IBM-939A</b> (Cp939A, ibm939A)  <b>IBM-930A</b> (ibm930A, Cp930A)  <b>IBM-33722A</b> (Cp33722A, ibm33722A)  <b>x-windows-iso2022jp</b> (windows-iso2022jp)  <b>x-windows-50221</b> (ms50221, cp50221)  <b>x-windows-50220</b> (cp50220, ms50220)  <b>X-UTF-32LE-BOM</b> (UTF_32LE_BOM, UTF-32LE-BOM)  <b>X-UTF-32BE-BOM</b> (UTF_32BE_BOM, UTF-32BE-BOM)  <b>x-SJIS_0213</b> ()  <b>IBM01140</b> (ccsid01140, cp01140, 1140, cp1140)  <b>IBM00858</b> (cp858, ccsid00858, 858, cp00858)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>IBM-859</b> (Cp859, ibm859)  <b>IBM-837</b> (ibm837, Cp837)         </p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>Windows</b> Windows</p>	<p> <b>IBM-836</b> (ibm836, Cp836)  <b>IBM-835</b> (ibm835, Cp835)  <b>IBM-833</b> (ibm833, Cp833)  <b>IBM-808</b> (Cp808, ibm808)  <b>IBM-720</b> (Cp720, ibm720)  <b>IBM-33722C</b> (ibm-eucjp, Cp33722c)  <b>IBM-301</b> (Cp301, ibm301)  <b>IBM-300</b> (Cp300, ibm300)  <b>IBM-290</b> (ibm290, Cp290)  <b>IBM-1399</b> (ibm1399, Cp1399)  <b>IBM-1390</b> (Cp1390, ibm1390)  <b>IBM-1388</b> (Cp1388, ibm1388)  <b>IBM-1385</b> (Cp1385, ibm1385)  <b>IBM-1382</b> (ibm1382, Cp1382)  <b>IBM-1088</b> (Cp1088, ibm1088)  <b>IBM-1043</b> (Cp1043, ibm1043)  <b>IBM-1041</b> (Cp1041, ibm1041)  <b>IBM-1027</b> (Cp1027, ibm1027)  <b>CESU-8</b> (CESU8)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>GB2312</b> (gb2312-1980, gb2312-80)  <b>GBK</b> (GBK)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>IBM-1115</b> (Cp1115, ibm1115)  <b>IBM-1351</b> (Cp1351, ibm1351)  <b>IBM-1362</b> (Cp1362, ibm1362)  <b>IBM-1363</b> (ibm1363, Cp1363)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-1370</b> (Cp1370, ibm1370)  <b>IBM-1371</b> (Cp1371, ibm1371)  <b>IBM-1380</b> (Cp1380, ibm1380)  <b>IBM-867</b> (Cp867, ibm867)  <b>IBM-897</b> (Cp897, ibm897)  <b>IBM-924</b> (Cp924, ibm924)  <b>IBM-927</b> (ibm927, Cp927)  <b>IBM-932</b> (ibm932, Cp932)  <b>IBM-947</b> (Cp947, ibm947)  <b>IBM-951</b> (Cp951, ibm951)  <b>IBM-954</b> (ibm954, Cp954)  <b>IBM-971</b> (Cp971, ibm971)  <b>ISO-8859-10</b> (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)                 </p>



Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>Linux</b> Red Hat Enterprise Linux on x86-64</p>	<p><b>windows-1256S</b> (Cp1256s, ibm-1256s)  <b>UTF-8J</b> (UTF8J)  <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>UTF-32</b> (UCS-4, UTF32, ISO-10646-UCS-4)  <b>PTCP154</b> (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)  <b>KOI8-RU</b> (ibm-1168, koi8_ru)  <b>ISO-8859-16</b> (8859-16, iso8859_16, iso8859-16)  <b>ISO-8859-14</b> (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)  <b>IBM01141</b> (cp1141, ccsid01141, cp01141, 1141)  <b>IBM01142</b> (cp01142, cp1142, 1142, ccsid01142)  <b>IBM01143</b> (cp01143, 1143, ccsid01143, cp1143)  <b>IBM01144</b> (cp01144, cp1144, ccsid01144, 1144)  <b>IBM01145</b> (cp1145, cp01145, ccsid01145, 1145)  <b>IBM01146</b> (ccsid01146, cp01146, cp1146, 1146)  <b>IBM01147</b> (ccsid01147, cp1147, 1147, cp01147)  <b>IBM01148</b> (cp1148, ccsid01148, 1148, cp01148)  <b>IBM01149</b> (cp1149, cp01149, ccsid01149, 1149)  <b>IBM1047</b> (cp1047, 1047, ibm-1047)  <b>IBM918</b> (cp918, ebcdic-cp-ar2, ibm-918, 918)  <b>ISO-2022-JP-2</b> (csISO2022JP2, iso2022jp2)  <b>x-Big5-Solaris</b> (Big5_Solaris)  <b>x-eucJP-Open</b> (EUC_JP_Solaris, eucJP-open)  <b>x-IBM33722</b> (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722)  <b>x-IBM930</b> (cp930, ibm930, ibm-930, 930)  <b>x-IBM939</b> (ibm-939, ibm939, cp939, 939)  <b>x-IBM964</b> (964, cp964, ibm-964, ibm964)  <b>x-ISO-2022-CN-CNS</b> (ISO-2022-CN-CNS, ISO2022CN_CNS)  <b>x-iso-8859-11</b> (iso-8859-11, iso8859_11)  <b>x-JISAutoDetect</b> (JISAutoDetect)  <b>x-MS932_0213</b> ()  <b>x-MS950-HKSCS</b> (MS950_HKSCS)  <b>x-PCK</b> (pck)  <b>x-IBM1363C</b> (ibm1363c, cp1363c, ibm-1363c)  <b>x-IBM420S</b> (420s, ibm-420s, csibm420s, ibm420s, cp420s)  <b>x-IBM864S</b> (csibm864s, ibm864s, cp864s, 864s, ibm-864s)  <b>x-IBM943C</b> (cp943c, 943c, ibm-943c, ibm943c)  <b>x-IBM949C</b> (ibm949c, cp949c, 949c, ibm-949c)  <b>x-IBM954C</b> (cp954c, 954c, ibm-954c, ibm954c)  <b>x-ISO-8859-6S</b> (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s)  <b>x-JIS0208</b> (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208)  <b>x-KSC5601</b> (ksc5601)  <b>x-MacDingbat</b> (macdingbat)  <b>x-MacHebrew</b> (machebrew)  <b>x-MacThai</b> (macthai)  <b>x-MacUkraine</b> (macukraine)  <b>x-IBM1046S</b> (ibm-1046s, 1046s, cp1046s, ibm1046s)  <b>x-IBM-udcJP</b> (IBM-udcJP)  <b>JIS_X0201</b> (jis_x0201, x0201, cshalfwidthkatakana, jis0201)  <b>IBM-939A</b> (Cp939A, ibm939A)  <b>IBM-930A</b> (ibm930A, Cp930A)  <b>IBM-33722A</b> (Cp33722A, ibm33722A)  <b>x-windows-iso2022jp</b> (windows-iso2022jp)  <b>x-windows-50221</b> (ms50221, cp50221)  <b>x-windows-50220</b> (cp50220, ms50220)  <b>X-UTF-32LE-BOM</b> (UTF_32LE_BOM, UTF-32LE-BOM)  <b>X-UTF-32BE-BOM</b> (UTF_32BE_BOM, UTF-32BE-BOM)  <b>x-SJIS_0213</b> ()  <b>IBM01140</b> (ccsid01140, cp01140, 1140, cp1140)  <b>IBM00858</b> (cp858, ccsid00858, 858, cp00858)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>IBM-859</b> (Cp859, ibm859)</p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>Linux</b> Red Hat Enterprise Linux on x86-64</p>	<p> <b>IBM-837</b> (ibm837, Cp837)  <b>IBM-836</b> (ibm836, Cp836)  <b>IBM-835</b> (ibm835, Cp835)  <b>IBM-833</b> (ibm833, Cp833)  <b>IBM-808</b> (Cp808, ibm808)  <b>IBM-720</b> (Cp720, ibm720)  <b>IBM-33722C</b> (ibm-eucjp, Cp33722c)  <b>IBM-301</b> (Cp301, ibm301)  <b>IBM-300</b> (Cp300, ibm300)  <b>IBM-290</b> (ibm290, Cp290)  <b>IBM-1399</b> (ibm1399, Cp1399)  <b>IBM-1390</b> (Cp1390, ibm1390)  <b>IBM-1388</b> (Cp1388, ibm1388)  <b>IBM-1385</b> (Cp1385, ibm1385)  <b>IBM-1382</b> (ibm1382, Cp1382)  <b>IBM-1088</b> (Cp1088, ibm1088)  <b>IBM-1043</b> (Cp1043, ibm1043)  <b>IBM-1041</b> (Cp1041, ibm1041)  <b>IBM-1027</b> (Cp1027, ibm1027)  <b>CESU-8</b> (CESU8)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>GB2312</b> (gb2312-1980, gb2312-80)  <b>GBK</b> (GBK)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>IBM-1114</b> (Cp1114, ibm1114)  <b>IBM-1115</b> (Cp1115, ibm1115)  <b>IBM-1351</b> (Cp1351, ibm1351)  <b>IBM-1362</b> (Cp1362, ibm1362)  <b>IBM-1363</b> (ibm1363, Cp1363)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-1370</b> (Cp1370, ibm1370)  <b>IBM-1371</b> (Cp1371, ibm1371)  <b>IBM-1380</b> (Cp1380, ibm1380)  <b>IBM-867</b> (Cp867, ibm867)  <b>IBM-897</b> (Cp897, ibm897)  <b>IBM-924</b> (Cp924, ibm924)  <b>IBM-927</b> (ibm927, Cp927)  <b>IBM-932</b> (ibm932, Cp932)  <b>IBM-947</b> (Cp947, ibm947)  <b>IBM-951</b> (Cp951, ibm951)  <b>IBM-954</b> (ibm954, Cp954)  <b>IBM-971</b> (Cp971, ibm971)  <b>ISO-8859-10</b> (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)                 </p>

Table 375. Platform-specific encodings by platform (continued)


Platform	Supported encodings (not in common encodings table)
<p> z/OS</p>	<p><b>windows-1256S</b> (Cp1256s, ibm-1256s)  <b>UTF-8J</b> (UTF8J)  <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>UTF-32</b> (UCS-4, UTF32, ISO-10646-UCS-4)  <b>UTF-16</b> (UTF16, Unicode, UTF_16, UCS-2)  <b>Shift_JIS</b> ()  <b>PTCP154</b> (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)  <b>MacUkraine</b> ()  <b>MacThai</b> ()  <b>MacHebrew</b> ()  <b>MacDingbat</b> ()  <b>KSC5601</b> ()  <b>KOI8-U</b> (koi8_u, ibm-1167)  <b>KOI8-RU</b> (ibm-1168, koi8_ru)  <b>Johab</b> (x-johab)  <b>JISO208</b> ()  <b>JISO201</b> ()  <b>ISO-8859-6S</b> (iso8859-6S, iso8859_6S)  <b>ISO-8859-16</b> (8859-16, iso8859_16, iso8859-16)  <b>ISO-8859-14</b> (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)  <b>ISO-8859-10</b> (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)  <b>IBM-971</b> (Cp971, ibm971)  <b>IBM-964</b> (ibm-euctw, Cp964)  <b>IBM-954C</b> (Cp954c)  <b>IBM-954</b> (ibm954, Cp954)  <b>IBM-951</b> (Cp951, ibm951)  <b>IBM-949C</b> (Cp949C, ibm949C)  <b>IBM-947</b> (Cp947, ibm947)  <b>IBM-943C</b> (ibm943C, Cp943C)  <b>IBM-942C</b> (Cp942C, ibm942C)  <b>IBM-939</b> (Cp5035, 5035)  <b>IBM-932</b> (ibm932, Cp932)  <b>IBM-930</b> (Cp5026, 5026)  <b>IBM-927</b> (ibm927, Cp927)  <b>IBM-924</b> (Cp924, ibm924)  <b>IBM-918</b> (ibm918, Cp918)  <b>IBM-897</b> (Cp897, ibm897)  <b>IBM-867</b> (Cp867, ibm867)  <b>IBM-1380</b> (Cp1380, ibm1380)  <b>IBM-1371</b> (Cp1371, ibm1371)  <b>IBM-1370</b> (Cp1370, ibm1370)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-1363C</b> (ibm1363C, Cp1363C)  <b>IBM-1363</b> (ibm1363, Cp1363)  <b>IBM-1088</b> (Cp1088, ibm1088)  <b>IBM-1382</b> (ibm1382, Cp1382)  <b>IBM-1385</b> (Cp1385, ibm1385)  <b>IBM-1386</b> (ibm1386, Cp1386)  <b>IBM-1388</b> (Cp1388, ibm1388)  <b>IBM-1390</b> (Cp1390, ibm1390)  <b>IBM-1399</b> (ibm1399, Cp1399)  <b>IBM-290</b> (ibm290, Cp290)  <b>IBM-300</b> (Cp300, ibm300)  <b>IBM-301</b> (Cp301, ibm301)  <b>IBM-33722</b> (5050, Cp5050)  <b>IBM-33722C</b> (ibm-eucjp, Cp33722c)  <b>IBM-930A</b> (ibm930A, Cp930A)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>IBM-864S</b> (ibm864S, Cp864S)  <b>IBM-859</b> (Cp859, ibm859)  <b>IBM-858</b> (Cp858, ibm858)</p>

Table 375. Platform-specific encodings by platform (continued)


Platform	Supported encodings (not in common encodings table)
 z/OS	<p> <b>IBM-837</b> (ibm837, Cp837)  <b>IBM-836</b> (ibm836, Cp836)  <b>IBM-835</b> (ibm835, Cp835)  <b>IBM-833</b> (ibm833, Cp833)  <b>IBM-808</b> (Cp808, ibm808)  <b>IBM-720</b> (Cp720, ibm720)  <b>IBM-420S</b> (Cp420S, ibm420S)  <b>IBM-1047</b> (Cp1047, ibm1047)  <b>IBM-1046S</b> (ibm1046S, Cp1046S)  <b>IBM-1043</b> (Cp1043, ibm1043)  <b>IBM-1041</b> (Cp1041, ibm1041)  <b>IBM-1027</b> (Cp1027, ibm1027)  <b>CESU-8</b> (CESU8)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>GB2312</b> (gb2312-1980, gb2312-80)  <b>GBK</b> (GBK)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>IBM-1114</b> (Cp1114, ibm1114)  <b>IBM-1115</b> (Cp1115, ibm1115)  <b>IBM-1140</b> (ibm1140, Cp1140)  <b>IBM-1141</b> (Cp1141, ibm1141)  <b>IBM-1142</b> (Cp1142, ibm1142)  <b>IBM-1143</b> (Cp1143, ibm1143)  <b>IBM-1144</b> (ibm1144, Cp1144)  <b>IBM-1145</b> (Cp1145, ibm1145)  <b>IBM-1146</b> (Cp1146, ibm1146)  <b>IBM-1147</b> (Cp1147, ibm1147)  <b>IBM-1148</b> (ibm1148, Cp1148)  <b>IBM-1149</b> (Cp1149, ibm1149)  <b>IBM-1351</b> (Cp1351, ibm1351)  <b>IBM-1362</b> (Cp1362, ibm1362)                 </p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>Linux</b> Linux on POWER Systems - Big Endian</p>	<p><b>windows-1256S</b> (Cp1256s, ibm-1256s)  <b>UTF-8J</b> (UTF8J)  <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>UTF-32</b> (UCS-4, UTF32, ISO-10646-UCS-4)  <b>UTF-16</b> (UTF16, Unicode, UTF_16, UCS-2)  <b>Shift_JIS</b> ()  <b>PTCP154</b> (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)  <b>MacUkraine</b> ()  <b>MacThai</b> ()  <b>MacHebrew</b> ()  <b>MacDingbat</b> ()  <b>KSC5601</b> ()  <b>KOI8-U</b> (koi8_u, ibm-1167)  <b>KOI8-RU</b> (ibm-1168, koi8_ru)  <b>Johab</b> (x-johab)  <b>JISO208</b> ()  <b>JISO201</b> ()  <b>ISO-8859-6S</b> (iso8859-6S, iso8859_6S)  <b>ISO-8859-16</b> (8859-16, iso8859_16, iso8859-16)  <b>ISO-8859-14</b> (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)  <b>ISO-8859-10</b> (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)  <b>IBM-971</b> (Cp971, ibm971)  <b>IBM-964</b> (ibm-euctw, Cp964)  <b>IBM-954C</b> (Cp954c)  <b>IBM-954</b> (ibm954, Cp954)  <b>IBM-951</b> (Cp951, ibm951)  <b>IBM-949C</b> (Cp949C, ibm949C)  <b>IBM-947</b> (Cp947, ibm947)  <b>IBM-943C</b> (ibm943C, Cp943C)  <b>IBM-942C</b> (Cp942C, ibm942C)  <b>IBM-939</b> (Cp5035, 5035)  <b>IBM-932</b> (ibm932, Cp932)  <b>IBM-930</b> (Cp5026, 5026)  <b>IBM-927</b> (ibm927, Cp927)  <b>IBM-924</b> (Cp924, ibm924)  <b>IBM-918</b> (ibm918, Cp918)  <b>IBM-897</b> (Cp897, ibm897)  <b>IBM-867</b> (Cp867, ibm867)  <b>IBM-1380</b> (Cp1380, ibm1380)  <b>IBM-1371</b> (Cp1371, ibm1371)  <b>IBM-1370</b> (Cp1370, ibm1370)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-1363C</b> (ibm1363C, Cp1363C)  <b>IBM-1047</b> (Cp1047, ibm1047)  <b>IBM-1088</b> (Cp1088, ibm1088)  <b>IBM-1382</b> (ibm1382, Cp1382)  <b>IBM-1385</b> (Cp1385, ibm1385)  <b>IBM-1386</b> (ibm1386, Cp1386)  <b>IBM-1388</b> (Cp1388, ibm1388)  <b>IBM-1390</b> (Cp1390, ibm1390)  <b>IBM-1399</b> (ibm1399, Cp1399)  <b>IBM-290</b> (ibm290, Cp290)  <b>IBM-300</b> (Cp300, ibm300)  <b>IBM-301</b> (Cp301, ibm301)  <b>IBM-33722</b> (5050, Cp5050)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>IBM-864S</b> (ibm864S, Cp864S)  <b>IBM-859</b> (Cp859, ibm859)  <b>IBM-858</b> (Cp858, ibm858)</p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
<p><b>Linux</b> Linux on POWER Systems - Big Endian</p>	<p> <b>IBM-837</b> (ibm837, Cp837)  <b>IBM-836</b> (ibm836, Cp836)  <b>IBM-835</b> (ibm835, Cp835)  <b>IBM-833</b> (ibm833, Cp833)  <b>IBM-808</b> (Cp808, ibm808)  <b>IBM-720</b> (Cp720, ibm720)  <b>IBM-420S</b> (Cp420S, ibm420S)  <b>IBM-33722C</b> (ibm-eucjp, Cp33722c)  <b>IBM-1046S</b> (ibm1046S, Cp1046S)  <b>IBM-1043</b> (Cp1043, ibm1043)  <b>IBM-1041</b> (Cp1041, ibm1041)  <b>IBM-1027</b> (Cp1027, ibm1027)  <b>CESU-8</b> (CESU8)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>GB2312</b> (gb2312-1980, gb2312-80)  <b>GBK</b> (GBK)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>IBM-1114</b> (Cp1114, ibm1114)  <b>IBM-1115</b> (Cp1115, ibm1115)  <b>IBM-1140</b> (ibm1140, Cp1140)  <b>IBM-1141</b> (Cp1141, ibm1141)  <b>IBM-1142</b> (Cp1142, ibm1142)  <b>IBM-1143</b> (Cp1143, ibm1143)  <b>IBM-1144</b> (ibm1144, Cp1144)  <b>IBM-1145</b> (Cp1145, ibm1145)  <b>IBM-1146</b> (Cp1146, ibm1146)  <b>IBM-1147</b> (Cp1147, ibm1147)  <b>IBM-1148</b> (ibm1148, Cp1148)  <b>IBM-1149</b> (Cp1149, ibm1149)  <b>IBM-1351</b> (Cp1351, ibm1351)  <b>IBM-1362</b> (Cp1362, ibm1362)  <b>IBM-1363</b> (ibm1363, Cp1363)                 </p>

Table 375. Platform-specific encodings by platform (continued)

Platform	Supported encodings (not in common encodings table)
HP (PA-RISC)	<p> <b>UTF-32LE</b> (UTF_32LE, X-UTF-32LE, UTF32LE)  <b>UTF-32BE</b> (UTF_32BE, X-UTF-32BE, UTF32BE)  <b>IBM01147</b> (ccsid01147, cp1147, 1147, cp01147)  <b>IBM01148</b> (cp1148, ccsid01148, 1148, cp01148)  <b>IBM01149</b> (cp1149, cp01149, ccsid01149, 1149)  <b>IBM1047</b> (cp1047, 1047, ibm-1047)  <b>IBM918</b> (cp918, ebcdic-cp-ar2, ibm-918, 918)  <b>ISO-2022-JP-2</b> (csISO2022JP2, iso2022jp2)  <b>Roman9</b> (Roman9)  <b>x-Big5-Solaris</b> (Big5_Solaris)  <b>x-eucJP-Open</b> (EUC_JP_Solaris, eucJP-open)  <b>x-IBM33722</b> (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722)  <b>x-IBM930</b> (cp930, ibm930, ibm-930, 930)  <b>x-IBM939</b> (ibm-939, ibm939, cp939, 939)  <b>x-windows-iso2022jp</b> (windows-iso2022jp)  <b>x-windows-50221</b> (ms50221, cp50221)  <b>x-windows-50220</b> (cp50220, ms50220)  <b>X-UTF-32LE-BOM</b> (UTF_32LE_BOM, UTF-32LE-BOM)  <b>X-UTF-32BE-BOM</b> (UTF_32BE_BOM, UTF-32BE-BOM)  <b>x-SJIS_0213</b> ()  <b>x-PCK</b> (pck)  <b>x-MS950-HKSCS</b> (MS950_HKSCS)  <b>x-MS932_0213</b> ()  <b>x-JISAutoDetect</b> (JISAutoDetect)  <b>x-iso-8859-11</b> (iso-8859-11, iso8859_11)  <b>x-ISO-2022-CN-CNS</b> (ISO-2022-CN-CNS, ISO2022CN_CNS)  <b>x-IBM964</b> (964, cp964, ibm-964, ibm964)  <b>IBM01146</b> (ccsid01146, cp01146, cp1146, 1146)  <b>IBM01145</b> (cp1145, cp01145, ccsid01145, 1145)  <b>IBM01144</b> (cp01144, cp1144, ccsid01144, 1144)  <b>IBM01143</b> (cp01143, 1143, ccsid01143, cp1143)  <b>IBM01142</b> (cp01142, cp1142, 1142, ccsid01142)  <b>IBM01141</b> (cp1141, ccsid01141, cp01141, 1141)  <b>IBM01140</b> (ccsid01140, cp01140, 1140, cp1140)  <b>IBM00858</b> (cp858, ccsid00858, 858, cp00858)  <b>X-UnicodeLittle</b> (UnicodeLittle)  <b>X-UnicodeBig</b> (UnicodeBig)  <b>COMPOUND_TEXT</b> (x-compound-text, x11-compound-text)  <b>hp-roman8</b> (roman8, ibm-1051, r8, Cp1051)  <b>IBM-1364</b> (Cp1364, ibm1364)  <b>IBM-942C</b> (Cp942C, ibm942C)  <b>IBM-943C</b> (ibm943C, Cp943C)  <b>IBM-949C</b> (Cp949C, ibm949C)  <b>JISO201</b> ()  <b>JISO208</b> ()  <b>KOI8-U</b> (koi8_u, ibm-1167)  <b>MacDingbat</b> ()  <b>MacHebrew</b> ()  <b>MacThai</b> ()  <b>MacUkraine</b> ()  <b>UTF-32</b> (UCS-4, UTF32, ISO-10646-UCS-4)                 </p>

**Platforms by encoding**

Table 376. Platform-specific encodings by encoding












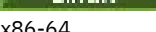



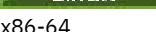



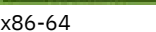
Encoding	Aliases	Platforms on which this encoding is supported
<b>x-MacUkraine</b>	macukraine	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-MacThai</b>	macthai	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-MacHebrew</b>	machebrew	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-MacDingbat</b>	macingbat	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-KSC5601</b>	ksc5601	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>



Table 376. Platform-specific encodings by encoding (continued)






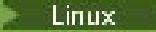


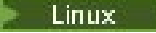










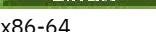
Encoding	Aliases	Platforms on which this encoding is supported
<b>x-JIS0208</b>	jis_c6226-1983, jis_x0208-1983, csiso87jjsx0208, x0208, iso-ir-87, jis0208	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-ISO-8859-6S</b>	8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-IBM954C</b>	cp954c, 954c, ibm-954c, ibm954c	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-IBM949C</b>	ibm949c, cp949c, 949c, ibm-949c	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-IBM943C</b>	cp943c, 943c, ibm-943c, ibm943c	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>

Table 376. Platform-specific encodings by encoding (continued)






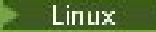


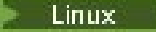









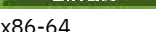
Encoding	Aliases	Platforms on which this encoding is supported
<b>x-IBM864S</b>	csibm864s, ibm864s, cp864s, 864s, ibm-864s	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-IBM420S</b>	420s, ibm-420s, csibm420s, ibm420s, cp420s	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-IBM1363C</b>	ibm1363c, cp1363c, ibm-1363c	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-IBM1046S</b>	ibm-1046s, 1046s, cp1046s, ibm1046s	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>
<b>x-IBM-udcJP</b>	IBM-udcJP	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64</p>

Table 376. Platform-specific encodings by encoding (continued)


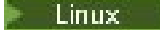














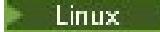






Encoding	Aliases	Platforms on which this encoding is supported
<b>JIS_X0201</b>	jis_x0201, x0201, cshalfwidthkatakana, jis0201	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> Linux Linux for IBM Z</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> </ul>
<b>IBM-939A</b>	Cp939A, ibm939A	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> </ul>
<b>IBM-930A</b>	ibm930A, Cp930A	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> </ul>
<b>IBM-924_LF</b>	Cp924_LF, ibm924_LF	<ul style="list-style-type: none"> <li> IBM i IBM i</li> </ul>
<b>IBM-33722A</b>	Cp33722A, ibm33722A	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> </ul>
<b>IBM-1141_LF</b>	Cp1141_LF, ibm1141_LF	<ul style="list-style-type: none"> <li> IBM i IBM i</li> </ul>
<b>IBM-1047_LF</b>	Cp1047_LF, ibm1047_LF	<ul style="list-style-type: none"> <li> IBM i IBM i</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)





















Encoding	Aliases	Platforms on which this encoding is supported
<b>x-windows-iso2022jp</b>	windows-iso2022jp	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-windows-50221</b>	ms50221, cp50221	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-windows-50220</b>	cp50220, ms50220	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>X-UTF-32LE-BOM</b>	UTF_32LE_BOM, UTF-32LE-BOM	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>X-UTF-32BE-BOM</b>	UTF_32BE_BOM, UTF-32BE-BOM	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>

Table 376. Platform-specific encodings by encoding (continued)


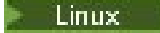


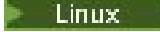




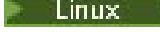









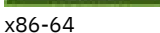
Encoding	Aliases	Platforms on which this encoding is supported
<b>x-SJIS_0213</b>		<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-PCK</b>	pck	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-MS950-HKSCS</b>	MS950_HKSCS	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-MS932_0213</b>		<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-JISAutoDetect</b>	JISAutoDetect	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>

Table 376. Platform-specific encodings by encoding (continued)
















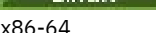



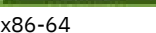
Encoding	Aliases	Platforms on which this encoding is supported
<b>x-iso-8859-11</b>	iso-8859-11, iso8859_11	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-ISO-2022-CN-CNS</b>	ISO-2022-CN-CNS, ISO2022CN_CNS	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-IBM964</b>	964, cp964, ibm-964, ibm964	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-IBM939</b>	ibm-939, ibm939, cp939, 939	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-IBM930</b>	cp930, ibm930, ibm-930, 930	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>

Table 376. Platform-specific encodings by encoding (continued)





















Encoding	Aliases	Platforms on which this encoding is supported
<b>x-IBM33722</b>	ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-eucJP-Open</b>	EUC_JP_Solaris, eucJP-open	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>x-Big5-Solaris</b>	Big5_Solaris	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>Roman9</b>	Roman9	HP (PA-RISC)
<b>ISO-2022-JP-2</b>	csISO2022JP2, iso2022jp2	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM918</b>	cp918, ebcdic-cp-ar2, ibm-918, 918	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>

Table 376. Platform-specific encodings by encoding (continued)





















Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM1047</b>	cp1047, 1047, ibm-1047	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM01149</b>	cp1149, cp01149, ccsid01149, 1149	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM01148</b>	cp1148, ccsid01148, 1148, cp01148	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM01147</b>	ccsid01147, cp1147, 1147, cp01147	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM01146</b>	ccsid01146, cp01146, cp1146, 1146	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>



Table 376. Platform-specific encodings by encoding (continued)
















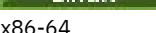



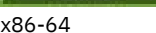
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM01145</b>	cp1145, cp01145, ccsid01145, 1145	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM01144</b>	cp01144, cp1144, ccsid01144, 1144	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM01143</b>	cp01143, 1143, ccsid01143, cp1143	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM01142</b>	cp01142, cp1142, 1142, ccsid01142	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>
<b>IBM01141</b>	cp1141, ccsid01141, cp01141, 1141	<p> SUSE Linux Enterprise Server on x86-64</p> <p> Linux for IBM Z</p> <p> Windows</p> <p> Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</p>

Table 376. Platform-specific encodings by encoding (continued)

















Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM01140</b>	ccsid01140, cp01140, 1140, cp1140	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> Linux Linux for IBM Z</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</li> </ul>
<b>IBM00858</b>	cp858, ccsid00858, 858, cp00858	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> Linux Linux for IBM Z</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64 HP (PA-RISC)</li> </ul>
<b>X-UnicodeLittle</b>	UnicodeLittle	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian HP (PA-RISC)</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)













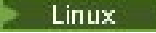











Encoding	Aliases	Platforms on which this encoding is supported
<b>X-UnicodeBig</b>	UnicodeBig	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian HP (PA-RISC)</li> </ul>
<b>IBM-864S</b>	ibm864S, Cp864S	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-859</b>	Cp859, ibm859	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-858</b>	Cp858, ibm858	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)



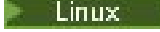
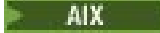












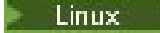

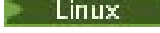
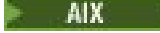



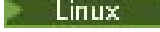
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-837</b>	ibm837, Cp837	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-836</b>	ibm836, Cp836	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-835</b>	ibm835, Cp835	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)



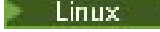
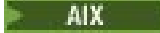












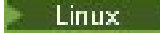

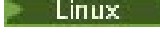
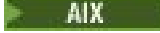



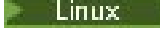
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-833</b>	ibm833, Cp833	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-808</b>	Cp808, ibm808	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-720</b>	Cp720, ibm720	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)

























Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-420S</b>	Cp420S, ibm420S	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-33722C</b>	ibm-eucjp, Cp33722c	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-33722</b>	5050, Cp5050	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-301</b>	Cp301, ibm301	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)














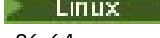










Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-300</b>	Cp300, ibm300	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-290</b>	ibm290, Cp290	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1399</b>	ibm1399, Cp1399	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)



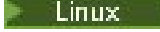
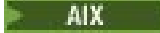




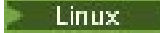










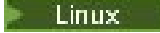
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1390</b>	Cp1390, ibm1390	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1388</b>	Cp1388, ibm1388	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1386</b>	ibm1386, Cp1386	<ul style="list-style-type: none"> <li> IBM i IBM i</li> <li> AIX AIX</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>



Table 376. Platform-specific encodings by encoding (continued)



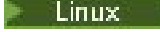
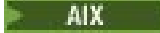












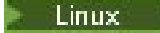

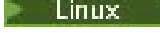
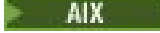



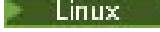
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1385</b>	Cp1385, ibm1385	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1382</b>	ibm1382, Cp1382	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1088</b>	Cp1088, ibm1088	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)

























Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1047</b>	Cp1047, ibm1047	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1046S</b>	ibm1046S, Cp1046S	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1043</b>	Cp1043, ibm1043	 SUSE Linux Enterprise Server on x86-64  IBM i  Linux for IBM Z  AIX  Windows  Red Hat Enterprise Linux on x86-64  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1041</b>	Cp1041, ibm1041	 SUSE Linux Enterprise Server on x86-64  IBM i  Linux for IBM Z  AIX  Windows  Red Hat Enterprise Linux on x86-64  z/OS  Linux on POWER Systems - Big Endian

Table 376. Platform-specific encodings by encoding (continued)

























Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1027</b>	Cp1027, ibm1027	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>CESU-8</b>	CESU8	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>COMPOUND_TEXT</b>	x-compound-text, x11-compound-text	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> <li>HP (PA-RISC)</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)

























Encoding	Aliases	Platforms on which this encoding is supported
<b>GB2312</b>	gb2312-1980, gb2312-80	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>GBK</b>	GBK	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>hp-roman8</b>	roman8, ibm-1051, r8, Cp1051	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> <li>HP (PA-RISC)</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)



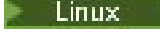
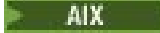




















Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1114</b>	Cp1114, ibm1114	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1115</b>	Cp1115, ibm1115	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1140</b>	ibm1140, Cp1140	<ul style="list-style-type: none"> <li> IBM i IBM i</li> <li> AIX AIX</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1141</b>	Cp1141, ibm1141	<ul style="list-style-type: none"> <li> IBM i IBM i</li> <li> AIX AIX</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)

























Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1142</b>	Cp1142, ibm1142	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1143</b>	Cp1143, ibm1143	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1144</b>	ibm1144, Cp1144	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1145</b>	Cp1145, ibm1145	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1146</b>	Cp1146, ibm1146	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1147</b>	Cp1147, ibm1147	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian

Table 376. Platform-specific encodings by encoding (continued)








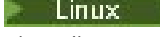
















Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1148</b>	ibm1148, Cp1148	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1149</b>	Cp1149, ibm1149	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1351</b>	Cp1351, ibm1351	 SUSE Linux Enterprise Server on x86-64  IBM i  Linux for IBM Z  AIX  Windows  Red Hat Enterprise Linux on x86-64  z/OS  Linux on POWER Systems - Big Endian
<b>IBM-1362</b>	Cp1362, ibm1362	 SUSE Linux Enterprise Server on x86-64  IBM i  Linux for IBM Z  AIX  Windows  Red Hat Enterprise Linux on x86-64  z/OS  Linux on POWER Systems - Big Endian

Table 376. Platform-specific encodings by encoding (continued)



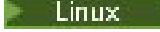
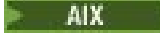








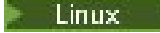






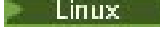
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1363</b>	ibm1363, Cp1363	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1363C</b>	ibm1363C, Cp1363C	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1364</b>	Cp1364, ibm1364	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> <li>HP (PA-RISC)</li> </ul>



Table 376. Platform-specific encodings by encoding (continued)



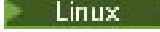
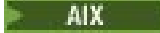












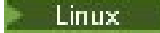

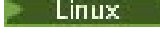
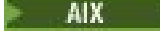



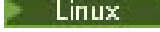
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-1370</b>	Cp1370, ibm1370	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1371</b>	Cp1371, ibm1371	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-1380</b>	Cp1380, ibm1380	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)




















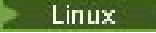
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-867</b>	Cp867, ibm867	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-897</b>	Cp897, ibm897	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-918</b>	ibm918, Cp918	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)



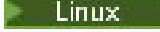
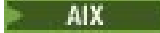




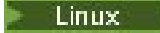










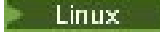
Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-924</b>	Cp924, ibm924	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-927</b>	ibm927, Cp927	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-930</b>	Cp5026, 5026	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)





















Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-932</b>	ibm932, Cp932	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-939</b>	Cp5035, 5035	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-942C</b>	Cp942C, ibm942C	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> <li>HP (PA-RISC)</li> </ul>
<b>IBM-943C</b>	ibm943C, Cp943C	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> <li>HP (PA-RISC)</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)













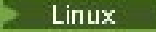

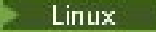





Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-947</b>	Cp947, ibm947	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-949C</b>	Cp949C, ibm949C	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> <li>HP (PA-RISC)</li> </ul>
<b>IBM-951</b>	Cp951, ibm951	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)



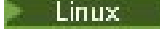
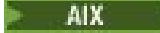




















Encoding	Aliases	Platforms on which this encoding is supported
<b>IBM-954</b>	ibm954, Cp954	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-954C</b>	Cp954c	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-964</b>	ibm-euctw, Cp964	<ul style="list-style-type: none"> <li> IBM i</li> <li> AIX</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>IBM-971</b>	Cp971, ibm971	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)



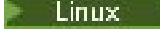
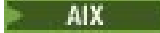












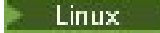

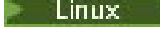
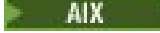



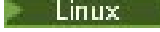
Encoding	Aliases	Platforms on which this encoding is supported
<b>ISO-8859-10</b>	latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>ISO-8859-14</b>	ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>
<b>ISO-8859-16</b>	8859-16, iso8859_16, iso8859-16	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)




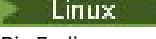



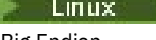







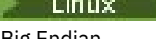








Encoding	Aliases	Platforms on which this encoding is supported
<b>ISO-8859-6S</b>	iso8859-6S, iso8859_6S	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>JIS0201</b>		 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian HP (PA-RISC)
<b>JIS0208</b>		 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian HP (PA-RISC)
<b>Johab</b>	x-johab	 IBM i  AIX  z/OS  Linux on POWER Systems - Big Endian
<b>KOI8-RU</b>	ibm-1168, koi8_ru	 SUSE Linux Enterprise Server on x86-64  IBM i  Linux for IBM Z  AIX  Windows  Red Hat Enterprise Linux on x86-64  z/OS  Linux on POWER Systems - Big Endian



Table 376. Platform-specific encodings by encoding (continued)




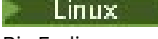



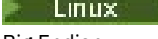







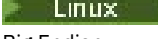



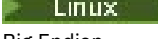
Encoding	Aliases	Platforms on which this encoding is supported
<b>KOI8-U</b>	koi8_u, ibm-1167	 IBM i  AIX  z/OS  Linux Linux on POWER Systems - Big Endian HP (PA-RISC)
<b>KSC5601</b>		 IBM i  AIX  z/OS  Linux Linux on POWER Systems - Big Endian
<b>MacDingbat</b>		 IBM i  AIX  z/OS  Linux Linux on POWER Systems - Big Endian HP (PA-RISC)
<b>MacHebrew</b>		 IBM i  AIX  z/OS  Linux Linux on POWER Systems - Big Endian HP (PA-RISC)
<b>MacThai</b>		 IBM i  AIX  z/OS  Linux Linux on POWER Systems - Big Endian HP (PA-RISC)

Table 376. Platform-specific encodings by encoding (continued)






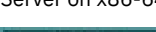














Encoding	Aliases	Platforms on which this encoding is supported
<b>MacUkraine</b>		 IBM i  AIX  z/OS  Linux Linux on POWER Systems - Big Endian HP (PA-RISC)
<b>PTCP154</b>	PT154, IBM-1169, Cyrillic-Asian, csPTCP154	 Linux SUSE Linux Enterprise Server on x86-64  IBM i  Linux Linux for IBM Z  AIX  Windows Windows  Linux Red Hat Enterprise Linux on x86-64  z/OS z/OS  Linux Linux on POWER Systems - Big Endian
<b>Shift_JIS</b>		 IBM i  AIX  z/OS  Linux Linux on POWER Systems - Big Endian
<b>UTF-16</b>	UTF16, Unicode, UTF_16, UCS-2	 IBM i  AIX  z/OS  Linux Linux on POWER Systems - Big Endian

Table 376. Platform-specific encodings by encoding (continued)











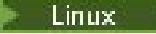




























Encoding	Aliases	Platforms on which this encoding is supported
<b>UTF-32</b>	UCS-4, UTF32, ISO-10646-UCS-4	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian HP (PA-RISC)</li> </ul>
<b>UTF-32BE</b>	UTF_32BE, X-UTF-32BE, UTF32BE	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li> Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian HP (PA-RISC)</li> </ul>
<b>UTF-32LE</b>	UTF_32LE, X-UTF-32LE, UTF32LE	<ul style="list-style-type: none"> <li> SUSE Linux Enterprise Server on x86-64</li> <li> IBM i</li> <li>Linux for IBM Z</li> <li> AIX</li> <li> Windows</li> <li> Red Hat Enterprise Linux on x86-64</li> <li> z/OS</li> <li> Linux on POWER Systems - Big Endian HP (PA-RISC)</li> </ul>

Table 376. Platform-specific encodings by encoding (continued)

Encoding	Aliases	Platforms on which this encoding is supported
UTF-8J	UTF8J	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>
windows-1256S	Cp1256s, ibm-1256s	<ul style="list-style-type: none"> <li> Linux SUSE Linux Enterprise Server on x86-64</li> <li> IBM i IBM i</li> <li> Linux Linux for IBM Z</li> <li> AIX AIX</li> <li> Windows Windows</li> <li> Linux Red Hat Enterprise Linux on x86-64</li> <li> z/OS z/OS</li> <li> Linux Linux on POWER Systems - Big Endian</li> </ul>

**Related tasks**

[Using transfer definition files](#)

**Related reference**

[“Transferring text files with MFT” on page 2507](#)

Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of Managed File Transfer.

[“fteCreateTransfer \(start a new file transfer\)” on page 2048](#)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

**How MFT agents use Java heap and native heap memory**

An IBM MQ Managed File Transfer agent runs as a Java process. As such, the agent runs in the virtualized environment of the Java Virtual Machine (JVM).

The JVM itself is a native process, is bounded by the hardware and operating system. The JVM maintains two memory areas:

- The Java heap

This contains the instances of Java objects and is managed by garbage collection processing. The maximum size of the Java heap is allocated during JVM startup using the **-Xmx** JVM option.

- The native heap

The native heap contains resources for the JVM itself; for example, the Just-In-Time Compiler, Classes, and ClassLoaders.

An agent primarily uses the Java heap. When performing managed transfers, the agent uses the Java heap to create Java objects that are required for the transfer. Any file data that is read into buffers by the agent is also stored in Java heap memory.

An agent does not itself contain any code that uses the native heap. However, there is native code in the Java message queuing interface (JMQUI) that the agent uses to communicate with its agent queue manager.

This native code is used when an agent connects to its agent queue manager using the BINDINGS transport. This is a local shared memory connection (sometimes referred to as interprocess communication, or IPC), rather than a TCP/IP connection which is used if an agent connects using the CLIENT transport. When an agent is configured to use the BINDINGS transport, the native heap is used to pass messages and commands between the agent and the agent queue manager.

This means that a heavily loaded agent that is connected to its agent queue manager using the BINDINGS transport makes more extensive use of the native heap, when compared to an equivalent agent which is connected using the CLIENT transport.

One common misconception is that the Java heap for an agent must be equal to (or greater than) the size of the largest file that is to be transferred. This is not correct as file data is read into memory in stages.

As a guide, the maximum amount of Java heap that is used to store file data for each transfer can be roughly calculated as follows:

```
Memory allocated for a transfer = agentCheckpointInterval *
agentFrameSize * agentWindowSize * agentChunkSize
```

## How Java heap and native heap usage affects agents

When a `java.lang.OutOfMemoryError` occurs, you might think it reasonable to increase the amount of Java heap available to the application, using the **-Xmx** Java System Property. For example, the following property setting attempts to allocate a maximum Java heap size of 2GB:

```
-Xmx2048M
```

However, allocating too much Java heap for an application can cause a `java.lang.OutOfMemoryError` to occur, due to native heap exhaustion. This is because, as the Java heap space grows, the native heap must shrink to accommodate it.

For information about how to prevent `java.lang.OutOfMemoryErrors` that are caused by native heap exhaustion, see [What to do if your MFT agent ABENDS with a java.lang.OutOfMemoryError due to native memory exhaustion](#).

## XML message formats used by MFT

Managed File Transfer uses messages in XML format for a number of purposes: to command an agent; to log information about the monitors, schedules, and transfers; and to define information used for configuration. The logical structure of the XML formats used for these purposes described by XML schema.

Each version of Managed File Transfer uses an XML schema to validate messages written in XML. The agent extracts the XML schema version and determines whether the schema is supported.

After you have installed Managed File Transfer, you can find the Managed File Transfer message schema files in the following directory: *MQ\_INSTALLATION\_PATH/mqft/samples/schema*. The following schemas are included:

#### **Schemas for XML messages that can be put on an agent command queue**

- FileTransfer.xsd
- Internal.xsd
- Monitor.xsd
- PingAgent.xsd

For more information about putting XML messages on an agent command queue, see [Controlling MFT by putting messages on the agent command queue](#).

#### **Schemas for XML messages that are published to the SYSTEM.FTE topic**

- MonitorList.xsd
- MonitorLog.xsd
- ScheduleList.xsd
- ScheduleLog.xsd
- TransferLog.xsd
- TransferStatus.xsd

For more information about XML messages that are published to the SYSTEM.FTE topic and the structure of the SYSTEM.FTE topic, see [SYSTEM.FTE topic](#).

#### **Other schemas used by Managed File Transfer**

*fteutils.xsd*. This schema contains common element definitions and is included by some of the other schemas.

- Notification.xsd
- ProtocolBridgeCredentials.xsd
- ProtocolBridgeProperties.xsd
- ConnectDirectCredentials.xsd
- ConnectDirectNodeProperties.xsd
- ConnectDirectProcessDefinitions.xsd
- Reply.xsd
- UserSandboxes.xsd

#### **Related reference**

[“MFT agent status message format” on page 2615](#)

When a Managed File Transfer Agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the SYSTEM.FTE/Agents/*agent name* topic).

[“File transfer request message format” on page 2665](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the *FileTransfer.xsd* schema and have the <request> element as the root element. The *FileTransfer.xsd* schema document is located in the *MQ\_INSTALLATION\_PATH/mqft/samples/schema* directory. The *FileTransfer.xsd* schema imports *fteutils.xsd*, which is in the same directory.

[“File transfer status message format” on page 2627](#)

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its SYSTEM.FTE/Transfers/*agent\_name/transfer ID* topic), which conforms to the *TransferStatus.xsd* XML schema. The *TransferStatus.xsd* file is located in the *MQ\_INSTALLATION\_PATH/mqft/samples/schema* directory of your WMQMFT installation.

[“File transfer log message formats” on page 2631](#)

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. These messages conform to the schema `TransferLog.xsd`, which is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your Managed File Transfer installation.

[“Scheduled file transfer log message formats” on page 2653](#)

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its `SYSTEM.FTE/Log/agent_name/schedule ID` topic). This message conforms to the `ScheduleLog.xsd` XML schema.

[“MFT monitor request message formats” on page 2682](#)

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the **fteCreateMonitor** command or using the IBM MQ Explorer interface.

[“MFT message formats for security” on page 2695](#)

This topic describes the messages published to the Managed File Transfer coordination queue manager relevant to security.

[“Protocol bridge credentials file format” on page 2700](#)

The `ProtocolBridgeCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

[“Protocol bridge properties file format” on page 2704](#)

The `ProtocolBridgeProperties.xml` file in the agent configuration directory defines properties for protocol file servers.

[“Connect:Direct credentials file format” on page 2714](#)

The `ConnectDirectCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

[“Connect:Direct node properties file format” on page 2721](#)

The `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory specifies information about remote Connect:Direct nodes that are involved in a file transfer.

[“Connect:Direct process definitions file format” on page 2718](#)

The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

[“Ping MFT agent request message format” on page 2692](#)

You can ping an agent by issuing an **ftePingAgent** command or by putting an XML message on the agent command queue. The ping agent request XML must conform to the `PingAgent.xsd` schema. After you have installed Managed File Transfer, you can find the `PingAgent.xsd` schema file in the following directory: `MQ_INSTALLATION_PATH/mqft/samples/schema`. The `PingAgent.xsd` schema imports `fteutils.xsd`, which is in the same directory.

[“MFT agent reply message format” on page 2693](#)

When an agent receives an XML message on its agent command queue, if a response is required, the agent will send an XML reply message to the reply queue defined in the original message. The reply XML conforms to the `Reply.xsd` schema. The `Reply.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `Reply.xsd` schema imports `fteutils.xsd`, which is in the same directory.

## MFT agent status message format

When a Managed File Transfer Agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the `SYSTEM.FTE/Agents/agent name` topic).

The following information is included:

- Agent name
- Platform the agent is running on
- Agent description (if provided)

- Agent's queue manager
- Time zone that the agent is running in
- Agent version
- Agent transfer limits
- State of each of the agent's current transfers. These states are listed in [Agent transfer states](#)
- Type of agent

If the agent is a protocol bridge agent the following information is also included:

- Type of protocol bridge agent
- Host name or IP address of the protocol bridge server

The agent status is republished whenever the agent transfer states change, but by default no more than every 30 seconds. You can change this default setting using the `agentStatusPublishRateLimit` agent property, which is described in [Advanced agent properties: General](#).

The following example output shows the keys used for each data element in the agent status:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="agentOsName">Windows 7</entry>
  <entry key="agentDescription"/>
  <entry key="queueManager">QM1</entry>
  <entry key="agentTimeZone">Europe/London</entry>
  <entry key="agentVersion">1.00</entry>
  <entry key="agentName">FTEAGENT</entry>
  <entry key="maxDestinationTransfers">25</entry>
  <entry key="maxSourceTransfers">25</entry>
  <entry key="maxQueuedTransfers">100</entry>
  <entry
key="DestinationTransferStates">414d51204d554e474f202020202020d857374a60a72622=RunningTransfer
414d51204d554e474f202020202020d857374a69a72622=RunningTransfer
414d51204d554e474f202020202020d857374a75a72622=RunningTransfer
  </entry>
  <entry
key="SourceTransferStates">414d51204d554e474f202020202020d857374a93a72622=NegotiatingTransfer
414d51204d554e474f202020202020d857374a78a72622=RunningTransfer
414d51204d554e474f202020202020d857374aaba72622=NewSenderTransfer
414d51204d554e474f202020202020d857374a63a72622=RunningTransfer
  </entry>
</properties>
```

The following example output shows the keys used for each data element in the agent status of a protocol bridge agent:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="agentOsName">Windows 7</entry>
  <entry key="agentDescription"/>
  <entry key="queueManager">QM1</entry>
  <entry key="agentTimeZone">Europe/London</entry>
  <entry key="agentVersion">1.00</entry>
  <entry key="agentName">BRIDGE</entry>
  <entry key="protocolBridgeType">ftp</entry>
  <entry key="protocolBridgeServerHost">ftpserver.example.org</entry>
  <entry key="maxDestinationTransfers">25</entry>
  <entry key="maxSourceTransfers">25</entry>
  <entry key="maxQueuedTransfers">100</entry>
  <entry key="DestinationTransferStates">414d51204d554e474f202020202020d857374a60a72622=RunningTransfer
  </entry>
  <entry key="SourceTransferStates">414d51204d554e474f202020202020d857374a93a72622=NegotiatingTransfer
  </entry>
</properties>
```

## Related reference

[“MFT agent transfer states” on page 2617](#)



A Managed File Transfer Agent that is started publishes its details to the SYSTEM.FTE topic on its coordination queue manager. These details include the states of each of the current transfers that involved that agent.

“File transfer request message format” on page 2665

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the FileTransfer.xsd schema and have the <request> element as the root element. The FileTransfer.xsd schema document is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

“File transfer status message format” on page 2627

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its SYSTEM.FTE/Transfers/agent\_name/transfer ID topic), which conforms to the TransferStatus.xsd XML schema. The TransferStatus.xsd file is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory of your WMQMFT installation.

“File transfer log message formats” on page 2631

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of Log/agent\_name/transfer\_id. These messages conform to the schema TransferLog.xsd, which is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory of your Managed File Transfer installation.

“Scheduled file transfer log message formats” on page 2653

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/agent\_name/schedule ID topic). This message conforms to the ScheduleLog.xsd XML schema.

“MFT monitor request message formats” on page 2682

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the **fteCreateMonitor** command or using the IBM MQ Explorer interface.

“MFT message formats for security” on page 2695

This topic describes the messages published to the Managed File Transfer coordination queue manager relevant to security.

**MFT agent transfer states**

A Managed File Transfer Agent that is started publishes its details to the SYSTEM.FTE topic on its coordination queue manager. These details include the states of each of the current transfers that involved that agent.

<i>Table 377. Agent transfer state names and explanations</i>	
<b>Transfer state</b>	<b>Explanation</b>
CancelledInProgressTransfer	A source agent has received a cancel message for an in-progress transfer.
CancelledNewTransfer	A source agent has received a cancel message for a new transfer.
CompletedTransfer	A destination agent has completed the transfer and has sent a completion message to the source agent. The destination agent is waiting for an acknowledgment message from the source agent.
CompleteReceivedTransfer	A source agent has received a completion message from the destination agent and has sent a message back to the destination agent to acknowledge the completion message.

Table 377. Agent transfer state names and explanations (continued)

Transfer state	Explanation
FailedTransferEnding	The transfer has failed but the completion log message has not been published and the transfer has not been removed from the state store. For example, this state can occur if an agent process is stopped after a failure response has been received from the destination agent but before the subsequent processing has been completed.
NegotiatingTransfer	A source agent is in negotiation with the destination agent before running a transfer.
NewReceiverTransfer	A new transfer has been created at the destination agent as part of negotiation, but the transfer is not yet running.
NewSenderTransfer	A new transfer from the source agent that the negotiation has not started for.
RecoveringTransfer	When either a source or destination agent starts the recovery process, any transfers in running state are moved into transfer state. Transfers are moved out of this state into ReSynchronisingTransfer state when a resynchronization message is sent to the peer agent.  For example, if the destination agent starts the recovery process for a running transfer, the transfer is moved into the ReSynchronisingTransfer state when a resynchronization message is sent to its source agent.
RecoveryTimedOut	From IBM MQ 9.1.5, if a transfer recovery timeout is set for a transfer, the source agent moves the transfer into this state if transfer recovery times out. After the transfer has been resynchronized, the destination agent removes any part files that were created during the transfer and sends a completion message to the source agent.
RestartingTransfer	A source or destination agent has received a resynchronize request message and is waiting for the respective destination or source agent to restart.
ResumingTransfer	A source agent has received a resynchronize response message and now schedules the transfer to restart.
ReSynchronisingTransfer	A transfer source or destination agent has found a problem and has sent a resynchronization message to its respective destination or source agent.
RunningTransfer	A transfer from either a source agent or destination agent that is in the normal running state
WaitingForDestinationCapacity	A source agent has received a DESTINATION_CAPACITY_EXCEEDED error from the destination agent. The transfer is now in a waiting state to be retried after a period.

**Related reference**

[“MFT agent status values” on page 2485](#)

The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

## MFT monitor list message format

The XML messages that are published as retained publications to the topic string `SYSTEM.FTE/monitors/agent_name/monitor_name` conform to the `MonitorList.xsd` schema. Each XML message lists an active monitor belonging to that agent. This information is used by the **fteListMonitors** command and the IBM MQ Explorer plug-in to display a list of monitors to the user. The `MonitorList.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `MonitorList.xsd` schema imports `Monitor.xsd`, which is in the same directory.

## Schema

The following schema describes which elements are valid in a monitor list XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  targetNamespace="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  xmlns="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">

  <xsd:include schemaLocation="Monitor.xsd"/>

  <xsd:element name="monitorList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="status" type="monitorStatusType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="configuration" type="monitorConfigurationType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="pollInterval" type="pollIntervalType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="batch" type="batchType" minOccurs="1" maxOccurs="1"/>
        <xsd:any minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="agent" type="xsd:string" use="required"/>
      <xsd:attribute name="monitor" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="monitorStatusType">
    <xsd:sequence>
      <xsd:any minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="state" type="xsd:token"/>
    <xsd:anyAttribute/>
  </xsd:complexType>

  <xsd:complexType name="monitorConfigurationType">
    <xsd:sequence>
      <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1" />
      <xsd:element name="resources" type="monitorResourcesType" minOccurs="0" maxOccurs="1" />
      <xsd:element name="triggerMatch" type="triggerMatchType" minOccurs="0" maxOccurs="1" />
      <xsd:element name="tasks" type="monitorListTasksType" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:anyAttribute/>
  </xsd:complexType>

  <xsd:complexType name="monitorListTasksType">
    <xsd:sequence>
      <xsd:element name="task" type="monitorListTaskType" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="monitorListTaskType">
    <xsd:sequence>
      <xsd:element name="name" type="monitorTaskNameType" minOccurs="0" maxOccurs="1" />
      <xsd:element name="description" type="xsd:string" minOccurs="0" maxOccurs="1" />
      <xsd:element name="taskXML" type="xsd:string" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

## Understanding the monitor list message

The elements and attributes used in the monitor list messages are described in the following list:

### <monitorList>

Group element containing the elements describe a monitor that is defined for the agent.

Attribute	Description
agent	Required. The name of the agent that the resource monitor is defined on.
monitor	Required. The name of the monitor. Unique for this agent.
version	Required. The version of the monitor list message format.

**<status>**

The status of the monitor.

Attribute	Description
state	The state of the monitor.

**<configuration>**

Group element containing the elements describe the configuration of the monitor.

**<description>**

A description of the monitor. (Not currently used.)

**<resources>**

The resource or resources being monitored.

**<directory>**

A directory to monitor.

Attribute	Description
recursionLevel	The number of directory levels down from the top level to monitor.
id	The ID of the resource.

**<queue>**

A queue to monitor.

Attribute	Description
id	The ID of the resource.

**<triggerMatch>**

Element that contains the <conditions> element.

**<conditions>**

Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain only one of the following elements: <allOf>, <anyOf>, or <condition>.

**<allOf>**

Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain one or many <condition> elements. For the resource monitor to be triggered all of the conditions inside of this element must be met.

**<anyOf>**

Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain one or many <condition> elements. For the resource monitor to be triggered only one of the conditions inside of this element must be met.

**<condition>**

Element that contains a single condition that the resource monitor is monitoring for. This element can contain only one of the following elements: <fileMatch>, <fileNoMatch>, <fileSize>, <queueNotEmpty>, <completeGroups>, or <fileSizeSame>. It can also contain a <name> element and a <resource> element.

If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:

- fileMatch
- fileNoMatch
- fileSize

If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:

- queueNotEmpty
- completeGroups

**<fileMatch>**

Group element for a file name match condition.

**<pattern>**

Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is \* (any file will match).

**<fileNoMatch>**

Group element for an inverse file name match condition.

**<pattern>**

Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is \* (the absence of any file will match).

**<fileSize>**

Group element for a file size comparison.

**<compare>**

Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only >=' is supported.
units	Specifies file size units, which can be one of: <ul style="list-style-type: none"><li>• B - bytes</li><li>• KB - kilobytes</li><li>• MB - megabytes</li><li>• GB - gigabytes</li></ul> The units value is case insensitive, so mb' works as well as MB'.

**<pattern>**

File name pattern to match. Default is \* (any file will match).

**<queueNotEmpty>**

This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.

**<completeGroups>**

This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

**<name>**

Name of the condition.

**<resource>**

Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

**<tasks>**

Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

**<task>**

Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

**<name>**

Name of the task. Accepts any alphanumeric characters.

**<description>**

Description of the task. Any text value is allowed.

**<taskXML>**

The XML message that describes the task that the monitor is to perform. The contents of this element are in an escaped XML format.

**<pollInterval>**

The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	Specifies the time units for the poll interval. Valid values are: <ul style="list-style-type: none"> <li>• seconds</li> <li>• minutes</li> <li>• hours</li> <li>• days</li> <li>• weeks</li> <li>• months</li> <li>• years</li> </ul>

**<batch>**

The maximum number of trigger matches to include in a single batch.

Attribute	Description
maxSize	The maximum number of trigger matches to include in a single batch

The following XML shows an example of a retained publication which is published to the topic string SYSTEM.FTE/monitors/*agent\_name*/MONITORTWO when the monitor called MONITORTWO is created on AGENT\_JUPITER. The escaped XML within the <taskXML> element describes the task that is submitted when the monitor condition is met.

```
<?xml version="1.0" encoding="UTF-8"?>
<lst:monitorList xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:lst="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition MonitorList.xsd"
  version="4.00"
  agent="AGENT_JUPITER"
  monitor="MONITORTWO">
  <status state="started"/>
  <configuration>
    <description/>
    <resources>
      <directory recursionLevel="0" id="">/srv/nfs/incoming</directory>
    </resources>
    <triggerMatch>
      <conditions>
        <condition>
          <name/>
          <resource id=""/>
          <fileMatch>
            <pattern>*.completed</pattern>
          </fileMatch>
        </condition>
      </conditions>
    </triggerMatch>
    <tasks>
      <task>
        <name/>
        <description/>
        <taskXML>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
          xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="4.00"
          xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
            &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
            &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
            &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
            &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
          &lt;/managedTransfer&gt;
        &lt;/request&gt;
      </taskXML>
    </task>
  </tasks>
</configuration>
</monitorList>
```

```

        &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
        &lt;source disposition="leave" recursive="false"&gt;&lt;file
        &gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
        &lt;destination exist="error" type="directory"&gt;
        &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
        &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;
        &lt;/request&gt;
    </taskXML>
</task>
</tasks>
</configuration>
<pollInterval units="minutes">1</pollInterval>
<batch maxSize="1"/>
</lst:monitorList>

```

## MFT schedule list message format

The XML message that is published to a retained publication to the topic string `SYSTEM.FTE/Scheduler/agent_name` conforms to the `ScheduleList.xsd` schema. This XML message lists all active schedules belonging to that agent. This information is used by the `fteListScheduledTransfers` command and the IBM MQ Explorer to display a list of schedules to the user. The `ScheduleList.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `ScheduleList.xsd` schema imports `FileTransfer.xsd`, which is in the same directory.

## Schema

The following schema describes which elements are valid in a monitor list XML message.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="FileTransfer.xsd"/>

    <xsd:element name="schedules">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="managedTransfer" type="scheduledManagedTransferType" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required"/>
            <xsd:attribute name="size" type="xsd:nonNegativeInteger" use="required"/>
            <xsd:attribute name="agent" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="scheduledManagedTransferType">
        <xsd:sequence>
            <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="schedule" type="scheduleListType" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="sourceAgent" type="agentType" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="destinationAgent" type="agentClientType" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="trigger" type="triggerType" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="transferSet" type="transferSetType" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="idType" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="scheduleListType">
        <xsd:sequence>
            <xsd:element name="submit" type="submitType" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="repeat" type="repeatType" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="next" type="noZoneTimeType" maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>

```

## Understanding the schedule list message

The elements and attributes used in the schedule list messages are described in the following list:

### <schedules>

Group element containing information about all of the schedules defined on a single agent.

Attribute	Description
agent	Required. The name of the source agent that the schedule is defined on.
size	Required. The number of schedules defined on this agent.
version	Required. The version of the schedule list message format.

**<managedTransfer>**

Group element containing information about a single schedule.

Attribute	Description
id	Required. The hexadecimal string ID of the schedule request message.

**<originator>**

The originator of the schedule request.

**<hostName>**

The host name of the machine that the schedule request was submitted from.

**<userID>**

The user ID of the user that submitted the schedule request.

**<mqmdUserID>**

The MQMD user ID of the user that submitted the schedule request.

**<schedule>**

Element that contains the elements that describe when the scheduled transfer occurs.

**<submit>**

Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. The value of this attribute can be one of the following values: <ul style="list-style-type: none"> <li>• source - use the time zone of the source agent</li> <li>• admin - use the time zone of the administrator issuing the command</li> <li>• UTC - use Coordinated Universal Time</li> </ul>
timezone	The time zone description according to the timebase value

**<repeat>**

Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

Attribute	Description
interval	The interval units, which must be one of the following values: <ul style="list-style-type: none"> <li>• minutes</li> <li>• hours</li> <li>• days</li> <li>• weeks</li> <li>• months</li> <li>• years</li> </ul>

**<frequency>**

The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: <ul style="list-style-type: none"> <li>• minutes</li> <li>• hours</li> </ul>



Attribute	Description
	<ul style="list-style-type: none"> <li>• days</li> <li>• weeks</li> <li>• months</li> <li>• years</li> </ul>

**<expireTime>**

Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

**<expireCount>**

Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

**<next>**

Specifies the date and time when the next scheduled transfer is due to start.

**<sourceAgent>**

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

**<destinationAgent>**

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

**<trigger>**

Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The following are valid values: <ul style="list-style-type: none"> <li>• yes - log entries are created for failed triggered transfers</li> <li>• no - log entries are not created for failed triggered transfers</li> </ul>

**<reply>**

Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file or the default of `WMQFTE.*` if not specified.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

**<transferSet>**

Specifies a group of file transfers you want the scheduled transfer to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

### <job>

Optional group element containing job information for the entire transfer specification. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that appears in the transfer log message, which is described in the following topic: [“File transfer log message formats” on page 2631.](#)

### Example

```
<?xml version="1.0" encoding="UTF-8"?>
<schedules xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  size="2"
  version="4.00"
  agent="AGENT_JUPITER"
  xsi:noNamespaceSchemaLocation="ScheduleList.xsd">
  <managedTransfer id="1">
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <schedule>
      <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00+0000</
submit>
      <next>2010-01-01T21:00+0000</next>
    </schedule>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <destinationAgent agent="AGENT_SATURN" QMgr="QM_JUPITER"/>
    <reply QMgr="QM_JUPITER">WMQFTE.4D400F8B20004E06</reply>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>/etc/passwd</file>
        </source>
        <destination type="directory" exist="overwrite">
          <file>/tmp</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
  <managedTransfer id="2">
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <schedule>
      <submit timebase="admin" timezone="Europe/London">2010-12-31T09:00+0000</
submit>
      <next>2010-12-31T09:00+0000</next>
    </schedule>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <destinationAgent agent="AGENT_NEPTUNE" QMgr="QM_JUPITER"/>
    <reply QMgr="QM_JUPITER">WMQFTE.4D400F8B20004E09</reply>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>/etc/passwd</file>
        </source>
        <destination type="directory" exist="overwrite">
          <file>/tmp</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</schedules>
```

## MFT example template XML message

When a template is created, a message is published to the SYSTEM.FTE topic with a topic string of `Templates/template_id`. This example XML describes a single template defined in your Managed File Transfer network.

```
<?xml version="1.0" encoding="UTF-8"?>
<transferTemplate version="4.00" id="baf9df73-45c2-4bb0-a085-292232ab66bc">
  <name>BASIC_TEMPLATE</name>
  <sourceAgentName>AGENT_JUPITER</sourceAgentName>
  <sourceAgentQMgr>QM_JUPITER</sourceAgentQMgr>
  <destinationAgentName>AGENT_SATURN</destinationAgentName>
  <destinationAgentQMgr>QM_JUPITER</destinationAgentQMgr>
  <fileSpecs>
    <item mode="binary" checksumMethod="MD5">
      <source recursive="false" disposition="leave">
        <file>/etc/passwd</file>
      </source>
      <destination type="directory" exist="overwrite">
        <file>/tmp</file>
      </destination>
    </item>
  </fileSpecs>
  <priority>0</priority>
</transferTemplate>
```

### Related tasks

[Creating a file transfer template using IBM MQ Explorer](#)

### Related reference

[“fteCreateTemplate \(create new file transfer template\)” on page 2033](#)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn template\_name** parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

## File transfer status message format

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its `SYSTEM.FTE/Transfers/agent_name/transfer ID` topic), which conforms to the `TransferStatus.xsd` XML schema. The `TransferStatus.xsd` file is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your WMQMFT installation.

## Schema

The following schema describes which elements are valid in a transfer status XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <xsd:element name="transaction">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="sourceAgent" type="agentType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="destinationAgent" type="agentType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="transferSet" type="transferSetType" maxOccurs="1" minOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="ID" type="IDType" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="transferSetType">
```

```

    <xsd:sequence>
      <xsd:element name="stats" type="statsType"
        maxOccurs="1" minOccurs="1" />
      <xsd:element name="current" type="currentType"
        maxOccurs="1" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="time" type="xsd:dateTime" use="required" />
  </xsd:complexType>

  <xsd:complexType name="currentType">
    <xsd:sequence>
      <xsd:element name="source" type="fileSourceType"
        maxOccurs="1" minOccurs="1" />
      <xsd:element name="destination" type="fileDestinationType"
        maxOccurs="1" minOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="transferred" type="xsd:nonNegativeInteger"
      use="required" />
    <xsd:attribute name="size" type="xsd:nonNegativeInteger" use="required" />
  </xsd:complexType>

  <xsd:complexType name="statsType">
    <xsd:attribute name="bytes" type="xsd:nonNegativeInteger"
      use="required" />
    <xsd:attribute name="seconds" type="xsd:decimal"
      use="required" />
    <xsd:attribute name="currentItem" type="xsd:nonNegativeInteger"
      use="required" />
    <xsd:attribute name="totalItems" type="xsd:nonNegativeInteger" use="required" />
  </xsd:complexType>
</xsd:schema>

```

## Understanding the transfer status message

The elements and attributes used in the transfer status messages are described in the following list:

### <transaction>

Group element that contains all of the elements for the file transfers.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.
ID	The unique identifier for the file transfer.

### <sourceAgent>

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	The name of the agent.
QMgr	The name of the agent queue manager.

### <destinationAgent>

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	The name of the agent.
QMgr	The name of the agent queue manager.

### <transferset>

Specifies a group of file transfers being performed together. All of the files in the transfer must originate at the same source agent and end at the same destination agent.

Attribute	Description
time	Specifies the date and time (in date time format).

**<stats>**

Required. Defines metrics about the transfer, including the number of bytes copied so far, in the given number of seconds. Also supplies the current item number out of the total number of items in the <transferSet>.

Attribute	Description
bytes	Number of bytes copied so far.
seconds	Number of seconds taken to transfer those bytes.
currentItem	The index of the current item being transferred.
totalItems	The total number of items being transferred.

**<current>**

Optional element. Group element that contains elements that specify the file transfer currently in progress. The <current> element indicates how many bytes of data have been transferred so far for the current item and the expected total number of bytes

**<source>**

Group element that contains the element specifying the source file name.

**<file>**

Specifies the source path of the file that is being transferred. The path is as specified for the transfer. This path might differ from the path that is output as part of the transfer log, which is the absolute form the of path.

**<destination>**

Group element that contains the element specifying the destination file name or specification.

**<file>**

Specifies the destination path of the file that is being transferred. The path is as specified for the transfer. This path might differ from the path that is output as part of the transfer log, which is the absolute form the of path.

Attribute	Description
alias	Specifies an alias for the destination file. This alias is the name of the source file, excluding any directory path specified for the transfer.
filesystem	Specifies the name of the file space where the destination file is written.

**<queue>**

When used with the <destination> element, specifies the name of the queue you want to transfer to. This name is in the format QUEUE or QUEUE@QUEUE\_MANAGER.

**Related reference**

[“File transfer progress message examples” on page 2630](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Transfers/agent_name/transfer_id`. The XML examples show the progress message for a single file transfer and for a multiple file transfer.

[“MFT agent status message format” on page 2615](#)

When a Managed File Transfer Agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the `SYSTEM.FTE/Agents/agent name` topic).

[“File transfer request message format” on page 2665](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must

conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

#### “File transfer log message formats” on page 2631

File transfer log messages are published to the `SYSTEM.FTE` topic with a topic string of `Log/agent_name/transfer_id`. These messages conform to the schema `TransferLog.xsd`, which is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your Managed File Transfer installation.

#### “Scheduled file transfer log message formats” on page 2653

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its `SYSTEM.FTE/Log/agent name/schedule ID` topic). This message conforms to the `ScheduleLog.xsd` XML schema.

#### “MFT monitor request message formats” on page 2682

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the `fteCreateMonitor` command or using the IBM MQ Explorer interface.

#### “MFT message formats for security” on page 2695

This topic describes the messages published to the Managed File Transfer coordination queue manager relevant to security.

### **File transfer progress message examples**

When a transfer is in progress, messages are published to the `SYSTEM.FTE` topic with a topic string of `Transfers/agent_name/transfer_id`. The XML examples show the progress message for a single file transfer and for a multiple file transfer.

#### **Single file transfer**

The following example shows the details of a single file transfer that is in progress.

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d223d0020"
  xsi:noNamespaceSchemaLocation="TransferStatus.xsd">
  <sourceAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <destinationAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <transferSet time="2011-01-26T13:03:26.542Z">
  <stats bytes="1198" seconds="0.018" currentItem="1" totalItems="1"/>
  <current transferred="1151" size="1151">
  <source>
  <file>/etc/passwd</file>
  </source>
  <destination>
  <file>/tmp/passwd</file>
  </destination>
  </current>
  </transferSet>
</transaction>
```

#### **Multiple file transfer**

If there were more files in the transfer set, the transfer status message indicates which one is being processed and how many bytes have been transferred so far.

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d035c0020"
  xsi:noNamespaceSchemaLocation="TransferStatus.xsd">
  <sourceAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <destinationAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <transferSet time="2011-01-26T13:12:58.636Z">
  <stats bytes="440" seconds="0.082" currentItem="10" totalItems="10"/>
```

```

    <current transferred="0" size="0">
      <source>
        <file>/srv/nfs/incoming/file10.txt</file>
      </source>
      <destination>
        <file>/srv/nfs/outgoing/file10.txt</file>
      </destination>
    </current>
  </transferSet>
</transaction>

```

## File transfer log message formats

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. These messages conform to the schema `TransferLog.xsd`, which is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your Managed File Transfer installation.

If you want to monitor file transfers or collect data about them, set up a subscription to a wildcard topic tailored to the transfers you are interested in. For example:

```
Log/#
```

or,

```
Log/FTEAGENT/#
```

This subscription can be durable or non-durable. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. Non-durable subscriptions exist only as long as a subscribing application's connection to the queue manager remains open.

## Schema

The following schema describes which elements are valid in a transfer log XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <xsd:element name="transaction">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="action" type="actionType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceAgent" type="agentExitStatusType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceWebGateway" type="webGatewayType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceWebUser" type="webUserType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="destinationAgent" type="agentExitStatusType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="destinationWebGateway" type="webGatewayType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="destinationWebUser" type="webUserType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="agent" type="agentExitStatusType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="originator" type="origRequestType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="status" type="statusType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="trigger" type="triggerType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="transferSet" type="transferSetType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="job" type="jobType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="scheduleLog" type="scheduleLogType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="statistics" type="statisticsType"

```

```

        maxOccurs="1"                minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required" />
    <xsd:attribute name="ID" type="IDType" use="required" />
    <xsd:attribute name="relatedID" type="IDType" use="optional" />
    <xsd:attribute name="agentRole" type="agentRoleType" use="optional" />
</xsd:complexType>
</xsd:element>

<xsd:complexType name="agentExitStatusType">
  <xsd:complexContent>
    <xsd:extension base="agentType">
      <xsd:sequence>
        <xsd:element name="startExits" type="exitGroupType" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="endExits" type="exitGroupType" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="systemInfo" type="systemInfoType" minOccurs="0"
maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="transferSetType">
  <xsd:sequence>
    <xsd:element name="metaDataSet" type="metaDataSetType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="call" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="preSourceCall" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="postSourceCall" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="preDestinationCall" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="postDestinationCall" type="callGroupType"
maxOccurs="1" minOccurs="0" />
    <xsd:element name="item" type="itemType"
maxOccurs="unbounded" minOccurs="0" />
  </xsd:sequence>
  <xsd:attribute name="index" type="xsd:nonNegativeInteger" use="optional" />
  <xsd:attribute name="size" type="xsd:nonNegativeInteger" use="optional" />
  <xsd:attribute name="startTime" type="xsd:dateTime" use="required" />
  <xsd:attribute name="total" type="xsd:nonNegativeInteger" use="required" />
  <xsd:attribute name="bytesSent" type="xsd:nonNegativeInteger" use="required" />
</xsd:complexType>

<xsd:complexType name="itemType">
  <xsd:sequence>
    <xsd:element name="source" type="fileSourceChecksumType"
maxOccurs="1" minOccurs="1" />
    <xsd:element name="destination" type="fileDestinationChecksumType"
maxOccurs="1" minOccurs="1" />
    <xsd:element name="status" type="statusType"
maxOccurs="1" minOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="mode" type="modeType" use="required" />
</xsd:complexType>

<xsd:complexType name="fileSourceChecksumType">
  <xsd:complexContent>
    <xsd:extension base="fileSourceType">
      <xsd:sequence>
        <xsd:element name="checksum" type="checksumType" minOccurs="0"
maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="fileDestinationChecksumType">
  <xsd:complexContent>
    <xsd:extension base="fileDestinationType">
      <xsd:sequence>
        <xsd:element name="checksum" type="checksumType"
minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```



```

<xsd:complexType name="actionType">
  <xsd:simpleContent>
    <xsd:extension base="actionEnumType">
      <xsd:attribute name="time" type="xsd:dateTime" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="actionEnumType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="cancelled"/>
    <xsd:enumeration value="started"/>
    <xsd:enumeration value="progress"/>
    <xsd:enumeration value="completed"/>
    <xsd:enumeration value="malformed"/>
    <xsd:enumeration value="notAuthorized"/>
    <xsd:enumeration value="deleted"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="systemInfoType">
  <xsd:attribute name="architecture" type="xsd:string" use="required"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="version" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:element name="malformed">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="action" type="actionType"
        maxOccurs="1" minOccurs="1"/>
      <xsd:element name="agent" type="agentExitStatusType"
        maxOccurs="1" minOccurs="0"/>
      <xsd:element name="status" type="statusType"
        maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required"/>
    <xsd:attribute name="ID" type="IDType" use="required"/>
    <xsd:attribute name="agentRole" type="agentRoleType" use="required"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="notAuthorized">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="action" type="actionType"
        maxOccurs="1" minOccurs="1"/>
      <xsd:element name="originator" type="origRequestType"
        maxOccurs="1" minOccurs="1"/>
      <xsd:element name="authority" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="status" type="statusType"
        maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required"/>
    <xsd:attribute name="ID" type="IDType" use="required"/>
    <xsd:attribute name="agentRole" type="agentRoleType" use="required"/>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="statisticsType">
  <xsd:sequence>
    <xsd:element name="actualStartTime" type="xsd:dateTime"
      maxOccurs="1" minOccurs="0"/>
    <xsd:element name="retryCount" type="xsd:nonNegativeInteger"
      maxOccurs="1" minOccurs="1"/>
    <xsd:element name="numFileFailures" type="xsd:nonNegativeInteger"
      maxOccurs="1" minOccurs="1"/>
    <xsd:element name="numFileWarnings" type="xsd:nonNegativeInteger"
      maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="webGatewayType">
  <xsd:attribute name="webGatewayName" type="xsd:string" use="optional" />
  <xsd:attribute name="webGatewayAgentName" type="xsd:string" use="optional" />
  <xsd:attribute name="webGatewayAgentQMgr" type="xsd:string" use="optional" />
</xsd:complexType>

<xsd:complexType name="webUserType">
  <xsd:attribute name="webGatewayName" type="xsd:string" use="required" />
  <xsd:attribute name="webGatewayAgentName" type="xsd:string" use="optional" />

```

```

<xsd:attribute name="webGatewayAgentQMgr" type="xsd:string" use="optional" />
</xsd:complexType>
</xsd:schema>

```

**Note:** From IBM MQ 9.0, Managed File Transfer does not support the Web Gateway or web agents.

## Understanding the transfer log message

### <transaction>

Group element that specifies a group of transfers you want to perform together.

Attribute	Description
version	Specifies the version of this element as detailed by Managed File Transfer.
ID	Specifies the unique transaction ID. The ID can be a maximum of 48 alphanumeric characters.
relatedID	Optional. If the transaction is the delete or download of a file from a file space, <b>relatedID</b> specifies the transaction ID of the transfer that uploaded the file to the file space.
agentRole	Optional. Specifies whether the agent concerned is on the source or destination system
xmlns:xsi	Namespace declaration. Indicates that the elements and data types used in this schema derive from the "https://www.w3.org/2001/XMLSchema-instance" namespace.
xsi:noNamespaceSchemaLocation	Specifies the name and location of the XML schema document to validate this message against if there is no namespace declaration. The value you specify for this attribute must refer to a Managed File Transfer TransferLog.xsd document.

### <action>

Describes the status of the file transfer at the time logged by the time attribute. The status can be one of the following values:

- started
- progress
- completed
- cancelled
- malformed (indicates the file transfer request message content can not be interpreted.)
- notAuthorized
- deleted

Attribute	Description
time	The time that the transfer status was captured, expressed in UTC format.

### <sourceAgent>

Specifies the name of the agent on the system where the source file is located. Only <sourceAgent> or <sourceWebUser> can be specified.

### <startExits>

Group element that contains one or more user exit elements. This element can occur once only.

### <endExits>

Group element that contains one or more user exit elements. This element can occur once only.

**<systemInfo>**

Describes the system architecture, name, and version. This element can occur once only.

Attribute	Description
agent	The name of the agent on the source system.
QMgr	The name of the queue manager on the source system.
agentType	The type of the agent. Valid values are: <ul style="list-style-type: none"> <li>• STANDARD - a normal agent</li> <li>• BRIDGE - a protocol bridge agent</li> <li>• CD_BRIDGE - a Connect:Direct bridge agent</li> <li>• EMBEDDED - an embedded agent</li> <li>• SFG - a Sterling File Gateway embedded agent</li> </ul>
bridgeURL	Optional. If the agent is a protocol bridge agent, the host name of the system hosting the protocol server.
pnode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct primary node involved in the transfer.
snode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct secondary node involved in the transfer.
bridgeNode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct node that is part of the Connect:Direct bridge. This is the same node as either the primary node or the secondary node.

**<destinationAgent>**

Specifies the name of the agent on the system the file was transferred to. Either <destinationAgent> or <destinationWebUser> can be specified.

Attribute	Description
agent	The name of the agent on the destination system.
QMgr	The name of the queue manager on the destination system.
agentType	The type of the agent. Valid values are: <ul style="list-style-type: none"> <li>• STANDARD - a normal agent</li> <li>• BRIDGE - a protocol bridge agent</li> <li>• CD_BRIDGE - a Connect:Direct bridge agent</li> <li>• EMBEDDED - an embedded agent</li> <li>• SFG - a Sterling File Gateway embedded agent</li> </ul>
bridgeURL	Optional. If the agent is a protocol bridge agent, the host name of the system hosting the protocol server.
pnode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct primary node involved in the transfer.
snode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct secondary node involved in the transfer.
bridgeNode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct node that is part of the Connect:Direct bridge. This is the same node as either the primary node or the secondary node.

**<startExits>**

Group element that contains one or more user exit elements. This element can occur once only.

**<endExits>**

Group element that contains one or more user exit elements. This element can occur once only.

**<systemInfo>**

Describes the system architecture, name, and version. This element can occur once only.

**<originator>**

Group element that contains the elements specifying the originator of the request.

**<hostName>**

The host name of the system where the source file is located.

**<userID>**

The user ID that originated the file transfer.

**<mqmdUserID>**

The IBM MQ user ID that was supplied in the message descriptor (MQMD)

**<webUserID>**

Optional. The user ID that was supplied to the web browser submitting the transfer request.

**<webBrowser>**

Optional. The web browser that the transfer request was submitted from.

**<status>**

The result code and supplement messages.

**<trigger>**

Group element that contains the trigger elements defined in the original transfer request. These elements can be either or both of the following:

**<fileExist>**

Trigger condition based on whether a file exists

**<fileSize>**

Trigger condition based on whether a file meets or exceeds the specified size

**<transferSet>**

Specifies a group of file transfers you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
startTime	Records the time that the set of transfers started, expressed in UTC format.
total	Specifies the total number of items in this set of transfers.
index	Optional attribute. Specifies the position of the first item in progress of the transfer set. The index attribute increments from zero. For example, if the index is set to 1, the progress message is the second of two items.
size	Optional attribute. Specifies the number of items in the progress report.
priority	Optional attribute. Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the source agent priority level.

**<metaDataSet>**

Group element containing one or more of the following attributes:

**<metaData>**

Attribute	Description
key	The key half of a metadata key-value pair. The <metaData> element content contains the value half of the pair. For example <metaData key="testkey1">testvalue1</metaData>

**<job>**

Group element that contains an element specifying job details. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that is included in the transfer request message, which is described in the following topic: [“File transfer request message format” on page 2665.](#)

**<name>**

The value of name can be any string.

**<scheduleLog>**

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
ID	Matches the schedule ID if the transfer is a scheduled transfer.

**<item>**

Group element that contains elements specifying the source and destination file names and locations.

**<source>**

Group element that contains the <file> element or the <queue> element, and the <checksum> element for the file on the source system.

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid options are as follows: <ul style="list-style-type: none"> <li>• leave - the source files are left unchanged.</li> <li>• delete - the source files are deleted from the source system after the source file is successfully transferred.</li> </ul>
correlationBoolean	A boolean correlation value. If the source is a Connect:Direct bridge, this specifies whether the Connect:Direct process is user-defined.
correlationString1	A string correlation value. If the source is a Connect:Direct bridge, this specifies the name of the Connect:Direct process that occurs at the destination of the transfer.
correlationNum1	A numeric correlation value. If the source is a Connect:Direct bridge, this specifies the ID number of the Connect:Direct process that occurs at the destination of the transfer.

**<queue>**

When used with the <source> element, specifies the name of the queue that the transferred messages were read from, which is located on the source agent queue manager.



Attribute	Description
messageCount	The number of messages that were read from the queue.

Attribute	Description
groupId	The IBM MQ group ID of the messages read from the queue.

**<destination>**

Group element that contains the <file> element or the <queue> element, and <checksum> element for the destination.

Only one of <file> and <queue> is present as a child element of destination.

Attribute	Description
type	<p>The type of destination. The valid options are as follows:</p> <ul style="list-style-type: none"> <li>• queue - specifies an IBM MQ queue as the destination</li> <li>• file - specifies a file as the destination</li> <li>• directory - specifies a directory as the destination</li> <li>•  dataset - specifies a z/OS data set as the destination</li> <li>•  pds - specifies a z/OS partitioned data set as the destination</li> </ul> <p>The option queue can be present only when the <b>&lt;destination&gt;</b> element has a child element of <b>&lt;queue&gt;</b>. The other options can be present only when the <b>&lt;destination&gt;</b> element has a child element of <b>&lt;file&gt;</b>.</p>
exist	<p>Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:</p> <ul style="list-style-type: none"> <li>• error - reports an error and the file is not transferred.</li> <li>• overwrite - overwrites the existing destination file.</li> </ul> <p>This attribute cannot be present if the <b>&lt;destination&gt;</b> element has a child element of <b>&lt;queue&gt;</b>.</p>
correlationBoolean	A boolean correlation value. If the destination is a Connect:Direct bridge, this specifies whether the Connect:Direct process is user-defined.
correlationString1	A string correlation value. If the destination is a Connect:Direct bridge, this specifies the name of the Connect:Direct process that occurs at the destination of the transfer.
correlationNum1	A numeric correlation value. If the destination is a Connect:Direct bridge, this specifies the ID number of the Connect:Direct process that occurs at the destination of the transfer.

**<file>**

Specifies the absolute path of the file that was transferred (both at the source and destination). The fully-qualified path is in the format consistent with your operating system, for example C:/from/here.txt. File URIs are not used.

**<queue>**

When used with the <destination> element, specifies the name of the queue that was transferred to, which is located on any queue manager that is connected to the destination agent queue manager.

Attribute	Description
messageCount	The number of messages that were written to the queue.
messageLength	The length of the messages written to the queue.

Attribute	Description
groupId	If the transfer request specified that the file is split into multiple messages, the value of this attribute is the IBM MQ group ID of the messages written to the queue.
messageId	If the transfer request did not specify that the file is split into multiple messages, the value of this attribute is the IBM MQ message ID of the message written to the queue.

#### <checksum>

Optional element.

Specifies the type of hash algorithm that generated the message digest to create the digital signature. Currently Managed File Transfer supports Message Digest algorithm 5 (MD5) only. The checksum provides a way for you to confirm the integrity of transferred files is intact.

#### <malformed>

Group element for malformed messages.

Attribute	Description
version	
ID	
agentRole	Either source agent or destination agent

#### <statistics>

Group element for statistical information for the transfer (when available).

##### <actualStartTime>

The actual time that the agent started running the transfer. Typically, the time is the same as (or very close to) the start time recorded for the transfer. However, when an agent is busy submitted transfers might be queued until the agent has capacity to run the transfers.

##### <retryCount>

The number of times that the transfer went into the recovery state and was retried by the agent. A transfer can go into a recovery state because the source and destination agents lose communication, either because of an IBM MQ network error or because they are not receiving data or acknowledgment messages for a period. This period is determined by the agent properties: transferAckTimeout and transferAckTimeoutRetries.

##### <numFileFailures>

The number of files in the transferSet that failed to transfer successfully.

##### <numFileWarnings>

The number of files in the transferSet that generated warnings while being transferred, but otherwise transferred successfully.

### Examples

Examples of XML messages that conform to this schema are provided for each of the following types of transfer:

- [A transfer of a single file](#)
- [A transfer that contains multiple files](#)
- [A failed file transfer](#)
- [A transfer defined with a trigger](#)
- [A transfer started by a schedule](#)
- [A transfer that calls user exits](#)
- [A transfer through a Connect:Direct bridge node](#)

## Related reference

[“Single transfer log message examples” on page 2640](#)

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages for a single file transfer being started, in progress, and completed.

[“Multiple file transfer log message examples” on page 2642](#)

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id` when a transfer that contains multiple files occurs.

[“Failed file transfer log message examples” on page 2644](#)

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages for a file transfer that fails being started, in progress, and completed.

[“Triggered file transfer log message example” on page 2646](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML example shows the log message that is created when a file transfer containing a trigger condition is started.

[“MFT user exit log message examples” on page 2648](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages that are created when a file transfer occurs that contains calls to user exits.

[“Connect:Direct bridge transfer log message examples” on page 2650](#)

The `destinationAgent` or `sourceAgent` element contains additional attributes when the destination agent or source agent is a Connect:Direct bridge agent. The Started log message contains only a subset of the information about the Connect:Direct transfer. The Progress and Completed log messages contain full information about the Connect:Direct transfer.

## Single transfer log message examples

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages for a single file transfer being started, in progress, and completed.

## Single file transfer - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d223d0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:03:26.484Z">started</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
  <originator>
    <hostName>dhcp-9-20-240-199.hursley.ibm.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <transferSet startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="0">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">dhcp-9-20-240-199.hursley.ibm.com.</
metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d223d0020</
metaData>
      <metaData key="com.ibm.wmqfte.ScheduleId">3</metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
</transaction>
```



```
<scheduleLog ID="3"/>
</transaction>
```

## Single file transfer success - progress

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d223d0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:03:26.615Z">progress</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </destinationAgent>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <transferSet index="0" size="1" startTime="2011-01-26T13:03:26.484Z" total="1"
  bytesSent="1198">
    <item mode="binary">
      <source disposition="leave" type="file">
        <file size="1151" last-modified="2009-11-02T10:37:01.000Z">/etc/passwd</file>
        <checksum method="MD5">2287181c07199f879de28296371cb24c</checksum>
      </source>
      <destination type="file">
        <file size="1151" last-modified="2011-01-26T13:03:26.000Z">/tmp/passwd</file>
        <checksum method="MD5">2287181c07199f879de28296371cb24c</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
  </transferSet>
</transaction>
```

## Single file transfer success - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d223d0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:03:26.622Z">completed</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </destinationAgent>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <status resultCode="0">
    <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
  </status>
  <transferSet startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="1198">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d223d0020</
metaData>
      <metaData key="com.ibm.wmqfte.ScheduleId">3</metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
```

```

</transferSet>
<statistics>
  <actualStartTime>2011-01-26T13:03:26.541Z</actualStartTime>
  <retryCount>0</retryCount>
  <numFileFailures>0</numFileFailures>
  <numFileWarnings>0</numFileWarnings>
</statistics>
</transaction>

```

### Related reference

[“Triggered file transfer log message example” on page 2646](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML example shows the log message that is created when a file transfer containing a trigger condition is started.

[“MFT user exit log message examples” on page 2648](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages that are created when a file transfer occurs that contains calls to user exits.

[“File transfer log message formats” on page 2631](#)

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. These messages conform to the schema `TransferLog.xsd`, which is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your Managed File Transfer installation.

### Multiple file transfer log message examples

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id` when a transfer that contains multiple files occurs.

### Multiple file transfer - started

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d035c0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:12:58.534Z">started</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
  <originator>
    <hostName>example.com</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <transferSet startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="0">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">example.com</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d035c0020</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
</transaction>

```

### Multiple file transfer - progress

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d035c0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">

```

```

<action time="2011-01-26T13:12:58.753Z">progress</action>
<sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
  <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
</sourceAgent>
<destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
  <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
</destinationAgent>
<originator>
  <hostName>example.com.</hostName>
  <userID>mqm</userID>
  <mqmdUserID>mqm</mqmdUserID>
</originator>
<transferSet index="0" size="6" startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="440">
  <item mode="binary">
    <source disposition="leave" type="file">
      <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file01.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </source>
    <destination type="file">
      <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file01.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </destination>
    <status resultCode="0"/>
  </item>
  <item mode="binary">
    <source disposition="leave" type="file">
      <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file02.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </source>
    <destination type="file">
      <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file02.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </destination>
    <status resultCode="0"/>
  </item>
  <item mode="binary">
    <source disposition="leave" type="file">
      <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file03.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </source>
    <destination type="file">
      <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file03.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </destination>
    <status resultCode="0"/>
  </item>
  <item mode="binary">
    <source disposition="leave" type="file">
      <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file04.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </source>
    <destination type="file">
      <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file04.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </destination>
    <status resultCode="0"/>
  </item>
  <item mode="binary">
    <source disposition="leave" type="file">
      <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file05.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </source>
    <destination type="file">
      <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file05.txt</
file>
      <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
    </destination>
    <status resultCode="0"/>
  </item>
  <item mode="binary">
    <source disposition="leave" type="file">
      <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file06.txt</
file>

```

```

        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </source>
      <destination type="file">
        <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file06.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
  </transferSet>
</transaction>

```

## Multiple file transfer - completed

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d035c0020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:12:58.766Z">completed</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </destinationAgent>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <status resultCode="0">
    <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
  </status>
  <transferSet startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="440">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d035c0020</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
  <statistics>
    <actualStartTime>2011-01-26T13:12:58.634Z</actualStartTime>
    <retryCount>0</retryCount>
    <numFileFailures>0</numFileFailures>
    <numFileWarnings>0</numFileWarnings>
  </statistics>
</transaction>

```

## Failed file transfer log message examples

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of Log/agent\_name/transfer\_id. The XML examples show the log messages for a file transfer that fails being started, in progress, and completed.

## File transfer failure - started

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d03620020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:19:15.767Z">started</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>

```

```

<originator>
  <hostName>example.com.</hostName>
  <userID>mqm</userID>
  <mqmdUserID>mqm</mqmdUserID>
</originator>
<transferSet startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
  <metaDataSet>
    <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
    <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
    <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
    <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
    <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d03620020</
metaData>
    <metaData key="com.ibm.wmqfte.Priority">0</metaData>
  </metaDataSet>
</transferSet>
</transaction>

```

## File transfer failure - progress

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d03620020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:19:15.944Z">progress</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </destinationAgent>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <transferSet index="0" size="1" startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
    <item mode="binary">
      <source disposition="leave" type="file">
        <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file01.txt</
file>
        <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
      </source>
      <destination type="file">
        <file>/srv/nfs/outgoing/file01.txt</file>
      </destination>
      <status resultCode="1">
        <supplement>BFGIO0006E: File "/srv/nfs/outgoing/file01.txt" already exists.</
supplement>
      </status>
    </item>
  </transferSet>
</transaction>

```

## File transfer failure - completed

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  ID="414d51205553322e42494e44494e47538b0f404d03620020"
  agentRole="sourceAgent"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-01-26T13:19:15.948Z">completed</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </sourceAgent>
  <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
    <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
  </destinationAgent>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>

```

```

    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <status resultCode="40">
    <supplement>BFGRP0034I: The file transfer request has
      completed with no files being transferred.
    </supplement>
  </status>
  <transferSet startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d5120555322e42494e44494e47538b0f404d03620020</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
  <statistics>
    <actualStartTime>2011-01-26T13:19:15.878Z</actualStartTime>
    <retryCount>0</retryCount>
    <numFileFailures>1</numFileFailures>
    <numFileWarnings>0</numFileWarnings>
  </statistics>
</transaction>

```

### ***Triggered file transfer log message example***

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML example shows the log message that is created when a file transfer containing a trigger condition is started.

### **Trigger single file transfer success - started**

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
  ID="414d5120514d312020202020202020207e970d492000a102" agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2008-11-02T22:05:18.703Z">started</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1" />
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1" />
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1 </mqmdUserID>
  </originator>
  <trigger log="yes">
    <fileExist comparison="=" value="exist">c:\trigger.txt</fileExist>
  </trigger>
  <transferSet startTime="2008-11-02T22:05:18.703Z" total="1"></transferSet>
</transaction>

```

#### **Related reference**

[“Single transfer log message examples” on page 2640](#)

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages for a single file transfer being started, in progress, and completed.

[“MFT user exit log message examples” on page 2648](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages that are created when a file transfer occurs that contains calls to user exits.

[“File transfer log message formats” on page 2631](#)



## MFT user exit log message examples

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages that are created when a file transfer occurs that contains calls to user exits.

### Exit single file transfer proceed - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
  ID="414d5120514d3120202020202020202020202020207e970d492000d502" agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2008-11-02T22:36:13.046Z">started</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1" />
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1" />
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1 </mqmdUserID>
  </originator>
  <transferSet startTime="2008-11-02T22:36:13.046Z" total="1">
    <metaDataSet>
      <metaData key="testkey1">testvalue1</metaData>
      <metaData key="testkey2">testvalue2</metaData>
    </metaDataSet>
  </transferSet>
</transaction>
```

### Exit single file transfer proceed - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
  ID="414d5120514d3120202020202020202020202020207e970d492000d502"
  agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2008-11-02T22:36:13.546Z">completed</action>
  <sourceAgent agent="FTEAGENT" QMgr="QM1">
    <startExits>
      <exit name="class testExits.SourceExit1">
        <status resultCode="proceed">
          <supplement>Source Start, modified metadata</supplement>
        </status>
      </exit>
    </startExits>
    <endExits>
      <exit name="class testExits.SourceExit1">
        <status>
          <supplement>Source End</supplement>
        </status>
      </exit>
    </endExits>
    <systemInfo architecture="x86" name="Windows 7"
      version="6.1 build 7601 Service Pack 1" />
  </sourceAgent>
  <destinationAgent agent="FTEAGENT" QMgr="QM1">
    <startExits>
      <exit name="class testExits.DestinationExitProceed">
        <status resultCode="proceed">
          <supplement>Destination start, with proceed</supplement>
        </status>
      </exit>
    </startExits>
    <endExits>
      <exit name="class testExits.DestinationExitProceed">
        <status>
          <supplement>destination end</supplement>
        </status>
      </exit>
    </endExits>
  </destinationAgent>
</transaction>
```





```

        </exit>
    </endExits>
    <systemInfo architecture="x86" name="Windows 7"
        version="6.1 build 7601 Service Pack 1" />
</destinationAgent>
<originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
    <mqmdUserID>USER1      </mqmdUserID>
</originator>
<transferSet startTime="2008-11-02T22:25:59.078Z" total="1" />
</transaction>

```

## Related reference

[“Single transfer log message examples” on page 2640](#)

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML examples show the log messages for a single file transfer being started, in progress, and completed.

[“Triggered file transfer log message example” on page 2646](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. The XML example shows the log message that is created when a file transfer containing a trigger condition is started.

[“File transfer log message formats” on page 2631](#)

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of `Log/agent_name/transfer_id`. These messages conform to the schema `TransferLog.xsd`, which is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your Managed File Transfer installation.

## Connect:Direct bridge transfer log message examples

The `destinationAgent` or `sourceAgent` element contains additional attributes when the destination agent or source agent is a Connect:Direct bridge agent. The Started log message contains only a subset of the information about the Connect:Direct transfer. The Progress and Completed log messages contain full information about the Connect:Direct transfer.

## Source agent is Connect:Direct bridge agent

### Started:

```

<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    ID="414d5120514d5f696b6b796f20202020a704654d20092507"
    agentRole="sourceAgent"
    version="4.00"
    xsi:noNamespaceSchemaLocation="TransferLog.xsd"
    xmlns="">
    <action time="2011-03-07T13:05:01.838Z">started</action>
    <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE" bridgeNode="CDNODE_VARUNA">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_KUIPER" agent="IXION"/>
    <originator>
        <hostName>kuiper.example.com.</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <transferSet bytesSent="0" startTime="2011-03-07T13:05:01.838Z" total="1">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">VARUNA</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">IXION</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">kuiper.example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d20092507</
metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
</transaction>

```

## Progress:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d20092507"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T13:05:03.448Z">progress</action>
  <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE"
    bridgeNode="CDNODE_VARUNA" pNode="CDNODE_VARUNA" snode="CDNODE_ERIS">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_KUIPER" agent="IXION" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>kuiper.example.com.</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <transferSet bytesSent="48" index="0" size="1" startTime="2011-03-07T13:05:01.838Z" total="1">
    <item mode="binary">
      <source disposition="leave" processName="f2007567" processNumber="68" type="file">
        <file last-modified="2011-03-07T13:05:02.573Z" size="4">CDNODE_ERIS:D:/AGENTS/
CDNODE_ERIS/test.txt</file>
        <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
      </source>
      <destination type="file">
        <file last-modified="2011-03-07T13:05:03.338Z" size="4">D:\AGENTS\IXION\test.txt</file>
        <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
  </transferSet>
</transaction>
```

## Completed:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d20092507"
  agentRole="sourceAgent"
  version="4.00" xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T13:05:03.495Z">completed</action>
  <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE"
    bridgeNode="CDNODE_VARUNA" pNode="CDNODE_VARUNA" snode="CDNODE_ERIS">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_KUIPER" agent="IXION" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>kuiper.example.com.</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <status resultCode="0">
    <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
  </status>
  <transferSet bytesSent="48" startTime="2011-03-07T13:05:01.838Z" total="1">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">VARUNA</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">IXION</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">kuiper.example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d20092507</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
  <statistics>
    <actualStartTime>2011-03-07T13:05:02.041Z</actualStartTime>
    <retryCount>0</retryCount>
    <numFileFailures>0</numFileFailures>
    <numFileWarnings>0</numFileWarnings>
```

```
</statistics>
</transaction>
```

## Destination agent is Connect:Direct bridge agent Started:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T10:29:44.854Z">started</action>
  <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_ASTEROID" agent="VESTA"/>
  <originator>
    <hostName>belt.example.com.</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <transferSet bytesSent="0" startTime="2011-03-07T10:29:44.854Z" total="1">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">PALLAS</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">VESTA</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">belt.example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d2008e102</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
</transaction>
```

## Progress:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T10:29:46.682Z">progress</action>
  <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_ASTEROID" agent="VESTA" agentType="CD_BRIDGE"
    bridgeNode="CDNODE_VESTA" pNode="CDNODE_VESTA" snode="CDNODE_HYGIEA">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>belt.example.com.</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <transferSet bytesSent="48" index="0" size="1" startTime="2011-03-07T10:29:44.854Z" total="1">
    <item mode="binary">
      <source disposition="leave" type="file">
        <file last-modified="2011-03-04T14:53:28.323Z" size="4">D:\AGENTS\PALLAS\test.txt</
file>
        <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
      </source>
      <destination processName="f2006965" processNumber="59" type="file">
        <file size="4">CDNODE_VESTA:D:/AGENTS/CDNODE_VESTA/test.txt</file>
        <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
      </destination>
      <status resultCode="0"/>
    </item>
  </transferSet>
</transaction>
```

## Completed:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
  agentRole="sourceAgent"
  version="4.00"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2011-03-07T10:29:46.698Z">completed</action>
  <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </sourceAgent>
  <destinationAgent QMgr="QM_ASTEROID" agent="VESTA" agentType="CD_BRIDGE"
    bridgeNode="CDNODE_VESTA" pnode="CDNODE_VESTA" snode="CDNODE_HYGIEA">
    <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
  </destinationAgent>
  <originator>
    <hostName>belt.example.com</hostName>
    <userID>sol</userID>
    <mqmdUserID>sol</mqmdUserID>
  </originator>
  <status resultCode="0">
    <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
  </status>
  <transferSet bytesSent="48" startTime="2011-03-07T10:29:44.854Z" total="1">
    <metaDataSet>
      <metaData key="com.ibm.wmqfte.SourceAgent">PALLAS</metaData>
      <metaData key="com.ibm.wmqfte.DestinationAgent">VESTA</metaData>
      <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
      <metaData key="com.ibm.wmqfte.OriginatingHost">belt.example.com.</metaData>
      <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d2008e102</
metaData>
      <metaData key="com.ibm.wmqfte.Priority">0</metaData>
    </metaDataSet>
  </transferSet>
  <statistics>
    <actualStartTime>2011-03-07T10:29:45.010Z</actualStartTime>
    <retryCount>0</retryCount>
    <numFileFailures>0</numFileFailures>
    <numFileWarnings>0</numFileWarnings>
  </statistics>
</transaction>
```

## Scheduled file transfer log message formats

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its `SYSTEM.FTE/Log/agent name/schedule ID` topic). This message conforms to the `ScheduleLog.xsd` XML schema.

## Schema

The following schema describes which elements are valid in a schedule log XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>

  <xsd:element name="schedulelog">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="originator" type="hostUserIDType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="action" type="actionType"
          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="schedule" type="scheduleType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceAgent" type="agentType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="destinationAgent" type="agentClientType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="status" type="statusType"
          maxOccurs="1" minOccurs="0"/>
        <xsd:element name="transferSet" type="transferSetType"
          maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

        <xsd:element name="job"
                    type="jobType"
                    maxOccurs="1"
                    minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required" />
    <xsd:attribute name="ID" type="xsd:string" use="required" />
</xsd:complexType>
</xsd:element>

<xsd:complexType name="actionType">
  <xsd:simpleContent>
    <xsd:extension base="actionEnumType">
      <xsd:attribute name="time" type="xsd:dateTime" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="actionEnumType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="submit" />
    <xsd:enumeration value="delete" />
    <xsd:enumeration value="expire" />
    <xsd:enumeration value="skipped" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="transferSetType">
  <xsd:sequence>
    <xsd:element name="item"
                  type="itemType"
                  maxOccurs="unbounded"
                  minOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="size" type="xsd:int" use="required" />
  <xsd:attribute name="priority" type="priorityType" use="optional" />
</xsd:complexType>

<xsd:complexType name="itemType">
  <xsd:sequence>
    <xsd:element name="source"
                  type="fileSourceType"
                  maxOccurs="1"
                  minOccurs="1" />
    <xsd:element name="destination"
                  type="fileDestinationType"
                  maxOccurs="1"
                  minOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="mode" type="modeType" use="required" />
  <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required" />
</xsd:complexType>

</xsd:schema>

```

## Understanding the schedule log message

The elements and attributes used in the schedule log message are described:

### <schedulelog>

Group element that describes a single submitted scheduled file transfer.

Attribute	Description
version	Specifies the version of this element as detailed by Managed File Transfer.
ID	The unique identifier for the submitted schedule file transfer.

### <originator>

Group element that contains the elements specifying the originator of the request.

### <hostName>

The host name of the system where the source file is located.

### <userID>

The user ID that originated the file transfer.

### <mqmdUserID>

The MQ user ID that was supplied in the message descriptor (MQMD)

**<action>**

Specifies the action to take with the scheduled transfer matching the ID attribute of <schedulelog> element. This element must be one of the following values:

- submit - new scheduled transfer
- delete - cancel schedule transfer
- expire - schedule transfer entry about to be processed
- skipped - a transfer that was scheduled cannot be started because the agent is offline. This message is logged when the agent becomes available to indicate the transfer was skipped.

Attribute	Description
time	Specifies the date and time the log entry was published (in date time format).

**<sourceAgent>**

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

**<destinationAgent>**

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

**<status>**

The result code and supplement messages.

**<transferSet>**

Specifies a group of file transfers you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
size	Specifies the number of transfer items.
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

**<item>**

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
mode	Specifies the transfer mode as being either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. Permitted values are MD5 or none



**<source>**

Group element that contains the <file> and <checksum> elements for the file on the source system.

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid options are as follows: <ul style="list-style-type: none"> <li>• leave - the source files are left unchanged.</li> <li>• delete - the source files are deleted from the source system after the source file is successfully transferred.</li> </ul>

#### <destination>

Group element that contains the <file> and <checksum> elements for the file on the destination system.

Attribute	Description
type	The type of file or directory at the destination. The valid options are as follows: <ul style="list-style-type: none"> <li>• file - specifies a file as the destination</li> <li>• directory - specifies a directory as the destination</li> <li>•  dataset - specifies a z/OS data set as the destination</li> <li>•  PDS - specifies a z/OS partitioned data set as the destination</li> </ul>
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows: <ul style="list-style-type: none"> <li>• error - reports an error and the file is not transferred.</li> <li>• overwrite - overwrites the existing destination file.</li> </ul>

#### <file>

Specifies the name of the file to transfer. Use the fully qualified path in the format consistent with your operating system, for example C : / from / here . txt. Do not use file URIs.

Attribute	Description
encoding	The encoding for a text file transfer.
EOL	Specifies the end of line marker. Permitted values are: <ul style="list-style-type: none"> <li>• LF - line feed character only</li> <li>• CRLF - carriage return and line feed character sequence</li> </ul>

#### <job>

Group element that contains an element specifying job details. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that is included in the transfer request message, which is described in the following topic: [“File transfer request message format” on page 2665.](#)

#### <name>

The value of name can be any string.

#### Examples

Examples of XML messages that conform to this schema are provided for each of the following scheduled transfer actions:

- [A scheduled transfer is created](#)



- [A scheduled transfer is canceled](#)
- [A schedule transfer expires](#)

Transfers that are started by a schedule are logged in the same way as a standard transfer. For examples of log messages for transfers started by a schedule, see [“Scheduled file transfer log message examples”](#) on page 2647.

### Related reference

[“MFT agent status message format”](#) on page 2615

When a Managed File Transfer Agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the SYSTEM.FTE/Agents/*agent name* topic).

[“File transfer request message format”](#) on page 2665

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the FileTransfer.xsd schema and have the <request> element as the root element. The FileTransfer.xsd schema document is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

[“File transfer status message format”](#) on page 2627

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its SYSTEM.FTE/Transfers/*agent\_name/transfer ID* topic), which conforms to the TransferStatus.xsd XML schema. The TransferStatus.xsd file is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory of your WMQMFT installation.

[“File transfer log message formats”](#) on page 2631

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent\_name/transfer\_id*. These messages conform to the schema TransferLog.xsd, which is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory of your Managed File Transfer installation.

[“MFT monitor request message formats”](#) on page 2682

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the **fteCreateMonitor** command or using the IBM MQ Explorer interface.

[“MFT message formats for security”](#) on page 2695

This topic describes the messages published to the Managed File Transfer coordination queue manager relevant to security.

### **Schedule file transfer log message examples**

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/*agent\_name/schedule\_id* when a scheduled transfer action occurs.

### **Scheduled transfer log message**

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent\_name/schedule ID* topic). This message conforms to the ScheduleLog.xsd XML schema. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulelog version="1.00" ID="5"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
  </originator>
  <action time="2008-11-23T21:32:01Z">submit</action>
  <schedule>
    <submit timebase="admin" timezone="Europe/London">2008-11-23T22:00</submit>
  </schedule>
</schedulelog>
```

```

<sourceAgent agent="FTEAGENT" QMgr="QM1" />
<destinationAgent agent="FTEAGENT" QMgr="QM1" />
<status resultCode="0" />
<transferSet size="1" priority="0">
  <item mode="binary" checksumMethod="MD5">
    <source recursive="false" disposition="leave">
      <file>c:\sourcefiles\source1.doc</file>
    </source>
    <destination type="file" exist="overwrite">
      <file>c:\destinationfiles\dest1.doc</file>
    </destination>
  </item>
</transferSet>
</schedulelog>

```

This message is a log of the following information:

- Who originated the request
- When the request was submitted
- When the scheduled transfer starts
- The source and destination agent details
- The transfer specification

The ID attribute of the <schedulelog> element is a unique ID for this scheduled transfer (in the source agent). This ID is used to correlate schedule entries with the actual file transfers.

The <action> element value of submit confirms the request has been received.

## Scheduled transfer cancel log message

When a request to cancel a pending scheduled file transfer is received by the agent, the following message is published to the SYSTEM.FTE/Log/agent\_name topic:

```

<?xml version="1.0" encoding="UTF-8"?>
<schedulelog version="1.00" ID="5"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
  <originator>
    <hostName>reportserver.com</hostName>
    <userID>USER1</userID>
  </originator>
  <action time="2008-11-23T21:56:27Z">delete</action>
  <status resultCode="0" />
</schedulelog>

```

The ID attribute value corresponds to the ID of the pending transfer request ID in the schedules message.

## Scheduled transfer expire log message

When the current time matches the time of the earliest pending file transfer in the schedule list (as indicated by the value of the <next> element), a schedule log message is published to indicate that the scheduled transfer entry has expired:

```

<?xml version="1.0" encoding="UTF-8"?>
<schedulelog xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00" ID="3"
  xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <action time="2011-01-26T13:03:26Z">expire</action>
  <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER" />
  <status resultCode="0" />
</schedulelog>

```

The <action> element value of "expire" confirms the schedule entry has now been removed from the schedule list and is being processed. A schedule message for the agent is published with the expired entry no longer present.

### Related reference

[“Scheduled file transfer log message formats” on page 2653](#)

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/agent\_name/schedule ID topic). This message conforms to the ScheduleLog.xsd XML schema.

[“Scheduled file transfer log message examples” on page 2647](#)

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/agent\_name/transfer\_id. The XML examples show the log messages that are created when a file transfer occurs as a result of a schedule.

## MFT monitor log message format

Monitor log messages are published to the SYSTEM.FTE topic with a topic string of Log/agent\_name/ Monitors/monitor\_name/monitor\_id.

If you want to collect data or view monitor actions, set up a subscription to a wildcard topic tailored to the monitors that you are interested in. For example:

```
Log/#
```

or,

```
Log/agent_name/#
```

This subscription can be durable or non-durable. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. Non-durable subscriptions exist only as long as a subscribing application's connection to the queue manager remains open.

The MonitorLog.xsd schema document is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory. The MonitorLog.xsd schema imports fteutils.xsd, which is in the same directory.

## Schema

The following schema describes which elements are valid in a monitor log XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <xsd:element name="monitorLog">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="originator" type="hostUserIDType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="references" type="referencesType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="action" type="monitorActionType" maxOccurs="1"
minOccurs="1"/>
        <xsd:element name="monitorAgent" type="agentType" maxOccurs="1"
minOccurs="1"/>
        <xsd:element name="status" type="statusType" maxOccurs="1"
minOccurs="1"/>
        <xsd:element name="monitorMetaData" type="monitorMetaDataType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="monitorExits" type="exitGroupType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="jobDetails" type="jobType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="taskXMLRequest" type="taskXMLRequestType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="monitorXMLRequest" type="monitorXMLRequestType"
minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="version" type="versionType" use="required"/>
  <xsd:attribute name="monitorName" type="xsd:string" use="required"/>
  <xsd:attribute name="referenceId" type="xsd:string" use="optional"/>
</xsd:complexType>
</xsd:element>

<xsd:complexType name="monitorActionType">
  <xsd:simpleContent>
    <xsd:extension base="monitorActionEnumType">
      <xsd:attribute name="time" type="xsd:dateTime" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="monitorActionEnumType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="create" />
    <xsd:enumeration value="delete" />
    <xsd:enumeration value="start" />
    <xsd:enumeration value="stop" />
    <xsd:enumeration value="triggerSatisfied" />
    <xsd:enumeration value="triggerNotSatisfied" />
    <xsd:enumeration value="triggerFail" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="monitorMetaDataType">
  <xsd:sequence>
    <xsd:element name="originalMetaData" type="metaDataSetType" maxOccurs="1" minOccurs="0"/>
    <xsd:element name="updatedMetaData" type="metaDataSetType" maxOccurs="unbounded"
minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="taskXMLRequestType">
  <xsd:sequence>
    <xsd:element name="originalRequest" type="xsd:string" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="updatedRequest" type="xsd:string" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="taskId" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="referencesType">
  <xsd:sequence>
    <xsd:element name="createRequest" type="xsd:string" maxOccurs="1" minOccurs="0"/>
    <xsd:element name="taskRequest" type="xsd:string" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="monitorXMLRequestType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="type" type="xmlContentEnumType" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="xmlContentEnumType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="escapedXML" />
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

## Understanding the monitor log message

The elements and attributes used in the monitor log messages are described in the following list:

### <monitorLog>

Group element containing the elements describe an action that has been performed by a monitor.

Attribute	Description
version	Required. The version of the monitor list message format.

Attribute	Description
monitorName	Required. The name of the monitor. Unique for the agent that the monitor is defined on.
referenceId	The ID of the monitor action.

**<originator>**

Group element that contains the elements specifying the originator of the request.

**<hostName>**

The host name of the system where the source file is located.

**<userID>**

The user ID that originated the file transfer.

**<mqmdUserID>**

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

**<references>**

References to the IDs of other messages associated with this monitor action.

**<createRequest>**

The message ID of the XML request message that was used to create the monitor.

**<taskRequest>**

The message ID of the XML request message that the monitor submits as a result of this action.

**<action>**

The action that occurred, which this log message is associated with. The value inside the element can be one of the following: create, delete, start, stop, triggerSatisfied, triggerNotSatisfied, or triggerFail.

**<monitorAgent>**

The agent that is monitoring the resource.

Attribute	Description
agent	Required. The name of the agent.
QMgr	Optional. The name of the queue manager that the agent connects to.
bridgeURL	Optional. If the agent is a protocol bridge agent, the URL of the protocol server.

**<status>**

The status of the resource monitor action being logged.

Attribute	Description
resultCode	Required. The integer result code from the action.

**<supplement>**

Additional information about the status of the resource monitor action being logged.

**<monitorMetaData>**

Group element that contains the <originalMetaData> and <updatedMetaData> elements.

**<originalMetaData>**

Element that contains one or more <metadata> elements that describe the metadata of the monitor before the action occurs.

**<updatedMetaData>**

Element that contains one or more <metadata> elements that describe the metadata of the monitor after the action occurs.

**<metadata>**

Defines a metadata key-value pair. The key is an attribute of the element; the value is the content of the element.

Attribute	Description
key	The key of the metadata.

**<monitorExits>**

Group element containing one or more <exit> elements.

**<exits>**

Element describing an exit run by the resource monitor.

Attribute	Description
name	Required. The name of the resource monitor exit.

**<status>**

The status of the resource monitor exit that is being logged.

Attribute	Description
resultCode	Required. The integer result code from the exit.

**<supplement>**

Additional information about the status of the resource monitor exit that is being logged.

**<jobDetails>**

Element containing a single <name> element.

**<name>**

The name of the job..

**<taskXMLRequest>**

Group element that contains the <originalRequest> and <updatedRequest> elements.

Attribute	Description
taskId	The ID of the task request message.

**<originalRequest>**

Element that contains the escaped XML request message for the task that the monitor performs.

**<updatedRequest>**

Element that contains the updated escaped XML request message for the task that the monitor performs.

**<monitorXMLRequest>**

The monitor XML request.

Attribute	Description
type	Required. The format of the monitor XML request data inside of the <monitorXMLRequest> element. The only valid value is escapedXML.

**Examples**

Examples of XML messages that conform to this schema are provided for each of the following monitor actions:

- [A monitor is created](#)
- [The condition of a monitor is satisfied when the monitor polls the resource](#)
- [The condition of a monitor is not satisfied when the monitor polls the resource](#)
- [A monitor is deleted](#)

## Related reference

[“MFT monitor log message examples” on page 2663](#)

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/agent\_name/monitor\_id when a monitor action occurs.

## MFT monitor log message examples

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/agent\_name/monitor\_id when a monitor action occurs.

### Monitor created log message

```
<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  monitorName="MONITORTWO"
  referenceId="414d51205553322e42494e44494e47538b0f404d04410020"
  xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <references>
    <createRequest>414d51205553322e42494e44494e47538b0f404d04410020</createRequest>
  </references>
  <action time="2011-01-26T12:41:24Z">start</action>
  <monitorAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
  <status resultCode="0"/>
</monitorLog>
```

### Monitor condition satisfied log message

```
<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  monitorName="MONITORONE"
  referenceId="414d51205553322e42494e44494e47538b0f404d09430020"
  xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <references>
    <createRequest>414d51205553322e42494e44494e47538b0f404d09430020</createRequest>
  </references>
  <action time="2011-01-26T12:56:46Z">triggerSatisfied</action>
  <monitorAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <status resultCode="0"/>
  <monitorMetaData>
    <originalMetaData>
      <metaData key="AGENTNAME">AGENT_JUPITER</metaData>
      <metaData key="LASTMODIFIEDDATEUTC">2011-01-26</metaData>
      <metaData key="CURRENTTIMESTAMPUTC">20110126125646793</metaData>
      <metaData key="CURRENTTIMESTAMP">20110126125646793</metaData>
      <metaData key="LASTMODIFIEDDATE">2011-01-26</metaData>
      <metaData key="FILENAME">new.completed</metaData>
      <metaData key="LASTMODIFIEDTIMEUTC">12.56</metaData>
      <metaData key="LASTMODIFIEDTIME">12.56</metaData>
      <metaData key="FILESIZE">0</metaData>
      <metaData key="FILEPATH">/srv/nfs/incoming/new.completed</metaData>
    </originalMetaData>
    <updatedMetaData>
      <metaData key="AGENTNAME">AGENT_JUPITER</metaData>
      <metaData key="LASTMODIFIEDDATEUTC">2011-01-26</metaData>
      <metaData key="CURRENTTIMESTAMPUTC">20110126125646793</metaData>
      <metaData key="CURRENTTIMESTAMP">20110126125646793</metaData>
      <metaData key="LASTMODIFIEDDATE">2011-01-26</metaData>
      <metaData key="FILENAME">new.completed</metaData>
      <metaData key="LASTMODIFIEDTIMEUTC">12.56</metaData>
      <metaData key="LASTMODIFIEDTIME">12.56</metaData>
      <metaData key="FILESIZE">0</metaData>
      <metaData key="FILEPATH">/srv/nfs/incoming/new.completed</metaData>
    </updatedMetaData>
  </monitorMetaData>
</monitorLog>
```

```

</monitorMetaData>
<taskXMLRequest taskId="null">
  <originalRequest>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="4.00"
    xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
      &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
      &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
      &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
      &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
      &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
        &lt;source disposition="leave" recursive="false"&gt;
          &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
          &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
        &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
      </originalRequest>
      <updatedRequest>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
          &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
          &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
          &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
          &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
          &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
            &lt;source disposition="leave" recursive="false"&gt;
              &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;
              &lt;file&gt;/srv/backup&lt;/file&gt;&lt;destination exist="error" type="directory"&gt;
                &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
              &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
            </updatedRequest>
          </taskXMLRequest>
        </monitorLog>

```

### Monitor condition not satisfied log message

```

<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  monitorName="MONITORONE"
  referenceId="414d51205553322e42494e44494e47538b0f404d09430020"
  xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
    <mqmdUserID>mqm</mqmdUserID>
  </originator>
  <references>
    <createRequest>414d51205553322e42494e44494e47538b0f404d09430020</createRequest>
  </references>
  <action time="2011-01-26T12:58:46Z">triggerNotSatisfied</action>
  <monitorAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
  <status resultCode="0"/>
</monitorLog>

```

### Monitor deleted log message

```

<?xml version="1.0" encoding="UTF-8"?>
<lst:monitorList xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:lst="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  version="4.00"
  agent="AGENT_JUPITER"
  monitor="MONITORONE"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition
MonitorList.xsd">
  <status state="deleted"/>
  <configuration>
    <description/>
    <resources>
      <directory recursionLevel="0" id="">/srv/nfs/incoming</directory>
    </resources>
    <triggerMatch>
      <conditions>
        <condition>
          <name/>
          <resource id="">
            <fileMatch>
              <pattern>*.completed</pattern>

```



```

        </fileMatch>
      </condition>
    </conditions>
  </triggerMatch>
  <tasks>
    <task>
      <name/>
      <description/>
      <taskXML>&lt;&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;&lt;request
        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;&lt;managedTransfer&gt;
          &lt;&lt;originator&gt;&lt;&lt;hostName&gt;example.ibm.com.&lt;&lt;/hostName&gt;
          &lt;&lt;userID&gt;mqm&lt;&lt;/userID&gt;&lt;&lt;/originator&gt;
          &lt;&lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
          &lt;&lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
          &lt;&lt;transferSet&gt;&lt;&lt;item checksumMethod="MD5" mode="binary"&gt;
          &lt;&lt;source disposition="leave" recursive="false"&gt;
          &lt;&lt;file&gt;/srv/nfs/incoming/*.txt&lt;&lt;/file&gt;&lt;&lt;/source&gt;
          &lt;&lt;destination exist="error" type="directory"&gt;
          &lt;&lt;file&gt;/srv/backup&lt;&lt;/file&gt;&lt;&lt;/destination&gt;
          &lt;&lt;/item&gt;&lt;&lt;/transferSet&gt;&lt;&lt;/managedTransfer&gt;&lt;&lt;/request&gt;
        </taskXML>
      </task>
    </tasks>
  </configuration>
  <pollInterval units="minutes">1</pollInterval>
  <batch maxSize="1"/>
</lst:monitorList>

```

## File transfer request message format

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

File transfer messages can have one of following three root elements:

- `<request>` - for new file transfer requests, managed call requests, or deleting scheduled transfers that are pending
- `<cancel>` - for canceling file transfers in progress
- `<transferSpecifications>` - for specifying multiple transfer file groups, used by the **fteCreateTransfer** command

For information about specifying multiple transfer groups by using the **transferSpecifications** element, see [Using transfer definition files](#).

## Schema

The following schema describes which elements are valid in a transfer request XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <!--
    Defines the request of a managed transfer and version number
  -->
  <request version="1.00" ...
    <managedTransfer>
      ...
    </managedTransfer>
  </request>
  <xsd:element name="request">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element name="managedTransfer" type="managedTransferType"/>
        <xsd:element name="deleteScheduledTransfer" type="deleteScheduledTransferType" />
        <xsd:element name="managedCall" type="managedCallType"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>

```

```

        <xsd:attribute name="version" type="versionType" use="required" />
    </xsd:complexType>
</xsd:element>

<!--
    Defines the cancel request of a managed transfer and version number
    <cancel version="1.00"
        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
        <originator>
            <hostName>myMachine</hostName>
            <userID>myUserId</userID>
        </originator>      - Delete a scheduled transfer.

        <transfer>
            Transfer ID to Cancel
        </transfer>
    </cancel>
-->
<xsd:element name="cancel">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="originator" type="hostUserIDType" maxOccurs="1" minOccurs="1" />
            <xsd:choice>
                <xsd:element name="transfer" type="IDType" maxOccurs="1" minOccurs="1" />
                <xsd:element name="call" type="IDType" maxOccurs="1" minOccurs="1" />
            </xsd:choice>
            <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="version" type="versionType" use="required" />
    </xsd:complexType>
</xsd:element>

<!--
    Defines the transfer definition element structure.
    <transferSpecifications>
        <item ...
        <item ...
    </transferSpecifications>
-->
<xsd:element name="transferSpecifications">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="item" type="itemType" minOccurs="1" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!--
    Define a managed transfer of an instigator and request
    <managedTransfer>

        <originator>
            .
            .
            .
        </originator>

        <schedule>
            <submit timebase="source"|"UTC">2008-12-07T16:07</submit>
            <repeat>
                <frequency interval="hours">2</frequency>
                <expireTime>2008-12-0816:07</expireTime>
            </repeat>
        </schedule>

        <sourceAgent agent="here" QMgr="near" />
        <destinationAgent agent="there" QMgr="far" />

        <trigger>
            .
            .
            .
        </trigger>

        <transferSet>
            .
            .
            .
        </transferSet>
    </managedTransfer>
-->

    <xsd:complexType name="managedTransferType">
        <xsd:sequence>
            <xsd:element name="originator" type="origTransferRequestType" maxOccurs="1"
minOccurs="1"/>
            <xsd:element name="schedule" type="scheduleType" maxOccurs="1" minOccurs="0"/>

```

```

        <xsd:element name="sourceAgent" type="agentType" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="destinationAgent" type="agentClientType" maxOccurs="1" minOccurs="1" />
        <xsd:element name="trigger" type="triggerType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="transferSet" type="transferSetType" maxOccurs="1" minOccurs="1" />
        <xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    This is a modified form of origRequestType which is used on a managed transfer request.
    The hostName and userID are mandatory attributes in this case.
-->
<xsd:complexType name="origTransferRequestType">
    <xsd:sequence>
        <xsd:element name="hostName" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="userID" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="mqmdUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="webBrowser" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="webUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    Defines the transfer set of source and destination agent and one or more files
    <transferSet priority="1">
        <metaDataSet>
            <metaData key="keyname">keyvalue</metaData>
            <metaData key="keyname">keyvalue</metaData>
        </metaDataSet>

        <item>
            ...
        </item>
    </transferSet>
-->
<xsd:complexType name="transferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet" type="metaDataSetType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="preSourceCall" type="commandActionType" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="postSourceCall" type="commandActionType" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="preDestinationCall" type="commandActionType" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="postDestinationCall" type="commandActionType" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="item" type="itemType" maxOccurs="unbounded" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="priority" type="priorityType" use="optional" />
</xsd:complexType>

<!--
    Define a file pair with source and destination
    <item mode=[binary|text]>
        <source recursive="false" disposition="leave">
            <file>filename</file>
        </source>

        <destination type="file" exist="error">
            <file>filename</file>
        </destination>
    </item>
-->
<xsd:complexType name="itemType">
    <xsd:sequence>
        <xsd:element name="source" type="fileSourceType" maxOccurs="1" minOccurs="1" />
        <xsd:element name="destination" type="fileDestinationType" maxOccurs="1" minOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="mode" type="modeType" use="required" />
    <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required" />
</xsd:complexType>

<!--
    Defines the request to delete scheduled file transfer.
    <deleteScheduledTransfer>
        <originator>
            <delete>
                <hostName>myMachine</hostName>
                <userID>myUserId</userID>

```

```

        </delete>
    </originator>
    <ID>56</ID>
</deleteScheduledTransfer>
-->
<xsd:complexType name="deleteScheduledTransferType">
  <xsd:sequence>
    <xsd:element name="originator" type="origDeleteType" maxOccurs="1" minOccurs="1" />
    <xsd:element name="ID" type="idType" maxOccurs="1" minOccurs="1" />
    <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="managedCallType">
  <xsd:sequence>
    <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="agent" type="agentType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="reply" type="replyType" maxOccurs="1" minOccurs="0" />
    <xsd:element name="transferSet" type="callTransferSetType" maxOccurs="1" minOccurs="1" />
    <xsd:element name="job" type="jobType" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="callTransferSetType">
  <xsd:sequence>
    <xsd:element name="metaDataSet" type="metaDataSetType" maxOccurs="1" minOccurs="0" />
    <xsd:element name="call" type="commandActionType" maxOccurs="1" minOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="priority" type="priorityType" use="optional" />
</xsd:complexType>
</xsd:schema>

```

## Understanding the transfer request message

The elements and attributes used in transfer request messages are described in the following list:

### Element descriptions

#### <request>

Group element containing all the elements required to specify a file transfer request.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

#### <managedTransfer>

Group element that contains all the elements required for a single file transfer or single group of file transfers.

#### <deleteScheduledTransfer>

Group element that contains originator and ID information to cancel a schedule transfer.

#### <managedCall>

Group element that contains all the elements required for a single managed call of a program or executable.

#### <ID>

Unique identifier that specifies the transfer request to delete from the list of pending scheduled transfers.

#### <originator>

Group element that contains the elements specifying the originator of the request.

#### <hostName>

The host name of the system where the source file is located.

#### <userID>

The user ID that originated the file transfer.

#### <mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

**<schedule>**

Group element describing the scheduled time for the file transfer, the repeat behavior, and when the next occurrence is due.

**<submit>**

Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. This attribute can have one of the following values: <ul style="list-style-type: none"> <li>• source - use the time zone of the source agent</li> <li>• admin - use the time zone of the administrator issuing the command</li> <li>• UTC - use Coordinated Universal Time</li> </ul>
timezone	The time zone description according to the timebase value

**<repeat>**

Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

**<frequency>**

The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: <ul style="list-style-type: none"> <li>• minutes</li> <li>• hours</li> <li>• days</li> <li>• weeks</li> <li>• months</li> <li>• years</li> </ul>

**<expireTime>**

Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

**<expireCount>**

Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

**<sourceAgent>**

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

**<destinationAgent>**

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

Attribute	Description
hostName	The host name or IP address of the agent queue manager.
portNumber	The port number used for client connections to the destination agent queue manager.
channel	The channel name used to connect to the destination agent queue manager.

#### <trigger>

Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The valid values are as follows: <ul style="list-style-type: none"> <li>• yes - log entries are created for failed triggered transfers</li> <li>• no - log entries are not created for failed triggered transfers</li> </ul>

#### <fileExist>

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileSize> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid values are as follows: <ul style="list-style-type: none"> <li>• = at least one file name in the name list must match</li> <li>• != a minimum of one of the files in the name list does not exist</li> </ul>
value	Indicates the comparison type: <ul style="list-style-type: none"> <li>• exist: file must exist</li> </ul>

#### <fileSize>

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileExist> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid value is as follows: <ul style="list-style-type: none"> <li>• &gt;= one of the file names in the name list exists and has a minimum size as specified in the value attribute</li> </ul>
value	File size specified as an integer value with units specified as one of the following: <ul style="list-style-type: none"> <li>• B - bytes</li> <li>• KB - kilobytes</li> <li>• MB - megabytes</li> <li>• GB - gigabytes</li> </ul> (the units value is not case-sensitive)

#### <reply>

Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key

**dynamicQueuePrefix** in the `command.properties` configuration file or the default of `WMQFTE.*` if not specified.

Attribute	Description
detailed	<p>Whether detailed transfer result information is required in the reply message. Multiple reply messages for each transfer can be generated. The valid values are as follows:</p> <ul style="list-style-type: none"> <li>• true - detailed reply information is required. The format of the information is the same as that published to the transfer log in the progress messages, that is, the <code>&lt;transferSet&gt;</code> element. For more information, see <a href="#">“File transfer log message formats” on page 2631</a>. Detailed reply information is present only when the transfer source agent has the <code>enableDetailedReplyMessages</code> property set to true.</li> <li>• false - detailed reply information is not required.</li> </ul> <p>The default value is false.</p>
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.
persistent	<p>Whether the message written to the reply queue is persistent. The valid values are as follows:</p> <ul style="list-style-type: none"> <li>• true - the message is persistent</li> <li>• false - the message is not persistent</li> <li>• qdef - the persistence of the message is defined by the properties of the reply queue</li> </ul> <p>The default value is false.</p>

#### <transferSet>

Specifies a group of file transfers you want to perform together or a group of managed calls that you want to perform together. During transmission `<transferSet>` is a group element containing `<item>` elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

#### <metaDataSet>

Optional group element containing one or more metadata items.

#### <metaData>

Specifies the user-defined metadata that is passed to the exit points called by the agent. The element contains the metadata value as a string.

Attribute	Description
key	Metadata name as a string

#### <call>

Group element that contains `<command>` elements specifying the program or executable to call.

#### <command>

Specifies the program or executable to call. The command must be located on the agent command path. For more information, see [Advanced agent properties](#). This element can contain optional `<argument>` elements.

Attribute	Description
name	The name of the command.

Attribute	Description
successRC	The successful return code that this command returns. Default is 0.
retryCount	The number of times that the command is to be retried if it fails.
retryWait	The time, in seconds, to wait between retries of the command.
type	The type of program to be called. The valid values are antscript, jcl, or executable.

**<argument>**

Specifies an argument to pass to the command.

**<item>**

Group element that contains elements specifying the source and destination file names and locations.


Attribute	Description
mode	Specifies the transfer mode as either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. The valid values are MD5 or none.


**<source>**

Group element that specifies files on the source system and whether they are removed after the transfer completes


Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid values are as follows: <ul style="list-style-type: none"> <li>• leave - the source files are left unchanged.</li> <li>• delete - the source files are deleted from the source system after the source file is successfully transferred.</li> </ul>

**<file>**

Specifies the transfer source.  For Multiplatforms, the transfer source can be a file or a directory name. For the z/OS platform, the transfer source can be a file, directory, data set, or PDS name. Use the fully qualified path in the format consistent with your operating system, for example C:/from/here.txt. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the source file. This alias is the name of the source file, excluding any directory path specified for the transfer.
EOL	Specifies the end of line marker for text transfers. Valid values are: <ul style="list-style-type: none"> <li>• LF - line feed character only</li> <li>• CRLF - carriage return and line feed character sequence</li> </ul>
encoding	The encoding of the source file for a text file transfer.
 delimiter	Specifies the delimiter that is included between records in record-oriented source files, for example, z/OS data sets. Specify the delimiter value as two hexadecimal digits in the range 00-FF, prefixed by x. For example, x12 or x03,x7F.



Attribute	Description
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values is as follows: <ul style="list-style-type: none"> <li>• binary - a hexadecimal delimiter</li> </ul> This attribute is available only if you have enabled the V7.0.4.1 function.
delimiterPosition	Specifies the position to insert delimiters when writing record-oriented source file records to a normal file. The valid values are as follows: <ul style="list-style-type: none"> <li>• prefix - the delimiter is inserted into the destination file before the data from each source record-oriented file record.</li> <li>• postfix - the delimiter is inserted into the destination file after the data from each source record-oriented file record.</li> </ul>
includeDelimiterInFile	Specifies whether to include a delimiter between records in record-oriented source files.
 keepTrailingSpaces	Specifies whether trailing spaces are to be kept on source records read from a fixed-length-format data set as part of a text mode transfer. The default is that trailing spaces are stripped. The valid values are as follows: <ul style="list-style-type: none"> <li>• true - trailing spaces are kept on source records read from a fixed-length-format data set</li> <li>• false - trailing spaces are stripped from source records read from a fixed-length-format data set</li> </ul>

#### <queue>

When used with the <source> element, specifies the name of the queue to transfer from, which must be located on the source agent queue manager. Use the format *QUEUE*. Do not include the queue manager name, the queue must be present on the source agent queue manager. You cannot use the <queue> element inside the <source> element, if you have used it inside of the <destination> element.



Attribute	Description
useGroups	Specifies whether to transfer all of the messages on the source queue, or either a complete message group or an individual message not in a group. The valid values are as follows: <ul style="list-style-type: none"> <li>• true - transfer only the first complete group of messages, or the first individual message not in a group.</li> <li>• false - transfer all messages on the source queue</li> </ul>
groupId	Specifies the group identifier of a complete message group, or the message identifier for an individual message not in a group, to read from the source queue. This attribute is valid only when the value of the useGroups attribute is true.
messageInGroup	Specifies whether the identifier in the groupId attribute represents a message group, or an individual message not in a group. This attribute is valid only when the value of the useGroups attribute is true. The valid values are as follows: <ul style="list-style-type: none"> <li>• true - the identifier in the groupId attribute represents a group identifier.</li> <li>• false - the identifier in the groupId attribute represents a message identifier.</li> </ul>

Attribute	Description
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values are as follows: <ul style="list-style-type: none"> <li>• text - a text or Java literal delimiter</li> <li>• binary - a hexadecimal delimiter</li> </ul>
delimiter	Specifies the delimiter that is included in the destination file between individual message data.
delimiterPosition	Specifies whether the delimiter is included in the destination file before or after individual message data. The valid values are as follows: <ul style="list-style-type: none"> <li>• prefix - the delimiter is included before the data</li> <li>• postfix - the delimiter is included after the data</li> </ul>
encoding	Specifies the source queue encoding.
waitTime	Specifies the time, in seconds, for the source agent to wait for either: <ul style="list-style-type: none"> <li>• a message to appear on the source queue, if the queue is empty or has become empty</li> <li>• a complete group to appear on the source queue, if the useGroups attribute has been set to true</li> </ul> <p>For information about setting the waitTime value, see <a href="#">“Guidance for specifying a wait time on a message-to-file transfer”</a> on page 2553.</p>

#### <destination>

Group element that specifies the destination and the behavior if files exist at the destination agent.

You can specify only one of <file> and <queue> as a child element of destination.

Attribute	Description
type	The type of destination. The valid values are as follows: <ul style="list-style-type: none"> <li>• file - specifies a file as the destination</li> <li>• directory - specifies a directory as the destination</li> <li>•  dataset - specifies a z/OS data set as the destination</li> <li>•  pds - specifies a z/OS partitioned data set as the destination</li> <li>• queue - specifies an IBM MQ queue as the destination</li> <li>• filespace - specifies a file space as the destination</li> </ul> <p>The value queue is valid only when the <b>&lt;destination&gt;</b> element has a child element of <b>&lt;queue&gt;</b>.</p> <p>The value filespace is valid only when the <b>&lt;destination&gt;</b> element has a child element of <b>&lt;filespace&gt;</b>.</p> <p>The other values are valid only when the <b>&lt;destination&gt;</b> element has a child element of <b>&lt;file&gt;</b>.</p>
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid values are as follows:

Attribute	Description
	<ul style="list-style-type: none"> <li>error - reports an error and the file is not transferred.</li> <li>overwrite - overwrites the existing destination file.</li> </ul> <p>This attribute is not valid if the <b>&lt;destination&gt;</b> element has a child element of <b>&lt;queue&gt;</b> or <b>&lt;filesystem&gt;</b>.</p>

#### <file>

Specifies additional settings for the previously-described **<destination>** element. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the <b>&lt;destination&gt;</b> file. This alias is the name of the source file, excluding any directory path specified for the transfer.
encoding	The encoding of the <b>&lt;destination&gt;</b> file for a text file transfer.
EOL	Specifies the end of line marker for text transfers. Valid values are: <ul style="list-style-type: none"> <li>LF - line feed character only</li> <li>CRLF - carriage return and line feed character sequence</li> </ul>
truncateRecords	Optional. Specifies that <b>&lt;destination&gt;</b> records longer than the LRECL data set attribute are truncated. <ul style="list-style-type: none"> <li>True - the records are truncated</li> <li>False - the records are wrapped</li> </ul> <p>The default setting is false.</p>

#### <queue>

When used with the **<destination>** element, specifies the name of the queue to transfer to, which can be located on any queue manager that is connected to the destination agent queue manager. Use the format `QUEUE@QM` where `QUEUE` is the name of the queue to put the messages on and `QM` is the queue manager where the queue is located. You cannot use the **<queue>** element inside the **<destination>** element, if you have used it inside of the **<source>** element.

Attribute	Description
delimiter	The delimiter to split the file into multiple messages.
delimiterType	Specifies the type of delimiter. The valid values are as follows: <ul style="list-style-type: none"> <li>text - a Java regular expression</li> <li>binary - a sequence of hexadecimal bytes</li> <li>size - a number of bytes, kibibytes, or mebibytes. For example, 1 B, 1 K, or 1 M.</li> </ul>
delimiterPosition	Specifies whether the delimiter is expected before or after the data to include in individual messages. The valid options are as follows: <ul style="list-style-type: none"> <li>prefix - the delimiter is expected before the data</li> <li>postfix - the delimiter is expected after the data</li> </ul>
includeDelimiterInMessage	A boolean specifying whether to include the delimiters that were used to split the file into multiple messages at the end of the messages.

Attribute	Description
encoding	Specifies the destination queue encoding.
persistent	Specifies whether the messages are persistent. The valid values are as follows: <ul style="list-style-type: none"> <li>• true - the messages are persistent</li> <li>• false - the messages are not persistent</li> <li>• qdef - the persistence value of the messages is defined by the settings on the destination queue</li> </ul>
setMqProps	A boolean specifying whether IBM MQ message properties are set on the first message in a file, and any messages written to the queue when an error occurs.
unrecognisedCodePage	Specifies whether a text mode transfer fails or conversion is performed, if the code page of the data is not recognized by the destination queue manager. The valid values are as follows: <ul style="list-style-type: none"> <li>• fail - the transfer reports a failure</li> <li>• binary - the data is converted to the destination code page and the IBM MQ message header describing the format of the data is set to MQFMT_NONE.</li> </ul> The default behavior is fail.

#### <filespace>

Group element specifying the name of the file space to transfer to.

#### <name>

When used with the <filespace> element, the value of this element specifies the name of the file space.

#### <preSourceCall>

Group element specifying a command to call at the source of the transfer, before the transfer starts.

#### <postSourceCall>

Group element specifying a command to call at the source of the transfer, after the transfer completes.

#### <preDestinationCall>

Group element specifying a command to call at the destination of the transfer, before the transfer starts.

#### <postDestinationCall>

Group element specifying a command to call at the destination of the transfer, after the transfer completes.

#### <command>

When used with the <preSourceCall>, <postSourceCall>, <preDestinationCall>, or <postDestinationCall> element, this element specifies the command to be called. The command must be located on the agent command path. For more information, see [Advanced agent properties](#).

Attribute	Description
name	The name of the command to run.
successRC	The return code that is expected if the command runs successfully.

#### <argument>

When used with the <command> element, this element specifies an argument to be passed in to the command. You can have any number of <argument> elements inside a <command> element.

**<job>**

Optional group element containing job information for the entire transfer specification. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that appears in the transfer log message, which is described in the following topic: [“File transfer log message formats” on page 2631.](#)

**<name>**

When used with the <job> element, the value of this element specifies the name of the job.

**<transferSpecifications>**

Group element that contains <item> elements for multiple transfer groups. See [Using transfer definition files](#) for further details about how to use this element.

**<cancel>**

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

**<transfer>**

When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

**<job>**

Group element containing job information.

**<jobName>**

Specifies logical job identifier.

**File transfer cancel message format**

A file transfer request returns a 48-character ID that identifies the transfer for a specific agent. This ID is used to cancel transfers.

**Understanding the transfer cancel message**

The elements and attributes used in transfer cancel messages are described:

**<cancel>**

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

**<originator>**

Group element that contains the elements specifying the originator of the request.

**<hostName>**

The host name of the system where the source file is located.

**<userID>**

The user ID that originated the file transfer.

**<mqmdUserID>**

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

**<transfer>**

When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

**<job>**

Optional. Group element containing job information.

**<jobName>**

Specifies logical job identifier.

## Examples

Examples of XML messages that conform to this schema are provided for each of the following requests:

- [Create a file transfer](#)
- [Create an asynchronous file transfer request](#)
- [Cancel a file transfer](#)
- [Create a scheduled transfer](#)
- [Delete a scheduled transfer](#)
- [Create a managed call](#)
- [Create a file transfer that includes managed calls](#)

## Related reference

[“File transfer request message examples” on page 2679](#)

Examples of the messages that you can put on the agent command queue to request that the agent create or cancel a transfer.

[“Scheduled file transfer message examples” on page 2680](#)

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a schedule.

[“MFT agent call request message examples” on page 2681](#)

Examples of the messages that you can put on the agent command queue to request that the agent creates a managed call or creates a transfer that calls programs.

[“MFT agent status message format” on page 2615](#)

When a Managed File Transfer Agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the SYSTEM.FTE/Agents/*agent name* topic).

[“File transfer status message format” on page 2627](#)

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its SYSTEM.FTE/Transfers/*agent\_name/transfer ID* topic), which conforms to the TransferStatus.xsd XML schema. The TransferStatus.xsd file is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory of your WMQMFT installation.

[“File transfer log message formats” on page 2631](#)

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent\_name/transfer\_id*. These messages conform to the schema TransferLog.xsd, which is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory of your Managed File Transfer installation.

[“Scheduled file transfer log message formats” on page 2653](#)

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent name/schedule ID* topic). This message conforms to the ScheduleLog.xsd XML schema.

[“MFT monitor request message formats” on page 2682](#)

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the **fteCreateMonitor** command or using the IBM MQ Explorer interface.

[“MFT message formats for security” on page 2695](#)

This topic describes the messages published to the Managed File Transfer coordination queue manager relevant to security.

### ***File transfer request message examples***

Examples of the messages that you can put on the agent command queue to request that the agent create or cancel a transfer.

#### **Create transfer request**

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
version="4.00"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
    <destinationAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
    <transferSet>
      <item checksumMethod="MD5" mode="binary">
        <source disposition="leave" recursive="false">
          <file>/etc/passwd</file>
        </source>
        <destination exist="overwrite" type="directory">
          <file>/tmp</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

#### **Create transfer request - synchronous**

When a user requests a blocking synchronous request, that is, they wait for the transfer to complete and receive status messages, the message placed on the command queue contains a reply element that specifies the queue that a reply message is sent to. The following example shows the message placed on the command queue used by FTEAGENT:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="FTEAGENT"
      QMgr="QM1"/>
    <destinationAgent agent="AGENT2"
      QMgr="QM2"/>
    <reply QMGR="QM1">WMQFTE.492D0D5502770020</reply>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\sourcefiles\source1.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\dest1.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

The <reply> element is populated with the name of the command queue manager where a temporary dynamic queue has been created to receive reply about the successful (or otherwise) completion of the transfer. The name of the temporary dynamic queue is composed of two parts:

- The prefix as defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file (it is `WMQFTE.` by default)
- The ID of the queue as generated by IBM MQ

## Cancel transfer request

```
<?xml version="1.0" encoding="UTF-8"?>
<cancel xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
  <transfer>414D5120553322E42494E444494E47538B0F404D032C0020</transfer>
  <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20002007</reply>
</cancel>
```

### Related reference

“File transfer request message format” on page 2665

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

### Scheduled file transfer message examples

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a schedule.

## Create scheduled transfer

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <schedule>
      <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00</submit>
    </schedule>
    <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
    <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
    <transferSet>
      <item checksumMethod="MD5" mode="binary">
        <source disposition="leave" recursive="false">
          <file>/etc/passwd</file>
        </source>
        <destination exist="overwrite" type="directory">
          <file>/tmp</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

## Delete scheduled transfer

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="4.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
```



```

<deleteScheduledTransfer>
  <originator>
    <delete>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </delete>
  </originator>
  <ID>1</ID>
  <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003902</reply>
</deleteScheduledTransfer>
</request>

```

## Related reference

[“File transfer request message format” on page 2665](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the FileTransfer.xsd schema and have the <request> element as the root element. The FileTransfer.xsd schema document is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

## MFT agent call request message examples

Examples of the messages that you can put on the agent command queue to request that the agent creates a managed call or creates a transfer that calls programs.

### Managed call request example

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="1.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedCall>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <agent agent="DNWE" QMgr="QM1"/>
    <transferSet>
      <call>
        <command name="echo" successRC="0">
          <argument>call</argument>
          <argument>test</argument>
        </command>
      </call>
    </transferSet>
  </managedCall>
  <job>
    <name>managedCallCalls.xml</name>
  </job>
</request>

```

### Managed transfer request example with calls

```

<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="1.00"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>mqm</userID>
    </originator>
    <sourceAgent agent="DNWE" QMgr="QM1"/>
    <destinationAgent agent="DNWE" QMgr="QM1"/>
    <transferSet>
      <preSourceCall>
        <command name="echo" successRC="0">
          <argument>preSourceCall</argument>
          <argument>test</argument>
        </command>

```

```

</preSourceCall>
<postSourceCall>
  <command name="echo" successRC="0">
    <argument>postSourceCall</argument>
    <argument>test</argument>
  </command>
</postSourceCall>
<preDestinationCall>
  <command name="echo" successRC="0">
    <argument>preDestinationCall</argument>
    <argument>test</argument>
  </command>
</preDestinationCall>
<postDestinationCall>
  <command name="echo" successRC="0">
    <argument>postDestinationCall</argument>
    <argument>test</argument>
  </command>
</postDestinationCall>
</transferSet>
<job>
  <name>managedTransferCalls.xml</name>
</job>
</managedTransfer>
</request>

```

## Related tasks

[Specifying programs to run with MFT](#)

## Related reference

“File transfer request message format” on page 2665

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

## MFT monitor request message formats

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the **fteCreateMonitor** command or using the IBM MQ Explorer interface.

The monitor XML must conform to the `Monitor.xsd` schema using the `<monitor>` element as the root element.

Monitor messages can have one of the following root elements:

- `<monitor>` - for creating and starting a new resource monitor
- `<deleteMonitor>` - for stopping and deleting an existing monitor

There is no command message for the **fteListMonitors** command because the command directly retrieves matching monitor definitions from the `SYSTEM.FTE` topic.

## Schema

The following schema describes which elements are valid in a monitor request XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  targetNamespace="https://www.ibm.com/xmlns/wmqfte/7.0.1/
  MonitorDefinition"
  xmlns="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">
  <xsd:include schemaLocation="FileTransfer.xsd" />
  <xsd:element name="monitor">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="monitorNameType"

```

```

        minOccurs="1"           maxOccurs="1" />
        <xsd:element name="description" type="xsd:string"
        minOccurs="0"           maxOccurs="1" />
        <xsd:element name="pollInterval" type="pollIntervalType"
        minOccurs="1"           maxOccurs="1"
default="10" />
        <xsd:element name="batch" type="batchType"
        minOccurs="0"           maxOccurs="1" />
        <xsd:element name="agent" type="agentNameType"
        minOccurs="1"           maxOccurs="1" />
        <xsd:element name="resources" type="monitorResourcesType"
        minOccurs="0"
maxOccurs="1" />
        <xsd:element name="triggerMatch" type="triggerMatchType"
        maxOccurs="1"           minOccurs="1" />
        <xsd:element name="reply" type="replyType"
        maxOccurs="1"           minOccurs="0" />
        <xsd:element name="tasks" type="monitorTasksType"
        maxOccurs="1"           minOccurs="1" />
        <xsd:element name="originator" type="origRequestType"
        maxOccurs="1"           minOccurs="1" />
        <xsd:element name="job" type="jobType"
        maxOccurs="1"           minOccurs="0" />
        <xsd:element name="defaultVariables" type="defaultVariablesType"
        maxOccurs="1"           minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required" />
</xsd:complexType>
</xsd:element>

<xsd:element name="deleteMonitor">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="name" type="monitorNameType"
        minOccurs="1"           maxOccurs="1" />
      <xsd:element name="originator" type="origRequestType"
        maxOccurs="1"           minOccurs="1" />
      <xsd:element name="reply" type="replyType"
        maxOccurs="1"           minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required" />
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="transferRequestType">
  <xsd:choice>
    <xsd:element name="managedTransfer" type="managedTransferType" />
    <xsd:element name="managedCall" type="managedCallType" />
  </xsd:choice>
  <xsd:attribute name="version" type="versionType" />
</xsd:complexType>

<xsd:complexType name="monitorResourcesType">
<xsd:choice>
  <xsd:sequence>
    <xsd:element name="directory" type="monitoredDirectoryType"
      minOccurs="1"           maxOccurs="1" />
  </xsd:sequence>
  <xsd:element name="queue" type="monitoredQueueType" />
</xsd:choice>
</xsd:complexType>

<xsd:complexType name="monitoredDirectoryType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="recursionLevel"
type="xsd:nonNegativeInteger" />
      <xsd:attribute name="id" type="resourceIdAttrType" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="monitoredQueueType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="id" type="resourceIdAttrType" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="triggerMatchType">
  <xsd:sequence>

```

```

        <xsd:element name="conditions" type="conditionsType"
          minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="conditionsType">
      <xsd:choice minOccurs="1">
        <xsd:element name="allOf" type="listPredicateType"
          minOccurs="1" maxOccurs="1" />
        <xsd:element name="anyOf" type="listPredicateType"
          minOccurs="1" maxOccurs="1" />
        <xsd:element name="condition" type="conditionType"
          minOccurs="1" maxOccurs="1" />
      </xsd:choice>
    </xsd:complexType>

    <xsd:complexType name="listPredicateType">
      <xsd:choice>
        <xsd:element name="condition" type="conditionType"
          minOccurs="1" maxOccurs="unbounded" />
      </xsd:choice>
    </xsd:complexType>

    <xsd:complexType name="conditionType">
      <xsd:sequence>
        <xsd:element name="name" type="conditionNameType"
          minOccurs="0" maxOccurs="1" />
        <xsd:element name="resource" type="resourceIdType"
          minOccurs="0" maxOccurs="1" />
        <xsd:choice minOccurs="1">
          <xsd:element name="fileMatch"
            type="fileMatchConditionType"
            minOccurs="1" maxOccurs="1" />
          <xsd:element name="fileNoMatch"
            type="fileNoMatchConditionType"
            minOccurs="1"
            maxOccurs="1" />
          <xsd:element name="fileSize"
            type="fileSizeConditionType"
            minOccurs="1" maxOccurs="1" />
          <xsd:element name="queueNotEmpty"
            type="queueNotEmptyConditionType"
            minOccurs="1" maxOccurs="1" />
          <xsd:element name="completeGroups"
            type="completeGroupsConditionType"
            minOccurs="1" maxOccurs="1" />
          <xsd:element name="fileSizeSame"
            type="fileSizeSameType"
            minOccurs="1" maxOccurs="1" />
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="fileMatchConditionType">
      <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
          minOccurs="0" default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
          minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="fileNoMatchConditionType">
      <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
          minOccurs="0" default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
          minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="fileSizeConditionType">
      <xsd:sequence>
        <xsd:element name="compare" type="sizeCompareType"
          minOccurs="1" default="0" />
        <xsd:element name="pattern" type="conditionPatternType"
          minOccurs="0" default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
          minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="sizeCompareType">

```

```

        <xsd:simpleContent>
            <xsd:extension base="xsd:int">
                <xsd:attribute name="operator" type="sizeOperatorType"
use="required" />
                <xsd:attribute name="units" type="fileSizeUnitsType"
use="required" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:simpleType name="sizeOperatorType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value=">=" />
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="fileSizeUnitsType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[bB]|[kK][bB]|[mM][bB]|[gG][bB]" />
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="conditionPatternType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="type" type="patternTypeAttributeType"
use="optional" default="wildcard"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:simpleType name="patternTypeAttributeType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="regex" />
            <xsd:enumeration value="wildcard" />
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="conditionNameType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string" />
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="queueNotEmptyConditionType"/>

    <xsd:complexType name="completeGroupsConditionType"/>

    <xsd:complexType name="fileSizeSameType">
        <xsd:sequence>
            <xsd:element name="pattern" type="conditionPatternType"
minOccurs="1" maxOccurs="1"/>
            <xsd:element name="exclude" type="conditionPatternType"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="polls" type="positiveIntegerType" use="required" />
    </xsd:complexType>

    <xsd:complexType name="pollIntervalType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:int">
                <xsd:attribute name="units" type="timeUnitsType"
use="optional" default="minutes" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="batchType">
        <xsd:attribute name="maxSize" type="positiveIntegerType" use="required"/>
    </xsd:complexType>

    <xsd:simpleType name="timeUnitsType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="seconds" />
            <xsd:enumeration value="minutes" />
            <xsd:enumeration value="hours" />
            <xsd:enumeration value="days" />
            <xsd:enumeration value="weeks" />
            <xsd:enumeration value="months" />
            <xsd:enumeration value="years" />
        </xsd:restriction>
    </xsd:simpleType>

```

```

<xsd:complexType name="monitorTasksType">
  <xsd:sequence>
    <xsd:element name="task" type="monitorTaskType"
      minOccurs="1" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="monitorTaskType">
  <xsd:sequence>
    <xsd:element name="name" type="monitorTaskNameType"
      minOccurs="1" maxOccurs="1" />
    <xsd:element name="description" type="xsd:string"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="transfer" type="transferTaskType"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="transferTaskType">
  <xsd:sequence>
    <xsd:element name="request" type="transferRequestType"
      minOccurs="1" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="resourceIdType">
  <xsd:attribute name="id" type="xsd:string" use="optional" />
</xsd:complexType>

<xsd:simpleType name="resourceIdAttrType">
  <xsd:restriction base="xsd:string"></xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="monitorNameType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[^%\*]+" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="agentNameType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[%_0-9A-Z]*" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="monitorTaskNameType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value=".*" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="defaultVariablesType">
  <xsd:sequence>
    <xsd:element name="variable" type="variableType"
      maxOccurs="unbounded" minOccurs="1" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="variableType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="key" type="xsd:string" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

</xsd:schema>

```

From IBM MQ 9.3.0, the `maxOccurs` attribute of the `directory` element is set to 1. This attribute was previously set to unbounded, which indicated that there could be multiple directory entries. However, this was incorrect because you can only specify one directory when creating a resource monitor that monitors a directory.

## Create monitor message elements

The following elements and attributes are used in create monitor messages:

**<monitor>**

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

**<name>**

The name of the monitor, unique within the monitor's agent.

**<description>**

Description of the monitor (not currently used).

**<pollInterval>**

The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	Specifies the time units for the poll interval. Valid values are: <ul style="list-style-type: none"> <li>• seconds</li> <li>• minutes</li> <li>• hours</li> <li>• days</li> <li>• weeks</li> <li>• months</li> <li>• years</li> </ul>

**<agent>**

Name of the agent the monitor is associated with.

**<resources>**

Group element that contains the elements specifying the resources to monitor.

**<directory>**

Fully qualified path specifying the directory on the monitor's agent machine to monitor.

Attribute	Description
recursionLevel	The number of subdirectories to monitor in addition to the specified directory.
id	Unique identifier for the resource.

**<queue>**

Queue name specifying the queue to monitor on the monitoring agent's queue manager.

**<triggerMatch>**

Group element that contains the elements specifying the trigger conditions to compare with the monitored resource.

**<conditions>**

Group element that contains the elements specifying the type of condition to compare with the monitored resource.

**<allOf>**

Predicate that specifies that all contained conditions must be satisfied.

**<anyOf>**

Predicate that specifies that any contained conditions must be satisfied.

**<condition>**

Defines a comparison condition that will contribute to the overall monitor trigger condition.

**<name>**

Name of the condition.

**<resource>**

Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:

- fileMatch
- fileNoMatch
- fileSize

If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:

- queueNotEmpty
- completeGroups

**<fileMatch>**

Group element for a file name match condition.

**<pattern>**

Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is \* (any file will match).

**<fileNoMatch>**

Group element for an inverse file name match condition.

**<pattern>**

Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is \* (the absence of any file will match).

**<fileSize>**

Group element for a file size comparison.

**<compare>**

Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only '>=' is supported.
units	Specifies file size units, which can be one of: <ul style="list-style-type: none"> <li>• B - bytes</li> <li>• KB - kilobytes</li> <li>• MB - megabytes</li> <li>• GB - gigabytes</li> </ul> The units value is case insensitive, so 'mb' works as well as 'MB'.

**<pattern>**

File name pattern to match. Default is \* (any file will match).

**<queueNotEmpty>**

This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.



**<completeGroups>**

This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

**<reply>**

Optional element that is used to specify reply queue for asynchronous requests.

Attribute	Description
QMGR	Queue manager name.

**<tasks>**

Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

**<task>**

Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

**<name>**

Name of the task. Accepts any alphanumeric characters.

**<description>**

Description of the task. Any text value is allowed.

**<transfer>**

Group element that defines a transfer task.

**<request>**

Group element that defines the type of task. This must contain one of the following elements which are inherited from the `FileTransfer.xsd` schema definition:

- [managedTransfer](#)
- [managedCall](#)

Attribute	Description
version	Version of the request as provided by Managed File Transfer. This is in the form n.mm where n is the major release version and mm is the minor version. For example 1.00.

**<originator>**

Group element that contains the elements specifying the originator of the request.

**<hostName>**

The host name of the system where the source file is located.

**<userID>**

The user ID that originated the file transfer.

**<mqmdUserID>**

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

**<job>**

Group element containing job information.

**<jobName>**

Specifies logical job identifier.

**<defaultVariables>**

Group element containing one or more variable elements. These variables are used in variable substitution when monitoring a queue. For more information about variable substitution, see [Customizing MFT tasks with variable substitution](#).

**<variable>**

Element containing the value associated with the key given by the key attribute.

Attribute	Description
key	The name of the default variable.

## Delete monitor message elements

The following elements and attributes are used in delete monitor messages:

### <deleteMonitor>

Group element containing all the elements required to stop and delete a monitor.

Attribute	Description
version	Specifies the version of this element as supplied by Managed File Transfer.

### <name>

Name of monitor to delete.

### <originator>

Group element that contains the elements specifying the originator of the request.

### <hostName>

The host name of the system where the source file is located.

### <userID>

The user ID that originated the file transfer.

### <mqmdUserID>

Optional. The IBM MQ user ID that was supplied in the message descriptor (MQMD).

### <reply>

Specifies the name of the temporary reply queue generated for the request. The name of the queue is as defined by the key `dynamicQueuePrefix` in the `command.properties` configuration file. If this is not specified, the queue name has a default value of `WMQFTE`.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

## Examples

Examples of XML messages that conform to this schema are provided for each of the following monitor requests:

- [Create a monitor](#)
- [Delete a monitor](#)

## Related tasks

[Monitoring MFT resources](#)

## Related reference

[“MFT monitor request message examples” on page 2691](#)

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a monitor.

[“MFT agent status message format” on page 2615](#)

When a Managed File Transfer Agent is created or started, the agent publishes its details to the `SYSTEM.FTE` topic on its coordination queue manager (on the `SYSTEM.FTE/Agents/agent name` topic).

[“File transfer request message format” on page 2665](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The

FileTransfer.xsd schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

#### [“File transfer status message format” on page 2627](#)

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its `SYSTEM.FTE/Transfers/agent_name/transfer ID` topic), which conforms to the schema TransferStatus.xsd XML schema. The TransferStatus.xsd file is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your WMQMFT installation.

#### [“File transfer log message formats” on page 2631](#)

File transfer log messages are published to the `SYSTEM.FTE` topic with a topic string of `Log/agent_name/transfer_id`. These messages conform to the schema TransferLog.xsd, which is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of your Managed File Transfer installation.

#### [“Scheduled file transfer log message formats” on page 2653](#)

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its `SYSTEM.FTE/Log/agent name/schedule ID` topic). This message conforms to the ScheduleLog.xsd XML schema.

#### [“MFT message formats for security” on page 2695](#)

This topic describes the messages published to the Managed File Transfer coordination queue manager relevant to security.

### **MFT monitor request message examples**

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a monitor.

### **Create monitor request**

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  version="4.00"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./
Monitor.xsd">
  <name>EXAMPLEMONITOR</name>
  <pollInterval>1</pollInterval>
  <agent>US2.BINDINGS.FILE</agent>
  <resources>
    <directory recursionLevel="0">/srv/nfs/incoming</directory>
  </resources>
  <triggerMatch>
    <conditions>
      <allOf>
        <condition>
          <fileMatch>
            <pattern>*.completed</pattern>
          </fileMatch>
        </condition>
      </allOf>
    </conditions>
  </triggerMatch>
  <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003702</reply>
  <tasks>
    <task>
      <name/>
      <transfer>
        <request xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
          version="4.00"
          xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
          <managedTransfer>
            <originator>
              <hostName>example.com.</hostName>
              <userID>mqm</userID>
            </originator>
            <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
            <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
          </managedTransfer>
        </request>
      </transfer>
    </task>
  </tasks>
</monitor>
```

```

        <item checksumMethod="MD5" mode="binary">
          <source disposition="leave" recursive="false">
            <file>/srv/nfs/incoming/*.txt</file>
          </source>
          <destination exist="error" type="directory">
            <file>/srv/backup</file>
          </destination>
        </item>
      </transferSet>
    </managedTransfer>
  </request>
</transfer>
</task>
</tasks>
<originator>
  <hostName>example.com.</hostName>
  <userID>mqm</userID>
</originator>
</monitor:monitor>

```

## Delete monitor request

```

<?xml version="1.0" encoding="UTF-8"?>
<monitor:deleteMonitor xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:monitor="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
  version="4.00"
  xsi:schemaLocation="https://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./
Monitor.xsd">
  <name>EXAMPLEMONITOR</name>
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
  <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003705</reply>
</monitor:deleteMonitor>

```

### Related reference

[“MFT monitor request message formats” on page 2682](#)

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the **fteCreateMonitor** command or using the IBM MQ Explorer interface.

## Ping MFT agent request message format

You can ping an agent by issuing an **ftePingAgent** command or by putting an XML message on the agent command queue. The ping agent request XML must conform to the `PingAgent.xsd` schema. After you have installed Managed File Transfer, you can find the `PingAgent.xsd` schema file in the following directory: `MQ_INSTALLATION_PATH/mqft/samples/schema`. The `PingAgent.xsd` schema imports `fteutils.xsd`, which is in the same directory.

When the agent receives a ping agent request message on its command queue, if the agent is active, it returns an XML response message to the command or application that put the ping agent request message on the command queue. The response message from the agent is in the format defined by `Reply.xsd`. For more information about this format, see [“MFT agent reply message format” on page 2693](#).

### Schema

The following schema describes which elements are valid in an ping agent request XML message.

```

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  xmlns="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
  targetNamespace="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent">
  <xsd:include schemaLocation="fteutils.xsd"/>
  <xsd:element name="pingAgent">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>

```

```

        <xsd:element name="agent"          type="agentType"          maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply"        type="replyType"         maxOccurs="1" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="version" type="versionType" use="required" />
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

## Understanding the ping agent request message

The elements and attributes used in the ping agent request messages are described in the following list:

### <pingAgent>

Group element containing all the elements required to specify a ping agent request.

### <originator>

Group element containing all the elements required to specify the originator of the ping request.

#### <hostName>

The host name of the machine where the request originated.

#### <userID>

The user name of the originator of the request.

#### <mqmdUserID>

The MQMD user name of the originator of the request.

### <agent>

The agent to ping.

Attribute	Description
agent	Required. The name of the agent.
QMGr	Optional. The queue manager that the agent connects to.

### <reply>

The name of the queue for the agent to send the reply message to.

Attribute	Description
QMGR	Required. The name of the queue manager where the reply queue is located.

## Example

This example shows a ping agent message sent to the agent AGENT\_JUPITER. If AGENT\_JUPITER is active and able to process agent requests, it sends a response message to the queue WMQFTE.4D400F8B20003708 on QM\_JUPITER.

```

<?xml version="1.0" encoding="UTF-8"?>
<ping:pingAgent xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xmlns:ping="https://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
  version="4.00">
  <originator>
    <hostName>example.com.</hostName>
    <userID>mqm</userID>
  </originator>
  <agent agent="AGENT_JUPITER" QMGr="QM_JUPITER"/>
  <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003708</reply>
</ping:pingAgent>

```

## MFT agent reply message format

When an agent receives an XML message on its agent command queue, if a response is required, the agent will send an XML reply message to the reply queue defined in the original message. The reply XML conforms to the Reply.xsd schema. The Reply.xsd schema document is located in

the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory. The `Reply.xsd` schema imports `fteutils.xsd`, which is in the same directory.

## Schema

The following schema describes which elements are valid in a reply XML message.

```
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="TransferLog.xsd"/>
  <xsd:element name="reply">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="transferSet" type="transferSetType" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="status" type="statusType" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="version" type="versionType" use="required"/>
      <xsd:attribute name="ID" type="IDType" use="required"/>
      <xsd:attribute name="detailedReplyMessagesDisabled" type="xsd:boolean"
use="optional"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

## Understanding the reply message

The elements and attributes used in the reply messages are described in the following list:

### <reply>

Element containing the elements that specify the reply information.

Attribute	Description
ID	The ID of the reply.
version	The version of the reply message format.
detailedReplyMessagesDisabled	A notification that the agent has disabled the detailed reply enableDetailedReplyMessages agent property is set to false).

### <transferSet>

Specifies the transfer result information of the files requested for transfer. For more information, see [“File transfer log message formats” on page 2631](#).

### <status>

The status of the action that the agent was requested to perform.

Attribute	Description
resultCode	The result code returned from the action that the agent performed.

### <supplement>

Additional response information about the action that the agent was requested to perform.

## Example

In the following section is an example reply message:

```
<reply version="1.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Reply.xsd"
  ID="0102020300000000000000000000000000000000000000000000000000000000">
  <status resultCode="65">
    <supplement>Additional reply information</supplement>
```

```
</status>
</reply>
```

## MFT message formats for security

This topic describes the messages published to the Managed File Transfer coordination queue manager relevant to security.

### Not authorized log message

If user authority checking is enabled the agent can publish not authorized messages to the coordination queue manager. [Restricting user authorities on MFT agent actions](#) describes how to enable user authority checking.

Every time a user submits a request to perform a restricted action to the agent, either by using an Managed File Transfer command or by using the IBM MQ Explorer plugin, the agent checks that the user has the authority to perform the action. If the user fails that authority check, a not authorized log message is published to the coordination queue manager on its SYSTEM.FTE/Log/*agent\_name*/NotAuthorized topic.

This message conforms to the TransferLog.xsd XML schema. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<notAuthorized version="3.00"
  ID="414d5120716d31202020202020202020204da5924a2010ce03"
  agentRole="sourceAgent"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="TransferLog.xsd"
  xmlns="">
  <action time="2009-08-28T12:31:15.781Z">not_authorized</action>
  <originator>
    <mqmdUserID>test1</mqmdUserID>
  </originator>
  <authority>administration</authority>
  <status resultCode="53">
    <supplement>BFGCH0083E: The user (test1) does not have the authority (ADMINISTRATION) required
to shut down agent 'AGENT'.</supplement>
    <supplement>
      &lt;?xml version="1.0" encoding="UTF-8"?&gt;
&lt;internal:request version="3.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-
instance"
  xmlns:internal="http://wmqfte.ibm.com/internal"&gt;
&lt;internal:shutdown agent="SYSTEM.FTE.COMMAND.AGENT" hostname="qm1"
mode="controlled"/&gt;
&lt;reply QMGR="qm1"&gt;WMQFTE.4A92A54D02CE1020&lt;/reply&gt;
&lt;/internal:request&gt;
    </supplement>
  </status>
</notAuthorized>
```

This message is a log of the following information:

- Who originated the request
- The level of Managed File Transfer access authority required to perform the request
- The status of the request
- The request specification

### Understanding the not authorized log message

The elements and attributes used in the not authorized message are described:

#### <notAuthorized>

Group element that describes a single failed user authorization check.

Attribute	Description
version	Specifies the version of this element as detailed by Managed File Transfer.

Attribute	Description
ID	The unique identifier for the request that was not authorized.

**<originator>**

Group element that contains the elements specifying the originator of the request.

**<authority>**

Specifies the level of Managed File Transfer access authority that the user required to perform the requested action.

**<mqmdUserID>**

The IBM MQ user ID that was supplied in the message descriptor (MQMD)

**<action>**

Specifies the authorization status of the request matching the ID attribute of <notAuthorized> element.

Attribute	Description
time	Specifies the date and time the log entry was published (in date time format).

**<status>**

The result code and supplement messages.

**Related reference**

[“MFT agent status message format” on page 2615](#)

When a Managed File Transfer Agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the SYSTEM.FTE/Agents/*agent name* topic).

[“File transfer request message format” on page 2665](#)

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the IBM MQ Explorer. The transfer request XML must conform to the FileTransfer.xsd schema and have the <request> element as the root element. The FileTransfer.xsd schema document is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

[“File transfer status message format” on page 2627](#)

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its SYSTEM.FTE/Transfers/*agent\_name/transfer ID* topic), which conforms to the TransferStatus.xsd XML schema. The TransferStatus.xsd file is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory of your WMQMFT installation.

[“File transfer log message formats” on page 2631](#)

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent\_name/transfer\_id*. These messages conform to the schema TransferLog.xsd, which is located in the MQ\_INSTALLATION\_PATH/mqft/samples/schema directory of your Managed File Transfer installation.

[“Scheduled file transfer log message formats” on page 2653](#)

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent\_name/schedule ID* topic). This message conforms to the ScheduleLog.xsd XML schema.

[“MFT monitor request message formats” on page 2682](#)



Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the **fteCreateMonitor** command or using the IBM MQ Explorer interface.


## MFT credentials file format

The `MQMFTCredentials.xml` file contains sensitive user ID and password information. The elements in the `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The security of credentials files is the responsibility of the user.

The **useMQCSPAAuthentication** parameter enables and disables MQCSP authentication for a Managed File Transfer agent. You can set this parameter in the `MQMFTCredentials.xml` file. For more information, see [Enabling connection authentication for MFT](#).

From IBM MQ 9.1.1, MQCSP authentication is enabled by default for the MFT agents and logger. If the **useMQCSPAAuthentication** parameter is not specified, it is by default set to `true`.

Before IBM MQ 9.1.1, compatibility mode is the default and if the **useMQCSPAAuthentication** parameter is not specified, it is by default set to `false`.

 From IBM MQ 8.0, the `MQMFTCredentials.xsd` file can also be a PDSE member on z/OS.

Before IBM WebSphere MQ 7.5 the information contained in the `MQMFTCredentials.xml` file was held in separate properties files.

The `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The `MQMFTCredentials.xml` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the Managed File Transfer installation.

## Schema

The following schema describes which elements are valid in the `MQMFTCredentials.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  @start_non_restricted_prolog@
  Version: %Z% %I% %W% %E% %U% [%H% %T%]

  Licensed Materials - Property of IBM

  5724-H72

  Copyright IBM Corp. 2012, 2024. All Rights Reserved.

  US Government Users Restricted Rights - Use, duplication or
  disclosure restricted by GSA ADP Schedule Contract with
  IBM Corp.
  @end_non_restricted_prolog@
-->

<!--
  This schema defines the format of an MQMFTCredentials file. Files of this type
  store credential information for agent and logger processes. They can contain
  user names and passwords either in clear text or which have been obfuscated
  using the fteObfuscate command.
-->

<!-- Example mqmftCredentials.xml file:
<?xml version="1.0" encoding="UTF-8"?>
<tns:mqmftCredentials xmlns:tns="http://wmqfte.ibm.com/
MQMFTCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/MQMFTCredentials MQMFTCredentials.xsd">

  <tns:logger name="LOG1" user="user1" password="passw0rd"/>
  <tns:logger name="ORACLE" userCipher="kj2h3dfkgf" passwordCipher="1a3n67eaer"/>
  <tns:file path="/home/emma/trust.jks" password="passw0rd"/>
  <tns:file path="/var/tmp/keystore.jks" passwordCipher="e71vKCg2pf"/>

  <tns:qmgr name="QM_COORD" user="tim" mqUserId="user1" mqPassword="passw0rd"/>
  <tns:qmgr name="QM_COORD" user="tom" mqUserId="user1" mqPasswordCipher="e71vKCg2pf"/>
```

```

    <tns:qmgr name="QM_COORD" user="ernest" mqUserId="ernest"
mqPassword="AveryL0ngPassw0rd2135" useMQCSPAAuthentication="true"/>
  </tns:mqmftCredentials>
-->

<schema targetNamespace="http://wmqfte.ibm.com/MQMFTCredentials"
elementFormDefault="qualified"
xmlns="https://www.w3.org/2001/XMLSchema"
xmlns:tns="http://wmqfte.ibm.com/MQMFTCredentials">

  <element name="mqmftCredentials" type="tns:mqmftCredentialsType"/>

  <complexType name="mqmftCredentialsType">
    <sequence>
      <choice minOccurs="0" maxOccurs="unbounded">
        <element name="logger" type="tns:loggerType"/>
        <element name="file" type="tns:fileType"/>
        <element name="qmgr" type="tns:mqUserPassType"/>
      </choice>
    </sequence>
  </complexType>

  <complexType name="loggerType">
    <attribute name="name" type="string" use="required"/>
    <attribute name="user" type="string" use="optional"/>
    <attribute name="userCipher" type="string" use="optional"/>
    <attribute name="password" type="string" use="optional"/>
    <attribute name="passwordCipher" type="string" use="optional"/>
  </complexType>

  <complexType name="fileType">
    <attribute name="path" type="string" use="required"/>
    <attribute name="password" type="string" use="optional"/>
    <attribute name="passwordCipher" type="string" use="optional"/>
  </complexType>

  <!-- Example XML:

  <tns:qmgr name="QM_COORD" user="tim" mqUserId="user1" mqPassword="passw0rd"/>
  <tns:qmgr name="QM_COORD" user="tom" mqUserIdCipher="xh5U7812x"
mqPasswordCipher="e71vKcg2pf"/>
  <tns:qmgr name="QM_COORD" mqUserId="defaultUser" mqPassword="passw0rd"/>
  <tns:qmgr name="QM_COORD" user="ernest" mqUserId="ernest"
mqPassword="AveryL0ngPassw0rd2135" useMQCSPAAuthentication="true"/>
-->

  <complexType name="mqUserPassType">
    <attribute name="name" type="string" use="required"/>
    <attribute name="user" type="string" use="optional"/>
    <attribute name="mqUserId" type="string" use="optional"/>
    <attribute name="mqUserIdCipher" type="string" use="optional"/>
    <attribute name="mqPassword" type="string" use="optional"/>
    <attribute name="mqPasswordCipher" type="string" use="optional"/>
    <attribute name="useMQCSPAAuthentication" type="boolean" use="optional"/>
  </complexType>

</schema>

```

## Understanding the MQMFTCredentials.xml file

The elements and attributes used in the MQMFTCredentials.xml file are described in the following list.

### <mqmftCredentials>

The root element of the XML document.

### <file>

The file in the transfer.

Attribute	Description
path	Path to the key or truststore file being accessed.
password	Password to access the file.

### <logger>

The logger responsible for logging activity.

Attribute	Description
name	The name of the logger.
user	The user name the logger will use to connect to its database.
password	The password the logger will use to connect to its database.

#### <qmgr>

The IBM MQ queue manager connection.

Attribute	Description
name	The name of the associated IBM MQ queue manager.
user	Optional: The name of user requesting the connection.
mqUserId or mqUserIdCipher	The clear text user ID (mqUserId), or obfuscated text user ID (mqUserIdCipher) to supply to an IBM MQ queue manager.
mqPassword or mqPasswordCipher	The clear text password (mqPassword), or obfuscated text password (mqPasswordCipher) to supply to an IBM MQ queue manager.

**Note:** The `MQMFTCredentials.xml` file can contain sensitive information, so when it is created ensure that the file permissions are reviewed. When using a sandbox, set it to be excluded. For more information on sandboxes, see [Working with MFT agent sandboxes](#).

#### Related concepts

[MFT and IBM MQ connection authentication](#)

#### Related tasks

 [Configuring MQMFTCredentials.xml on z/OS](#)

#### Related reference

[“fteObfuscate \(encrypt sensitive data\)” on page 2102](#)

The **fteObfuscate** command encrypts sensitive data in credentials files. This stops the contents of credentials files being read by someone who gains access to the file. From IBM MQ 9.2.0, the command provides a more secure credentials protection method to improve the encryption.

## Additional MFT agent configuration files

In addition to the `agent.properties` file, the Managed File Transfer agent can have a number of XML configuration files in its configuration directory.

### Configuration files

The following XML configuration files can be used to specify additional information used by the agent:

#### **ProtocolBridgeCredentials.xml**

If your agent is a protocol bridge agent, you can use this file to specify the credentials to use to log in to the FTP or SFTP server that the agent connects to.

#### **ProtocolBridgeProperties.xml**

If your agent is a protocol bridge agent, you can use this file to define the properties of non-default protocol file servers that the agent connects to. The **fteCreateBridgeAgent** command creates a default protocol file server in this file for you.

#### **ConnectDirectCredentials.xml**

If your agent is a Connect:Direct bridge agent, you can use this file to specify the credentials to use to connect to the Connect:Direct nodes involved in a transfer.

#### **ConnectDirectNodeProperties.xml**

If your agent is a Connect:Direct bridge agent, you can use this file to specify the operating system information about the Connect:Direct nodes involved in a transfer.

## **ConnectDirectProcessDefinition.xml**

If your agent is a Connect:Direct bridge agent, you can use this file to specify the user-defined Connect:Direct processes to call as part of a file transfer.

## **UserSandboxes.xml**

You can use this file to specify which areas of the file system the agent can read from or write to.

## **Updating the configuration files**

Unlike the `agent.properties` file, you can update the XML configuration files and have the agent pick up the changes without having to restart the agent.

When you submit a transfer, if it has been more than 10 second since the last time the agent checked the XML configuration file, the agent checks the last modified time of the XML configuration file. If the XML configuration file has been modified since the last time the agent read the file, the agent reads the file again. If the contents of the file are valid when compared to the XML schema, the agent updates its information. If the contents of the file are not valid, the agent uses the information from the previous version of the file and writes a message to the `output.log` file.

### **Related concepts**

[Working with MFT user sandboxes](#)

### **Related reference**

[“Protocol bridge credentials file format” on page 2700](#)

The `ProtocolBridgeCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

[“Protocol bridge properties file format” on page 2704](#)

The `ProtocolBridgeProperties.xml` file in the agent configuration directory defines properties for protocol file servers.

[“Connect:Direct credentials file format” on page 2714](#)

The `ConnectDirectCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

[“Connect:Direct node properties file format” on page 2721](#)

The `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory specifies information about remote Connect:Direct nodes that are involved in a file transfer.

[“Connect:Direct process definitions file format” on page 2718](#)

The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

## ***Protocol bridge credentials file format***

The `ProtocolBridgeCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

The `ProtocolBridgeCredentials.xml` file must conform to the `ProtocolBridgeCredentials.xsd` schema. The `ProtocolBridgeCredentials.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MQMFT installation. Users are responsible for manually creating the `ProtocolBridgeCredentials.xml` file, it is no longer created by the **fteCreateBridgeAgent** command. Sample files are available in the `MQ_INSTALLATION_PATH/mqft/samples` directory of the MQMFT installation.

V7.5 introduced a new `<agent>` element that contains the `<server>` or `<serverHost>` element for the named agent.

The `ProtocolBridgeCredentials.xml` file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

## Schema - V7.5 or later

The following schema describes which elements are valid in the ProtocolBridgeCredentials.xml file for V8.

```
<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeCredentials" elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/
ProtocolBridgeCredentials">
  <!--
  <?xml version="1.0" encoding="UTF-8"?>
  <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
  ProtocolBridgeCredentials.xsd">
    <tns:agent name="agent1">
      <tns:serverHost name="myserver">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
        <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
          <tns:privateKey associationName="test" keyPassword="pwd2">
            ... private key ...
          </tns:privateKey>
        </tns:user>
      </tns:serverHost>
    </tns:agent>

    <tns:agent name="agent2">
      <tns:server name="server*" pattern="wildcard">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
        <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
          <tns:privateKey associationName="test" keyPassword="pwd2">
            ... private key ...
          </tns:privateKey>
        </tns:user>
      </tns:server>
    </tns:agent>

    <tns:agent name="agent3">
      <tns:serverHost name="ftpsServer"
        keyStorePassword="keypass"
        trustStorePassword="trustpass">
        <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
      </tns:serverHost>
    </tns:agent>
  </tns:credentials>
  -->
  <element name="credentials" type="tns:credentialsType"/>
  <complexType name="credentialsType">
    <sequence>
      <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="agentType">
    <choice minOccurs="0" maxOccurs="1">
      <element name="serverHost" type="tns:serverHostType" minOccurs="0" maxOccurs="unbounded"/>
      <element name="server" type="tns:serverType" minOccurs="0" maxOccurs="unbounded"/>
    </choice>
    <attribute name="name" type="string" use="required"/>
  </complexType>
  <complexType name="serverHostType">
    <sequence>
      <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="keyStorePassword" type="string" use="optional"/>
    <attribute name="keyStorePasswordCipher" type="string" use="optional"/>
    <attribute name="trustStorePassword" type="string" use="optional"/>
    <attribute name="trustStorePasswordCipher" type="string" use="optional"/>
  </complexType>
  <complexType name="serverType">
    <sequence>
      <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
  </complexType>
```

```

<attribute name="pattern" type="tns:patternType" use="optional" />
<attribute name="keyStorePassword" type="string" use="optional"/>
<attribute name="keyStorePasswordCipher" type="string" use="optional"/>
<attribute name="trustStorePassword" type="string" use="optional"/>
<attribute name="trustStorePasswordCipher" type="string" use="optional"/>
</complexType>

<element name="user" type="tns:userType"/>

<complexType name="userType">
  <sequence>
    <element ref="tns:privateKey" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="serverUserId" type="string" use="optional"/>
  <attribute name="serverUserIdCipher" type="string" use="optional"/>
  <attribute name="serverPassword" type="string" use="optional"/>
  <attribute name="serverPasswordCipher" type="string" use="optional"/>
  <attribute name="hostKey" use="optional">
    <simpleType>
      <restriction base="string">
        <pattern
          value="([a-zA-F0-9]){2}(:([a-zA-F0-9]){2})*" />
        </pattern>
      </restriction>
    </simpleType>
  </attribute>
</complexType>

<element name="privateKey" type="tns:privateKeyType"/>

<complexType name="privateKeyType">
  <simpleContent>
    <extension base="string">
      <attribute name="keyPassword" type="string" use="optional"/>
      <attribute name="keyPasswordCipher" type="string" use="optional"/>
      <attribute name="associationName" type="string" use="required"/>
    </extension>
  </simpleContent>
</complexType>

<!--
-->
Determines the type of pattern matching to use.
-->
<simpleType name="patternType">
  <restriction base="string">
    <enumeration value="regex" />
    <enumeration value="wildcard" />
  </restriction>
</simpleType>
</schema>

```

## Understanding the ProtocolBridgeCredentials.xml file

The elements and attributes used in the ProtocolBridgeCredentials.xml file are described in the following list.

### <credentials>

Group element containing elements that describe the credentials used by a protocol bridge agent to connect to a protocol server.

### <agent>

Element containing a <server> or <serverHost> definition for a named agent.

### <server>

The protocol server that the protocol bridge connects to.

The <server> element is not supported for V7.0.4 or earlier.

Attribute	Description
name	The name of the protocol server.
pattern	If you have used wildcards or regular expressions to specify the pattern of a protocol server name, use either wildcard or regex.

Attribute	Description
trustStorePassword or trustStorePasswordCipher	Required when the <server> element refers to an FTPS server. The password used to access the truststore. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.

#### <serverHost>

The host name of the protocol server that the protocol bridge connects to.

The ProtocolBridgeCredentials.xml file can either contain <serverHost> elements or <server> elements but you cannot use a mixture of the two different types. When you use <serverHost>, the name is matched against the protocol server's host name. When you use <server>, the name is matched against the protocol server's name (as defined in the ProtocolBridgeProperties.xml file).

Attribute	Description
name	The host name or IP address of the protocol server.
trustStorePassword or trustStorePasswordCipher	Required when the <serverHost> element refers to an FTPS server. The password used to access the truststore. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. This property is optional unless you set the keyStore attribute, in which case it is required. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.

#### <user>

A user mapping from a Managed File Transfer user name to a protocol server user name.

Attribute	Description
name	A Java regular expression to match the MQMD user ID associated with the managed transfer request.
serverUserId or serverUserIdCipher	The user name that is used with the protocol server. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.
serverPassword or serverPasswordCipher	The password for the user name used on the protocol server. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.
hostKey	The server host SSH fingerprint.

#### <privateKey>

The private key of a user.

Attribute	Description
keyPassword or keyStorePasswordCipher	The password for the private key. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.

Attribute	Description
associationName	A name used for trace and logging.

### Related tasks

[Mapping credentials for a file server by using the ProtocolBridgeCredentials.xml file](#)

[Defining properties for protocol file servers using the ProtocolBridgeProperties.xml file](#)

### Related reference

[The protocol bridge](#)

[Example: How to configure a bridge agent to use private key credentials with a UNIX SFTP server “fteObfuscate \(encrypt sensitive data\)” on page 2102](#)

The **fteObfuscate** command encrypts sensitive data in credentials files. This stops the contents of credentials files being read by someone who gains access to the file. From IBM MQ 9.2.0, the command provides a more secure credentials protection method to improve the encryption.

### Protocol bridge properties file format

The ProtocolBridgeProperties.xml file in the agent configuration directory defines properties for protocol file servers.

The ProtocolBridgeProperties.xml file must conform to the ProtocolBridgeProperties.xsd schema. The ProtocolBridgeProperties.xsd schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the Managed File Transfer installation. A template file, ProtocolBridgeProperties.xml, is created by the **fteCreateBridgeAgent** command in the agent configuration directory.

The ProtocolBridgeProperties.xml file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property **xmlConfigReloadInterval** in the agent.properties file.

### Schema

The following schema describes the ProtocolBridgeProperties.xml file.

**V 9.3.0** If you do not use the attributes of:

- maxActiveDestinationTransfers (global property)
- maxActiveDestinationTransfers (individual server property)
- failTransferWhenCapacityReached

managed file transfer processing continues in its current form, and the following schema does not change.



**Attention:** Both the source and destination protocol bridge agents must be at IBM MQ 9.2.1 or later, to use the additional attributes.

**V 9.3.0** See [“Changes to the sequence group” on page 2708](#), [“Changes to the limits group” on page 2708](#) for changes these attributes make to the schema, and [Scenarios and examples for limiting the number of file transfers to individual file servers](#) for information on how the changes affect the working of the protocol bridge agent.

```
<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeProperties" elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties">
  <!--
    Example: ProtocolBridgeProperties.xml
  -->
  <?xml version="1.0" encoding="UTF-8"?>
  <tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
      ProtocolBridgeProperties.xsd">
    <tns:credentialsFile path="$HOME/ProtocolBridgeCredentials.xml" />
  </tns:serverProperties>
</schema>
```



```

    <tns:defaultServer name="myserver" />
    <tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234" platform="windows"
        timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
        listFormat="unix" limitedWrite="false" />
    <tns:sftpServer name="server1" host="myhost.hursley.ibm.com" platform="windows"
        fileEncoding="UTF-8" limitedWrite="false">
        <limits maxListFileNames="10" />
    </tns:sftpServer>
</tns:serverProperties>
-->

<!-- Root element for the document -->
<element name="serverProperties" type="tns:serverPropertiesType"></element>

<!--
A container for all protocol bridge server properties
-->
<complexType name="serverPropertiesType">
    <sequence>
        <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1" />
        <element name="defaultServer" type="tns:serverName" minOccurs="0" maxOccurs="1" />
        <choice minOccurs="0" maxOccurs="unbounded">
            <element name="ftpServer" type="tns:ftpServerType" />
            <element name="sftpServer" type="tns:sftpServerType" />
            <element name="ftpsServer" type="tns:ftpsServerType" />
            <element name="ftpsfgServer" type="tns:ftpsfgServerType" />
            <element name="ftpsfgServer" type="tns:ftpsfgServerType" />
        </choice>
    </sequence>
</complexType>

<!--
A container for a server name
-->
<complexType name="serverName">
    <attribute name="name" type="tns:serverNameType" use="required" />
</complexType>

<!--
A container for a credentials file name
-->
<complexType name="credentialsFileName">
    <attribute name="path" type="string" use="required" />
</complexType>

<!--
A container for all the information about an FTP server
-->
<complexType name="ftpServerType">
    <sequence>
        <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
    </sequence>
    <attributeGroup ref="tns:ftpServerAttributes"/>
    <attribute name="passiveMode" type="boolean" use="optional" />
</complexType>

<!--
A container for all the information about an SFG FTP server
-->
<complexType name="ftpsfgServerType">
    <sequence>
        <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
    </sequence>
    <attributeGroup ref="tns:ftpServerAttributes"/>
</complexType>

<!--
A container for all the information about an SFTP server
-->
<complexType name="sftpServerType">
    <sequence>
        <element name="limits" type="tns:sftpLimitsType" minOccurs="0" maxOccurs="1" />
    </sequence>
    <attributeGroup ref="tns:sftpServerAttributes"/>
</complexType>

<!--
A container for all the information about a FTPS server
-->
<complexType name="ftpsServerType">
    <sequence>
        <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
    </sequence>

```

```

    </sequence>
    <attributeGroup ref="tns:ftpsServerAttributes"/>
</complexType>

<!--
  A container for all the information about a SFG FTPS server
-->
<complexType name="ftpsSfgServerType">
  <sequence>
    <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attributeGroup ref="tns:ftpsServerAttributes"/>
</complexType>

<!--
  Attributes common to all server types
-->
<attributeGroup name="generalServerAttributes">
  <attribute name="name" type="tns:serverNameType" use="required" />
  <attribute name="host" type="string" use="required" />
  <attribute name="port" type="nonNegativeInteger" use="optional" />
  <attribute name="platform" type="tns:platformType" use="required" />
  <attribute name="fileEncoding" type="string" use="required" />
  <attribute name="limitedWrite" type="boolean" use="optional" />
  <attribute name="controlEncoding" type="string" use="optional" />
</attributeGroup>

<!--
  Attributes common to ftp and ftps server types
-->
<attributeGroup name="ftpServerAttributes">
  <attributeGroup ref="tns:generalServerAttributes"/>
  <attribute name="timeZone" type="string" use="required" />
  <attribute name="locale" type="tns:localeType" use="required" />
  <attribute name="listFormat" type="tns:listFormatType" use="optional" />
  <attribute name="listFileRecentDateFormat" type="tns:dateFormatType" use="optional" />
  <attribute name="listFileOldDateFormat" type="tns:dateFormatType" use="optional" />
  <attribute name="monthShortNames" type="tns:monthShortNamesType" use="optional" />
</attributeGroup>

<!--
  Attributes common to ftps server types
-->
<attributeGroup name="ftpsServerAttributes">
  <attributeGroup ref="tns:ftpServerAttributes"/>
  <attribute name="ftpsType" type="tns:ftpsTypeType" use="optional" />
  <attribute name="trustStore" type="string" use="required" />
  <attribute name="trustStoreType" type="string" use="optional" />
  <attribute name="keyStore" type="string" use="optional" />
  <attribute name="keyStoreType" type="string" use="optional" />
  <attribute name="ccc" type="boolean" use="optional" />
  <attribute name="protFirst" type="boolean" use="optional" />
  <attribute name="auth" type="string" use="optional" />
  <attribute name="connectTimeout" type="nonNegativeInteger" use="optional"/>
</attributeGroup>

<!--
  A container for limit-type attributes for a server. Limit parameters
  are optional, and if not specified a system default will be used.
-->
<complexType name="generalLimitsType">
  <attributeGroup ref="tns:generalLimitAttributes"/>
</complexType>

<complexType name="sftplimitsType">
  <attributeGroup ref="tns:generalLimitAttributes"/>
  <attribute name="connectionTimeout" type="nonNegativeInteger" use="optional" />
</complexType>

<!--
  Attributes for limits common to all server types
-->
<attributeGroup name="generalLimitAttributes">
  <attribute name="maxListFileNames" type="positiveInteger" use="optional" />
  <attribute name="maxListDirectoryLevels" type="nonNegativeInteger" use="optional" />
  <attribute name="maxReconnectRetry" type="nonNegativeInteger" use="optional" />
  <attribute name="reconnectWaitPeriod" type="nonNegativeInteger" use="optional" />
  <attribute name="maxSessions" type="positiveInteger" use="optional" />
  <attribute name="socketTimeout" type="nonNegativeInteger" use="optional" />
</attributeGroup>

```

```

<!--
and The type for matching valid server names. Server names must be at least 2 characters in length
-->
are limited to alphanumeric characters and the following characters: ".", "_", "/" and "%".
-->
<simpleType name="serverNameType">
  <restriction base="string">
    <pattern value="[0-9a-zA-Z\._/%]{2,}" />
  </restriction>
</simpleType>

<!--
The types of platform supported.
-->
<simpleType name="platformType">
  <restriction base="string">
  </restriction>
</simpleType>

<!--
The type for matching a locale specification.
-->
<simpleType name="localeType">
  <restriction base="string">
    <pattern value="(.)[-_](.)" />
  </restriction>
</simpleType>

<!--
The types of list format supported (for FTP servers).
-->
<simpleType name="listFormatType">
  <restriction base="string">
  </restriction>
</simpleType>

<!--
Date format for FTP client directory listing on an FTP server. This is
the format to be passed to methods setDefaultDateFormatStr and
setRecentDateFormatStr for Java class:
org.apache.commons.net.ftp.FTPClientConfig
-->
<simpleType name="dateFormatType">
  <restriction base="string">
  </restriction>
</simpleType>

<!--
A list of language-defined short month names can be specified. These are
used for translating the directory listing received from the FTP server.
The format is a string of three character month names separated by "|"
-->
<simpleType name="monthShortNamesType">
  <restriction base="string">
    <pattern value="(...\|){11}(...)" />
  </restriction>
</simpleType>

<!--
The enumerations of the allowed FTPS types: "implicit" & "explicit"
If not specified the default is "explicit"
-->
<simpleType name="ftpsTypeType">
  <restriction base="string">
    <enumeration value="explicit" />
    <enumeration value="implicit" />
  </restriction>
</simpleType>

<!--
Attribute Group for SFTP Servers
-->
<attributeGroup name="sftpServerAttributes">
  <attributeGroup ref="tns:generalServerAttributes" />
  <attribute name="cipherList" type="string" use="optional" />
</attributeGroup>
</schema>

```

## Changes to the sequence group

V9.3.0

Following the additions of the **maxActiveDestinationTransfers** (global) and **failTransferWhenCapacityReached** attributes, the sequence group under ComplexType is as follows, with the changes shown in bold text:

```
<!--
  A container for all protocol bridge server properties
-->
<complexType name="serverPropertiesType">
  <sequence>
    <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1" />
    <element name="credentialsKeyFile" type="tns:credentialsKeyFileName" minOccurs="0" maxOccurs="1" />
    <element name="maxActiveDestinationTransfers"
      type="tns:maxActiveDestinationTransfersValue" minOccurs="0"
      maxOccurs="1" />
    <element name="failTransferWhenCapacityReached"
      type="tns:failTransferWhenCapacityReachedValue" minOccurs="0"
      maxOccurs="1" />
    <element name="defaultServer" type="tns:serverName" minOccurs="0" maxOccurs="1" />
    <choice minOccurs="0" maxOccurs="unbounded">
      <element name="ftpServer" type="tns:ftpServerType" />
      <element name="sftpServer" type="tns:sftpServerType" />
      <element name="ftpsServer" type="tns:ftpsServerType" />
      <element name="ftpsfgServer" type="tns:ftpsfgServerType" />
      <element name="ftpsfigServer" type="tns:ftpsfigServerType" />
    </choice>
  </sequence>
</complexType>

<!--
  A container for default value for maxActiveDestinationTransfers
-->
<complexType name="maxActiveDestinationTransfersValue">
  <attribute name="value" type="positiveInteger" use="required" />
</complexType>

<!--
  A container for a boolean value to decide to fail a transfer if max capacity is reached
-->
<complexType name="failTransferWhenCapacityReachedValue">
  <attribute name="value" type="boolean" use="required" />
</complexType>
```

## Changes to the limits group

V9.3.0

Following the addition of the **maxActiveDestinationTransfers** (individual server) attribute, the limits group is as follows, with the changes shown in bold text:

```
<!--
Attributes for limits common to all server types
-->
<attributeGroup name="generalLimitAttributes">
  <attribute name="maxListFileNames" type="positiveInteger"
    use="optional"/>
  <attribute name="maxListDirectoryLevels" type="nonNegativeInteger"
    use="optional"/>
  <attribute name="maxReconnectRetry" type="nonNegativeInteger"
    use="optional" />
  <attribute name="reconnectWaitPeriod" type="nonNegativeInteger"
    use="optional" />
  <attribute name="maxSessions" type="positiveInteger" use="optional" />
  <attribute name="socketTimeout" type="nonNegativeInteger" use="optional"
    />
  <attribute name="connectionTimeout" type="nonNegativeInteger"
    use="optional" />
  <attribute name="maxActiveDestinationTransfers" type="nonNegativeInteger"
```

```
use="optional" />
</attributeGroup>
```

## Example XML file

V9.3.0

```
<tns:serverProperties
  xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
  ProtocolBridgeProperties.xsd">

  <tns:maxActiveDestinationTransfers value="5" />
  <tns:failTransferWhenCapacityReached value="true"/>
  <tns:defaultServer name="guestServer" />

  <tns:sftpServer name="guestServer" host="9.202.177.44" port="22"
  platform="unix" fileEncoding="UTF-8" limitedWrite="false">

  <tns:limits connectionTimeout="60"
  maxActiveDestinationTransfers="2"/>
  </tns:sftpServer>

  <tns:sftpServer name="nixonServer" host="9.199.145.33" port="22"
  platform="unix" fileEncoding="UTF-8" limitedWrite="false">

  <tns:limits connectionTimeout="60"/>
  </tns:sftpServer>

  <tns:sftpServer name="mySFTPserver" host="Harrison.hursley.ibm.com"
  port="22" platform="unix" fileEncoding="UTF-8" limitedWrite="false"
  >

  <tns:limits connectionTimeout="60" />
  </tns:sftpServer>

</tns:serverProperties>
```

## Understanding the ProtocolBridgeProperties.xml file

The elements and attributes that are used in the ProtocolBridgeProperties.xml file are described in the following list:

### <serverProperties>

Root element of the XML document

### <credentialsFile>

Path to the file containing credentials. The value of this property can contain environment variables. For more information, see [Environment variables in MFT properties](#)

### <defaultServer>

The protocol file server that acts as the default server for file transfers

### <ftpServer>

An FTP file server

### <sftpServer>

An SFTP file server

### <ftpsServer>

An FTPS file server

### General server attributes that apply to all types of protocol file server:

Attribute	Description
name	Required. The name of the protocol file server. Protocol server names must be at least two characters in length, are not case-sensitive, and are limited to alphanumeric characters and the following characters:

Attribute	Description
	<ul style="list-style-type: none"> <li>• period (.)</li> <li>• underscore (_)</li> <li>• forward slash (/)</li> <li>• percent sign (%)</li> </ul>
host	Required. The host name or IP address of the protocol file server that you want to send files to or receive files from.
port	Optional. The port number of the protocol file server that you want to send files to or receive files from.
platform	Required. The platform of the protocol file server that you want to send files to or receive files from. Specify either UNIX or WINDOWS. Set this property according to how you enter paths on your FTP, FTPS, or SFTP server. For example, if you are running an FTP server on Windows but when you log in to the server, you must enter UNIX-style paths (that is, with forward slashes), set this value to UNIX and not WINDOWS. Servers running on Windows often present a UNIX-style file system.
fileEncoding	Required. Defines the character encoding that is used by the file server. This property is used when you transfer files in text mode so that the correct encoding sequences are changed when the files are moved between platforms. For example, UTF-8.
limitedWrite	Optional. The default mode when writing to a file server is to create a temporary file and then rename that file when the transfer has completed. For a file server that is configured as write only, the file is created directly with its final name. The value of this property can be true or false. The default is false.
controlEncoding	Optional. The control encoding value for control messages being sent to the protocol file server. This property affects the encoding of the file name that is used and must be compatible with the control encoding of the protocol file server. The default is UTF-8.

**General attributes that apply to FTP and FTPS servers only:**

Attribute	Description
timeZone	Required. The time zone of the protocol file server that you want to send files to or receive files from. For example: America/New_York or Asia/Tokyo.
locale	Required. The language that is used on the protocol file server that you want to send files to or receive files from. For example: en_US or ja_JP
listFormat	Optional. The listing format that defines the format of the file-listed information that is returned from the protocol file server. Use either Windows or UNIX. The default is UNIX.
listFileRecentDateFormat	Optional. The recent date format (less than a year) for FTP client directory listing on an FTP server. This attribute and the <b>listFileOldDateFormat</b> attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.
listFileOldDateFormat	Optional. The old date format (more than a year) for FTP client directory listing on an FTP server. This attribute and the <b>listFileRecentDateFormat</b> attribute allow you to redefine the

Attribute	Description
	expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.
monthShortNames	Optional. A replacement list of month names that are used to decode date information returned from the protocol file server. This property consists of a list of 12 comma-separated names to override the default locale month values. The default is as defined by the protocol file server.

**General attributes that apply to FTP servers only:**

Attribute	Description
passiveMode	Optional. Controls whether the connection to the FTP server is passive or active. If you set the value of this property to <code>false</code> , the connection is active. If you set the value to <code>true</code> , the connection is passive. The default is <code>false</code> .

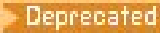





**General attributes that apply to FTPS servers only:**

Attribute	Description
ftpsType	Optional. Specifies whether the explicit or implicit form of the FTPS protocol is used. The default is <code>explicit</code> .
trustStore	Required. The location of the truststore that is used to determine whether the certificate presented by the FTPS server is trusted.
trustStoreType	Optional. The format of the truststore file. The default is <code>JKS</code> .
keyStore	Optional. The location of the keystore that is used to provide certificate information if challenged by the FTPS server. The default is for the protocol bridge to not be able to connect to FTPS servers that are configured to require the authentication of clients.
keyStoreType	Optional. The format of the keystore file. The default is <code>JKS</code> .
ccc	Optional. Selects whether a clear (unencrypted) command channel is used when authentication has completed. The default value is <code>false</code> , which means that the command channel remains encrypted for the entire duration of the FTPS session. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .
protFirst	Optional. Specifies whether the <b>USER/PASS</b> commands are issued to the FTPS server before or after the <b>PBSZ/PROT</b> commands. The default value is <code>false</code> , which means <b>USER/PASS</b> commands are sent first followed by <b>PBSZ/PROT</b> commands. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .
auth	Optional. Specifies the protocol that is specified as part of the <b>AUTH</b> command. A specified protocol will be tried first, then the default is to try <code>TLS</code> , <code>SSL</code> , <code>TLS-C</code> , or <code>TLS-P</code> until the FTPS server does not reject with a 504 reply code. This attribute is applicable only when the <code>ftpsType</code> is set to <code>explicit</code> .

**<limits>**

Container element for attributes that are common to all types of server and for attributes that are specific to a type of server:

**General limit attributes that apply to all types of protocol file server:**

Attribute	Description
maxListFileNames	Optional. The maximum number of names that are collected when scanning a directory on the protocol file server for file names. The default is 999999999.
maxListDirectoryLevels	Optional. The maximum number of directory levels on the protocol server to recursively scan for file names. The default is 1000.
 maxReconnectRetry	Optional. The maximum number of times a protocol server tries to reconnect before the protocol bridge agent stops trying. The default is 2.
 reconnectWaitPeriod	Optional. The time period, in seconds, to wait to before attempting to reconnect. The default is 10 seconds.
maxSessions	Optional. The maximum number of sessions for the protocol server. This number must be greater than or equal to the sum of the maximum number of source and destination transfers for the protocol bridge agent. The default is the sum of the values for the agent properties maxSourceTransfers, maxDestinationTransfers, and maxCommandHandlerThreads, plus 1. If these three properties are using their default values of 25, 25, and 5, the maxSessions default is then 56.
socketTimeout	Optional. The socket timeout in seconds. The value of this attribute is used during file streaming. The default is 30 seconds.
 maxActiveDestinationTransfers (global property)	Optional. Used to specify the global value, to limit the number of active transfers for each destination *ftp* endpoint. This is a non-zero positive integer with a minimum value of 0 and a maximum value of 1.  maxActiveDestinationTransfers requires the use of an AttributeName.
 maxActiveDestinationTransfers (individual server level)	Optional. Used to specify the limit of the number of active transfers for each destination *ftp* endpoint. This is a non-negative integer.  This property can be utilized by any of the three servers, and if specified, this value overrides the global value of maxActiveDestinationTransfers for that endpoint server.  The value of this property cannot exceed the value of maxDestinationTransfers. If you exceed this value, the protocol bridge agent assumes that this value is not set and processes the managed transfers in the standard existing flow. Message BFGSS0088W is logged in the output0.log file.   <b>Attention:</b> It is possible to over commit the new maxActiveDestinationTransfers properties. That is, you can have the sum of maxActiveDestinationTransfers for all the endpoints greater than the value of maxDestinationTransfers. You must consider whether this feature is appropriate for your enterprise.
 failTransferWhenCapacityReached (global property)	Optional. This is a non-zero positive integer with a minimum value of 0 and a maximum value of 1.



Attribute	Description
	<p>failTransferWhenCapacityReached requires the use of an AttributeName.</p> <p>Applies to both maxDestinationTransfers and maxActiveDestinationTransfers and can be used to specify whether to fail a protocol bridge agent transfer in the following cases:</p> <ul style="list-style-type: none"> <li>When the total number of active transfers for an endpoint server exceeds the maxDestinationTransfers count, the two possible conditions are: <ul style="list-style-type: none"> <li><b>failTransferWhenCapacityReached = false</b> Takes the standard existing route of handling the managed transfers.</li> <li><b>failTransferWhenCapacityReached = true</b> Fails the transfer if the total number of active transfers is greater than maxActiveTransfers</li> </ul> </li> <li>When the total number of active transfers for an endpoint server exceeds the maxActiveDestinationTransfers count, the two possible values are: <ul style="list-style-type: none"> <li><b>failTransferWhenCapacityReached = false</b> The default value which applies if maxActiveDestinationTransfers is not set.  Once the number of active transfers for an endpoint server exceeds the maxActiveDestinationTransfers value, the next managed transfer to that particular endpoint server is rejected and moved to a new state called WaitingForDestinationFileServerCapacity by the protocol bridge agent.  Source agents then handle this state in the same way as they currently do for managed transfers that go into a WaitingForDestinationCapacity state; that is, wait for a period of time before contacting the destination agent again.</li> <li><b>failTransferWhenCapacityReached = true</b> Once the number of active transfers for an endpoint server exceeds maxActiveDestinationTransfers value, the next managed transfer to that particular endpoint server is rejected and marked as failed by the protocol bridge agent.</li> </ul> </li> </ul>

**Limit attribute that applies to SFTP servers only:**

Attribute	Description
connectionTimeout	Optional. The time, in seconds, to wait for a response from the protocol file server to a connection request. A timeout indicates that the protocol file server is not available. The default value is 30 seconds.
cipherList	Optional. Specifies a comma-separated list of ciphers that are used to communicate between the protocol bridge agent and the SFTP server. The ciphers are called in the order that they are specified in this list. The cipher must be available on the server and the client before it can be used.

Attribute	Description
	<p>The ciphers that the protocol bridge agent supports are as follows:</p> <ul style="list-style-type: none"> <li>• blowfish-cbc</li> <li>• 3des-cbc</li> <li>• aes128-cbc</li> <li>• aes192-cbc</li> <li>• aes256-cbc</li> <li>• aes128-ctr</li> <li>• aes192-ctr</li> <li>• aes256-ctr</li> <li>• 3des-ctr</li> <li>• arcfour</li> <li>• arcfour128</li> <li>• arcfour256</li> </ul> <p>By default, the list of ciphers used by protocol bridge agents is aes128-cbc , aes192-cbc , aes256-cbc.</p>

### Related tasks

[Defining properties for protocol file servers using the ProtocolBridgeProperties.xml file](#)

[Mapping credentials for a file server by using the ProtocolBridgeCredentials.xml file](#)

### Related reference

[The protocol bridge](#)

[Example: How to configure a bridge agent to use private key credentials with a UNIX SFTP server](#)

[Environment variables in MFT properties](#)

### **Connect:Direct credentials file format**

The `ConnectDirectCredentials.xml` file in the Managed File Transfer Agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

The `ConnectDirectCredentials.xml` file must conform to the `ConnectDirectCredentials.xsd` schema. The `ConnectDirectCredentials.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MQMFT installation. A sample `ConnectDirectCredentials.xml` file is located in the `MQ_INSTALLATION_PATH/mqft/samples/credentials` directory of the MQMFT installation.

The file `ConnectDirectCredentials.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

### Schema

The following schema describes which elements are valid in the `ConnectDirectCredentials.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  This schema defines the format of the XML file that is located in the agent properties
  directory of a Connect:Direct bridge agent. The XML file ConnectDirectCredentials.xml
  is used by the default credential validation of the Connect:Direct bridge.
  For more information, see the WebSphere MQ InfoCenter
-->
```

```

<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectCredentials"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"

  <!--
  <?xml version="1.0" encoding="UTF-8"?>

  <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials
      ConnectDirectCredentials.xsd">
    <tns:agent name="CDAGENT01">
      <tns:pnode name="cdnode*" pattern="wildcard">
        <tns:user name="MUSR_.*"
          ignorecase="true"
          pattern="regex"
          cdUserId="bob"
          cdPassword="password"
          pnodeUserId="bill"
          pnodePassword="alacazam">
          <tns:snode name="cdnode2" pattern="wildcard" userId="sue" password="foo"/>
        </tns:user>
      </tns:pnode>
    </tns:agent>
  </tns:credentials>

  -->

  <element name="credentials" type="tns:credentialsType"/>

  <complexType name="credentialsType">
    <sequence>
      <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="agentType">
    <sequence>
      <element name="pnode" type="tns:pnodeType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
  </complexType>

  <complexType name="pnodeType">
    <sequence>
      <element name="user" type="tns:userType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
  </complexType>

  <complexType name="userType">
    <sequence>
      <element name="snode" type="tns:snodeType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="ignorecase" type="boolean" use="optional"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
    <attribute name="cdUserId" type="string" use="optional"/>
    <attribute name="cdUserIdCipher" type="string" use="optional"/>
    <attribute name="cdPassword" type="string" use="optional"/>
    <attribute name="cdPasswordCipher" type="string" use="optional"/>
    <attribute name="pnodeUserId" type="string" use="optional"/>
    <attribute name="pnodeUserIdCipher" type="string" use="optional"/>
    <attribute name="pnodePassword" type="string" use="optional"/>
    <attribute name="pnodePasswordCipher" type="string" use="optional"/>
  </complexType>

  <complexType name="snodeType">
    <attribute name="name" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
    <attribute name="userId" type="string" use="optional"/>
    <attribute name="userIdCipher" type="string" use="optional"/>
    <attribute name="password" type="string" use="optional"/>
    <attribute name="passwordCipher" type="string" use="optional"/>
  </complexType>

  <simpleType name="patternType">
    <restriction base="string">

```

```

    <enumeration value="regex"/>
    <enumeration value="wildcard"/>
  </restriction>
</simpleType>
</schema>

```

## Understanding the ConnectDirectCredentials.xml file

The elements and attributes used in the ConnectDirectCredentials.xml file are described in the following list.

### <credentials>

Group element containing elements that describe the credentials used by a Connect:Direct bridge agent to connect to a Connect:Direct node.

### <agent>

Group element containing elements for <pnode> definitions for a named agent.

### <pnode>

The primary node (PNODE) in the Connect:Direct transfer. This node initiates the connection to the secondary node (SNODE).

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> <li>wildcard - wildcards are used</li> <li>regex - Java regular expressions are used</li> </ul>

### <user>

The IBM MQ user that submits the transfer request.

Attribute	Description
name	The user name that is used with Managed File Transfer. The value of this attribute can be a pattern that matches many user names.
ignorecase	Specifies whether the case of the name is ignored. Valid values for the ignorecase attribute are <ul style="list-style-type: none"> <li>true - the name is not case sensitive</li> <li>false - the name is case sensitive</li> </ul>
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> <li>wildcard - wildcards are used</li> <li>regex - Java regular expressions are used</li> </ul>
cdUserId or cdUserIdCipher	The user name that is used by the Connect:Direct bridge to connect to its associated Connect:Direct node. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.
cdPassword or cdPasswordCipher	The password associated with the user name specified by the cdUserId attribute. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.
pnodeUserId or pnodeUserIdCipher	The user name that is used by the Connect:Direct primary node. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.

Attribute	Description
pnodePassword or pnodePasswordCipher	The password associated with the user name specified by the pnodeUserId attribute. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.

#### <snode>

The Connect:Direct node that performs the role of secondary node (SNODE) during the Connect:Direct file transfer.

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> <li>wildcard - wildcards are used</li> <li>regex - Java regular expressions are used</li> </ul>
userId or userIdCipher	The user name used to connect to this node during a file transfer. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.
password or passwordCipher	The password associated with the user name specified by the userId attribute. If the <b>fteObfuscate</b> command has been used then the cipher version of the attribute must be used.

#### Example

In this example, the Connect:Direct bridge agent connects to the Connect:Direct node pnode1. When an IBM MQ user with the user name beginning with the prefix fteuser followed by a single character, for example fteuser2, requests a transfer involving the Connect:Direct bridge, the Connect:Direct bridge agent will use the user name cduser and the password passw0rd to connect to the Connect:Direct node pnode1. When the Connect:Direct node pnode1 performs its part of the transfer it uses the user name pnodeuser and the password passw0rd1.

If the secondary node in the Connect:Direct transfer has a name that begins with the prefix FISH, the node pnode1 uses the user name fishuser and the password passw0rd2 to connect to the secondary node. If the secondary node in the Connect:Direct transfer has a name that begins with the prefix CHIPS, the node pnode1 uses the user name chipsuser and the password passw0rd3 to connect to the secondary node.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials
ConnectDirectCredentials.xsd">
  <tns:agent name="CDAGENT01">
    <tns:pnode name="pnode1" pattern="wildcard">
      <tns:user name="fteuser?" pattern="wildcard" ignorecase="true"
        cdUserId="cduser" cdPassword="passw0rd"
        pnodeUserId="pnodeuser" pnodePassword="passw0rd1">
      <tns:snode name="FISH*" pattern="wildcard"
        userId="fishuser" password="passw0rd2"/>
      <tns:snode name="CHIPS*" pattern="wildcard"
        userId="chipsuser" password="passw0rd3"/>
    </tns:user>
  </tns:pnode>
</tns:agent>
</tns:credentials>
```

## Related reference

[“fteCreateCDAgent \(create a Connect:Direct bridge agent\)” on page 2012](#)

The fteCreateCDAgent command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

[The Connect:Direct bridge](#)

[“Regular expressions used by MFT” on page 2523](#)

Managed File Transfer uses regular expressions in a number of scenarios. For example, regular expressions are used to match user IDs for Connect:Direct security credentials, or to split a file into multiple messages by creating a new message each time a regular expression is matched. The regular expression syntax used by Managed File Transfer is the syntax supported by the `java.util.regex` API. This regular expression syntax is similar to, but not the same as, the regular expression syntax used by the Perl language.

## Connect:Direct process definitions file format

The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

The `ConnectDirectProcessDefinitions.xml` file must conform to the `ConnectDirectProcessDefinitions.xsd` schema. The `ConnectDirectProcessDefinitions.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MFT installation. A template `ConnectDirectProcessDefinitions.xml` file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file `ConnectDirectProcessDefinitions.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

## Schema

The following schema describes which elements are valid in the `ConnectDirectProcessDefinitions.xml` file.

```
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions">

  <element name="cdprocess" type="tns:cdprocessType"></element>

  <complexType name="cdprocessType">
    <sequence>
      <element name="processSet" type="tns:processSetType"
        minOccurs="0" maxOccurs="unbounded"></element>
    </sequence>
  </complexType>

  <complexType name="processSetType">
    <sequence>
      <element name="condition" type="tns:conditionType"
        minOccurs="0" maxOccurs="1" />
      <element name="process" type="tns:processType"
        minOccurs="1" maxOccurs="1" />
    </sequence>
  </complexType>

  <complexType name="conditionType">
    <choice minOccurs="0" maxOccurs="unbounded">
      <element name="match" type="tns:matchType" />
      <element name="defined" type="tns:definedType" />
    </choice>
  </complexType>

  <complexType name="matchType">
    <attribute name="variable" type="string" use="required" />
    <attribute name="value" type="string" use="required" />
  </complexType>
</schema>
```

```

    <attribute name="pattern" type="tns:patternType" use="optional" />
  </complexType>

  <complexType name="definedType">
    <attribute name="variable" type="string" use="required" />
  </complexType>

  <complexType name="processType">
    <sequence>
      <element name="preTransfer" type="tns:transferType"
        minOccurs="0" maxOccurs="1" />
      <element name="transfer" type="tns:transferType"
        minOccurs="0" maxOccurs="1" />
      <element name="postTransferSuccess" type="tns:transferType"
        minOccurs="0" maxOccurs="1" />
      <element name="postTransferFailure" type="tns:transferType"
        minOccurs="0" maxOccurs="1" />
    </sequence>
  </complexType>

  <complexType name="transferType">
    <attribute name="process" type="string" use="required" />
  </complexType>

  <simpleType name="patternType">
    <restriction base="string">
      <enumeration value="regex" />
      <enumeration value="wildcard" />
    </restriction>
  </simpleType>
</schema>

```

## Understanding the ConnectDirectProcessDefinitions.xml file

The elements and attributes used in the ConnectDirectProcessDefinitions.xml file are described in the following list.

### cdProcess

The root element of the XML document.

### processSet

Group element containing all the information about a set of user-defined processes.

### condition

Group element containing the conditions that a transfer is tested against to determine whether the set of processes contained in the processSet element are used.

### match

A condition that tests whether a the value of a variable matches a given value.

Attribute	Description
variable	Specifies a variable. The value of this variable is compared with the value of the value attribute. The variable is an intrinsic symbol. For more information, see <a href="#">“Substitution variables for use with user-defined Connect:Direct processes”</a> on page 2523.
value	Specifies a pattern to match against the value of the variable specified by the variable attribute.
pattern	Specifies the type of pattern that is used for the value of the value attribute. Valid values for the pattern attribute are <ul style="list-style-type: none"> <li>wildcard - wildcards are used</li> <li>regex - Java regular expressions are used</li> </ul> This attribute is optional and the default is wildcard.

### defined

A condition that tests whether a variable has been defined.

Attribute	Description
variable	Specifies a variable. If this variable exists, the match condition is satisfied. The variable is an intrinsic symbol. For more information, see <a href="#">“Substitution variables for use with user-defined Connect:Direct processes”</a> on page 2523.

### process

Group element containing the information about where to locate the Connect:Direct processes to call when a match is found.

### transfer

The Connect:Direct process to call during a transfer request.

Attribute	Description
process	Optional. Specifies the name of a file that contains a Connect:Direct process to call during a transfer request. The file path is relative to the Connect:Direct bridge agent configuration directory. This attribute is optional, the default is to use a process generated by MFT. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables. For more information, see <a href="#">Environment variables in MFT properties</a>

### Example

In this example, there are three processSet elements.

The first processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*` and a **%FTESUSER** variable with a value of `Admin`, the Connect:Direct bridge agent submits the Connect:Direct process located in the `agent_configuration_directory/AdminClient.cdp` as part of the transfer.

The second processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*`, the Connect:Direct bridge agent submits the Connect:Direct process located in the `agent_configuration_directory/Client.cdp` as part of the transfer. The Connect:Direct bridge agent reads the processSet elements in the order that they are defined, and if it finds a match, it uses the first match and does not look for another match. For transfer requests that match the conditions of both the first and second processSet, the Connect:Direct bridge agent calls only the processes specified by the first processSet.

The third processSet element has no conditions and matches all transfers. If the transfer request does not match the conditions of the first or second processSet, the Connect:Direct bridge agent submits the Connect:Direct process specified by the third condition. This process is located in the `agent_configuration_directory/Default.cdp` as part of the transfer.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions
ConnectDirectProcessDefinitions.xsd">

  <tns:processSet>
    <tns:condition>
      <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard" />
      <tns:match variable="%FTESUSER" value="Admin" pattern="wildcard" />
    </tns:condition>
    <tns:process>
      <tns:transfer process="AdminClient.cdp" />
    </tns:process>
  </tns:processSet>

  <tns:processSet>
    <tns:condition>
      <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard" />
    </tns:condition>
    <tns:process>
      <tns:transfer process="Client.cdp" />
    </tns:process>
  </tns:processSet>
</tns:cdprocess>
```



```

</tns:processSet>

<tns:processSet>
  <tns:process>
    <tns:transfer process="Default.cdp" />
  </tns:process>
</tns:processSet>

</tns:cdprocess>

```

### Related tasks

[Specifying the Connect:Direct process to start by using the ConnectDirectProcessDefinition.xml file](#)

### Related reference

[“fteCreateCDAgent \(create a Connect:Direct bridge agent\)” on page 2012](#)

The fteCreateCDAgent command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

[The Connect:Direct bridge](#)

[“Regular expressions used by MFT” on page 2523](#)

Managed File Transfer uses regular expressions in a number of scenarios. For example, regular expressions are used to match user IDs for Connect:Direct security credentials, or to split a file into multiple messages by creating a new message each time a regular expression is matched. The regular expression syntax used by Managed File Transfer is the syntax supported by the `java.util.regex` API. This regular expression syntax is similar to, but not the same as, the regular expression syntax used by the Perl language.

[Environment variables in MFT properties](#)

### **Connect:Direct node properties file format**

The `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory specifies information about remote Connect:Direct nodes that are involved in a file transfer.

The `ConnectDirectNodeProperties.xml` file must conform to the `ConnectDirectNodeProperties.xsd` schema. The `ConnectDirectNodeProperties.xsd` schema document is located in the `MQ_INSTALLATION_PATH/mqft/samples/schema` directory of the MFT installation. A template `ConnectDirectNodeProperties.xml` file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file `ConnectDirectNodeProperties.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

### Schema

The following schema describes which elements are valid in the `ConnectDirectNodeProperties.xml` file.

```

<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
  elementFormDefault="qualified"
  xmlns="https://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties">
  <element name="nodeProperties" type="tns:nodePropertiesType"></element>
  <complexType name="nodePropertiesType">
    <sequence>
      <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1" />
      <element name="node" type="tns:nodeType" minOccurs="0" maxOccurs="unbounded"></element>
    </sequence>
  </complexType>
  <complexType name="nodeType">
    <attribute name="name" type="string" use="required" />
    <attribute name="pattern" type="tns:patternType" use="optional" />
    <attribute name="type" type="string" use="required" />
  </complexType>

```

```

</complexType>

<simpleType name="patternType">
  <restriction base="string">
    <enumeration value="regex" />
    <enumeration value="wildcard" />
  </restriction>
</simpleType>

</schema>

```

## Understanding the ConnectDirectNodeProperties.xml file

The elements and attributes used in the ConnectDirectNodeProperties.xml file are described in the following list.

### nodeProperties


Root element of the XML document.

### credentialsFile

Path to the credentials file where sensitive information is stored. For IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables. For more information, see [Environment variables in MFT properties](#)

### node

Specifies one or more Connect:Direct nodes.

Attribute	Description
name	A pattern that identifies the names of Connect:Direct nodes that use the definitions specified by the node element. Pattern matching is not case sensitive.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are: <ul style="list-style-type: none"> <li>wildcard - wildcards are used</li> <li>regex - Java regular expressions are used</li> </ul> For information about the types of regular expressions used by MFT, see <a href="#">“Regular expressions used by MFT” on page 2523</a> .
type	Specifies the operating system type of the Connect:Direct node or nodes that match the pattern given by the name attribute. Valid values for the type attribute are: <ul style="list-style-type: none"> <li>Windows - the node runs on Windows</li> <li>UNIX - the node runs on AIX and Linux</li> <li> z/OS, zos, os/390, or os390 - the node runs on z/OS</li> </ul> The value of this attribute is not case sensitive.

### Example

In this example, the Connect:Direct credentials filename is specified as ConnectDirectCredentials.xml. The example code specifies the following platform connections:

- All Connect:Direct nodes that have a name that begins with "cdnodew" run on the Windows platform.
- All Connect:Direct nodes that have a name that begins with "cdnodeu" run on the AIX and Linux platforms.
- All Connect:Direct nodes that have a name that begins with "cdnodez" run on the z/OS platform.
- All other Connect:Direct nodes run on the AIX and Linux platforms.

The Connect:Direct bridge agent searches for matches from the start of the file to the end, and uses the first match that it finds.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:nodeProperties xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectNodeProperties
    ConnectDirectNodeProperties.xsd">

  <tns:credentialsFile path="ConnectDirectCredentials.xml" />
  <tns:node name="cdnodew*" pattern="wildcard" type="windows" />
  <tns:node name="cdnodeu.*" pattern="regex" type="unix" />
  <tns:node name="cdnodez*" pattern="wildcard" type="zos" />
  <tns:node name="*" pattern="wildcard" type="unix" />

</tns:nodeProperties>
```

### Related reference

[“fteCreateCDAgent \(create a Connect:Direct bridge agent\)” on page 2012](#)

The fteCreateCDAgent command creates a Managed File Transfer Agent and its associated configuration for use with the Connect:Direct bridge.

[The Connect:Direct bridge](#)

[“Regular expressions used by MFT” on page 2523](#)

Managed File Transfer uses regular expressions in a number of scenarios. For example, regular expressions are used to match user IDs for Connect:Direct security credentials, or to split a file into multiple messages by creating a new message each time a regular expression is matched. The regular expression syntax used by Managed File Transfer is the syntax supported by the `java.util.regex` API. This regular expression syntax is similar to, but not the same as, the regular expression syntax used by the Perl language.

[Environment variables in MFT properties](#)

## fteutils.xsd schema file

This schema defines elements and types used by many of the other Managed File Transfer schemas.

### Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]

Licensed Materials - Property of IBM

5724-H72

Copyright IBM Corp. 2008, 2024. All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
@end_non_restricted_prolog@
-->

<!--
This schema defines elements and types used by many of the other MQMFT schemas.
For more information about MQMFT XML message formats, see
https://www.ibm.com/docs/SSEP7X_7.0.4/com.ibm.wmqfte.doc/message_formats.htm
-->
<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">
  <!--
  Defines the version type 1.00 - 99.00
  <transaction version= 1.00
  -->
  <xsd:simpleType name="versionType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9]+\.[0-9][0-9]" />
    </xsd:restriction>
  </xsd:simpleType>
```

```

<!--
  Defines the transaction reference
  <transaction version= 1.00 ID="414d5120514d31202020202020202020205ecf0a4920011802"
-->
<xsd:simpleType name="IDType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9a-fA-F]{48}" />
  </xsd:restriction>
</xsd:simpleType>
<!--
  This is an alias for hostUserIDType.
  Here to allow addition of attributes on originator elements
-->
<xsd:complexType name="origRequestType">
  <xsd:complexContent>
    <xsd:extension base="hostUserIDType">
      <xsd:sequence>
        <xsd:element name="webBrowser" type="xsd:string" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="webUserID" type="xsd:string" minOccurs="0"
maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!--
  Defines a Delete originator as a machine and user pair
  <hostName>myMachine</hostName>
  <userName>myUserId</userName>
-->
<xsd:complexType name="origDeleteType">
  <xsd:sequence>
    <xsd:element name="delete" type="hostUserIDType" maxOccurs="1" minOccurs="0" />
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines a machine, user, MQMD userID triple
  <hostName>myMachine</hostName>
  <userID>myUserId</userID>
  <mqmdUserID>MQMDUSERID</mqmdUserID>
-->
<xsd:complexType name="hostUserIDType">
  <xsd:sequence>
    <xsd:element name="hostName" type="xsd:string" minOccurs="0" maxOccurs="1" />
    <xsd:element name="userID" type="xsd:string" minOccurs="0" maxOccurs="1" />
    <xsd:element name="mqmdUserID" type="xsd:string" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
<!--
  Define the destinationAgent with agent and queue manager name
  <destinationAgent agent="there" QMgr="far" agentType="BRIDGE" bridgeURL="ftp://
server.example.net:21" bridgeNode="DEST_NODE" />
  optional agentType attribute expected to be one of STANDARD, BRIDGE, WEB_GATEWAY,
EMBEDDED, CD_BRIDGE
-->
<xsd:complexType name="agentType">
  <xsd:attribute name="agent" type="xsd:string" use="required" />
  <xsd:attribute name="agentType" type="xsd:string" use="optional" />
  <xsd:attribute name="QMgr" type="xsd:string" use="optional" />
  <xsd:attribute name="bridgeURL" type="xsd:string" use="optional" />
  <xsd:attribute name="bridgeNode" type="xsd:string" use="optional" />
  <xsd:attribute name="pnode" type="xsd:string" use="optional" />
  <xsd:attribute name="snode" type="xsd:string" use="optional" />
</xsd:complexType>
<!--
  Defines the status type; attr/resultCode and 0 or many supplements
  There may also be additional command specific data, either: transfer, ping or call data
  <status resultCode="8011">
    <supplement>Azionamento del USB</supplement>
    <supplement>morto come norweign azzurro</supplement>
  </status>
-->
<xsd:complexType name="statusType">
  <xsd:sequence>
    <xsd:element name="supplement" type="xsd:string" maxOccurs="unbounded"
minOccurs="0" />
    <xsd:choice>
      <xsd:element name="filesystem" type="fileSpaceReplyType" minOccurs="0"
maxOccurs="1" />
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="resultCode" type="resultCodeType" use="required" />

```

```

</xsd:complexType>
<!--
  Defines the filespace type for use with communication between a web agent
  and a web gateway
  <filespace name="" location=""><Quota bytes=""></filespace>
-->
<xsd:complexType name="fileSpaceReplyType">
  <xsd:attribute name="name" use="required" type="xsd:string"/>
  <xsd:attribute name="location" use="required" type="xsd:string"/>
  <xsd:attribute name="quota" use="required" type="xsd:long"/>
</xsd:complexType>
<!--
  Defines the destinationAgent with agent and queue manager name, plus connection
  details.
  <destinationAgent agent="there" QMgr="far"/>
-->
<xsd:complexType name="agentClientType">
  <xsd:attribute name="agent" type="xsd:string" use="required"/>
  <xsd:attribute name="QMgr" type="xsd:string" use="optional"/>
  <xsd:attribute name="hostName" type="xsd:string" use="optional"/>
  <xsd:attribute name="portNumber" type="xsd:nonNegativeInteger" use="optional"/>
  <xsd:attribute name="channel" type="xsd:string" use="optional"/>
</xsd:complexType>
<!--
  Defines the fileURI type as string
  <file encoding="UTF8" EOL="CR">C:/from/here.txt</file>
-->
<xsd:complexType name="fileType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="filespace" type="filespaceNameType" use="optional"/>
      <xsd:attribute name="alias" type="xsd:string" use="optional"/>
      <xsd:attribute name="encoding" type="encodingType" use="optional"/>
      <xsd:attribute name="EOL" type="EOLType" use="optional"/>
      <xsd:attribute name="size" type="xsd:long" use="optional"/>
      <xsd:attribute name="last-modified" type="xsd:dateTime" use="optional"/>
      <xsd:attribute name="delimiter" type="xsd:string" use="optional"/>
      <xsd:attribute name="delimiterType" type="xsd:string" use="optional"/>
      <xsd:attribute name="delimiterPosition" type="delimiterPositionType"
use="optional"/>
      <xsd:attribute name="includeDelimiterInFile" type="xsd:boolean" use="optional"/>
      <xsd:attribute name="keepTrailingSpaces" type="xsd:boolean" use="optional"/>
      <xsd:attribute name="truncateRecords" type="xsd:boolean" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  Defines the filespace type as string
  <filespace>
    <name>tarquin</name>
  </filespace>
-->
<xsd:complexType name="filespaceType">
  <xsd:sequence>
    <xsd:element name="name" type="filespaceNameType"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines a name element
  <name>bob</name>
-->
<xsd:simpleType name="filespaceNameType">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<!--
  Defines the accepted choices for the persistent attribute.
-->
<xsd:simpleType name="persistenceType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="true"/>
    <xsd:enumeration value="false"/>
    <xsd:enumeration value="qdef"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the queueURI type as string with all supported attributes.
  <queue>QUEUE@QM</queue>
-->
<xsd:complexType name="queueType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="persistent" type="persistenceType" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

```



```

    Defines the enumeration values for source type
    type="file|queue"
-->
<xsd:simpleType name="SourceType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="file"/>
    <xsd:enumeration value="directory"/>
    <xsd:enumeration value="queue"/>
    <xsd:enumeration value="dataset"/>
    <xsd:enumeration value="pds"/>
    <xsd:enumeration value="filespace"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumeration values for source disposition
    disposition="leave|delete"
-->
<xsd:simpleType name="sourceDispositionType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="leave"/>
    <xsd:enumeration value="delete"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
    Defines a single destination file reference
    <destination type="file" exist="overwrite">
      <file>filename</file>
    </destination>
-->
<xsd:complexType name="fileDestinationType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="file" type="fileType"/>
      <xsd:element name="filespace" type="filespaceType"/>
      <xsd:element name="queue" type="queueType"/>
    </xsd:choice>
    <xsd:element name="attributes" type="attributeType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="type" type="DestinationType" use="optional"/>
  <xsd:attribute name="exist" type="existType" use="optional"/>
  <xsd:attribute name="correlationString1" type="xsd:string" use="optional"/>
  <xsd:attribute name="correlationNum1" type="xsd:nonNegativeInteger" use="optional"/>
  <xsd:attribute name="correlationBoolean1" type="xsd:boolean" use="optional"/>
</xsd:complexType>
<!--
    Defines the enumeration values for destination file type
    type="file|directory|queue|dataset|pds|filespace"
    'dataset' and 'pds' only apply to z/OS environments.
-->
<xsd:simpleType name="DestinationType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="file"/>
    <xsd:enumeration value="directory"/>
    <xsd:enumeration value="queue"/>
    <xsd:enumeration value="dataset"/>
    <xsd:enumeration value="pds"/>
    <xsd:enumeration value="filespace"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumerations values for file exists on destination behavior
    exist="error|overwrite"
-->
<xsd:simpleType name="existType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="error"/>
    <xsd:enumeration value="overwrite"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
    Defines one or more file attributes
    <destination encoding=? CFLF=?>
      <file>filename</file>
      <attributes>
        <attribute>DIST(MIRRORED,UPDATE)</attribute>
      </attributes>
    </destination>
-->
<xsd:complexType name="attributeType">
  <xsd:sequence>
    <xsd:element name="attribute" type="xsd:string" maxOccurs="unbounded"

```

```

minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!--
  Defines a single file reference
  <source encodings=? CFLF=?>
    <file>filename</file>
    <checksum method="MD5">3445678</checksum>
  </source>
  .. OR ..
  <destination encoding=? CFLF=?>
    <file>filename</file>
    <checksum method="MD5">3445678</checksum>
  </destination>
-->
<xsd:complexType name="fileChecksumType">
  <xsd:sequence>
    <xsd:element name="file" type="fileType"/>
    <xsd:element name="checksum" type="checksumType" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines the checksum type and method
  <checksum method="MD5|none">3445678</checksum>
-->
<xsd:complexType name="checksumType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="method" type="checkSumMethod" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  Defines the enumeration values for checksumMethod
  <checksum method="MD5|none">3445678</checksum>
  Note: uppercase is used because MD5 is an acronym and normally written uppercase.
-->
<xsd:simpleType name="checkSumMethod">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="MD5"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the enumeration values for agentRole
  agentRole="sourceAgent|destinationAgent"
-->
<xsd:simpleType name="agentRoleType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="sourceAgent"/>
    <xsd:enumeration value="destinationAgent"/>
    <xsd:enumeration value="callAgent"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the enumeration values for mode.
  text, binary or a substitution variable
  <item mode="binary|text|&#x27;${variableName}&#x27;">
-->
<xsd:simpleType name="modeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="binary|text|&#x27;.*&#x27;" />
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the enumeration values for EOL
  <file EOL="LF|CRLF">
-->
<xsd:simpleType name="EOLType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="LF"/>
    <xsd:enumeration value="CRLF"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the encoding type as a string
-->
<xsd:simpleType name="encodingType">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

```



```

<!--
  <schedule>
    <submit timebase="source|admin">2008-12-07T16:07</submit>
    <repeat>
      <frequency interval="hours">2</frequency>
      <expireTime>2008-12-0816:07</exipreTime>
    </repeat>
  </schedule>
-->
<xsd:complexType name="scheduleType">
  <xsd:sequence>
    <xsd:element name="submit" type="submitType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="repeat" type="repeatType" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  <submit timebase="source|admin|UTC">2008-12-07T16:07</submit>
-->
<xsd:complexType name="submitType">
  <xsd:simpleContent>
    <xsd:extension base="noZoneTimeType">
      <xsd:attribute name="timebase" type="timebaseType" use="required"/>
      <xsd:attribute name="timezone" type="xsd:string" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  <repeat>
    <frequency interval="hours">2</frequency>
    ..optionally..
    <expireTime>2008-12-0816:07</expireTime>
    ..or..
    <expireCount>2</expireCount>
  </repeat>
-->
<xsd:complexType name="repeatType">
  <xsd:sequence>
    <xsd:element name="frequency" type="freqType" maxOccurs="1" minOccurs="1"/>
    <xsd:choice minOccurs="0">
      <xsd:element name="expireTime" type="noZoneTimeType"/>
      <xsd:element name="expireCount" type="positiveIntegerType"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
<!--
  <frequency interval="hours">2</frequency>
-->
<xsd:complexType name="freqType">
  <xsd:simpleContent>
    <xsd:extension base="positiveIntegerType">
      <xsd:attribute name="interval" type="intervalType" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  Defines positive integer type
  i.e., 1+
-->
<xsd:simpleType name="positiveIntegerType">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the interval enumeration values of
  "minutes", "hours", "days", "weeks", "months" or "years"
-->
<xsd:simpleType name="intervalType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="minutes"/>
    <xsd:enumeration value="hours"/>
    <xsd:enumeration value="days"/>
    <xsd:enumeration value="weeks"/>
    <xsd:enumeration value="months"/>
    <xsd:enumeration value="years"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the interval of either "source", "admin" or "UTC"
  source = use timezone of the source Agent.
  admin = use timezone of the administrator executing the command script.
  UTC = Timezone is UTC.
-->

```

```

-->
<xsd:simpleType name="timebaseType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="source"/>
    <xsd:enumeration value="admin"/>
    <xsd:enumeration value="UTC"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines a date and time without a time zone (2008-12-08T16:07)
-->
<xsd:simpleType name="noZoneTimeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[\n\r\t ]*\d{4}\-(0[1-9]|1[0-2])\-(0[1-9]|[1-2][0-9]|
3[0-1])T([0-1][0-9]|2[0-3]):[0-5][0-9]([\+\-]\d{4}Z)?[\n\r\t ]*" />
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the ID element, e.g. 56
-->
<xsd:simpleType name="idType">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<!--
  Defines the resultCode type -2 - 9999
<status resultCode="8011">
-->
<xsd:simpleType name="resultCodeType">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="-2"/>
    <xsd:maxInclusive value="9999"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Define the metaDataSet type comprising one or more key value pairs
<metaDataSet>
  <metaData key="name">value</metaData>
  <metaData key="name">value</metaData>
</metaDataSet>
-->
<xsd:complexType name="metaDataSetType">
  <xsd:sequence>
    <xsd:element name="metaData" type="metaDataType" maxOccurs="unbounded"
minOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
<!--
  Define the metaData type which is made up of a key and a value
<metaData key="name">value</metaData>
-->
<xsd:complexType name="metaDataType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="key" type="xsd:string" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<!--
  Defines containing element for triggers
<trigger log="yes">
  <fileExist comparison="=" value="Exist">file1</fileExist>
  <fileSize comparison=">=" value="1GB">file1</fileSize>
</trigger>
-->
<xsd:complexType name="triggerType">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="fileExist" type="fileExistTriggerType" maxOccurs="unbounded"
minOccurs="1" />
    <xsd:element name="fileSize" type="fileSizeTriggerType" maxOccurs="unbounded"
minOccurs="1" />
  </xsd:choice>
  <xsd:attribute name="log" type="logEnabledType" use="required"/>
</xsd:complexType>
<!--
  Defines the file exists trigger type
<fileExist comparison="=" value="Exist">file1</trigger>
-->
<xsd:complexType name="fileExistTriggerType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="comparison" type="comparisonFileExistTriggerType"
use="required"/>
    </xsd:extension>
  </xsd:simpleContent>

```

```

        <xsd:attribute name="value" type="valueFileExistTriggerType" use="required"/>
    </xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
<!--
    Defines file size trigger type
    <fileSize comparison="=" value="1GB">file1,file2,file3</trigger>
-->
<xsd:complexType name="fileSizeTriggerType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="comparison" type="comparisonFileSizeTriggerType"
use="required"/>
            <xsd:attribute name="value" type="valueFileSizeTriggerType" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<!--
    Defines the enumeration values for file exists trigger conditions
    valueFileExistTriggerType="exist|noexist"
-->
<xsd:simpleType name="valueFileExistTriggerType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="exist"/>
        <xsd:enumeration value="noexist"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumeration values for file exists trigger comparison operator
    comparisonFileExistTriggerType="="|"!="
-->
<xsd:simpleType name="comparisonFileExistTriggerType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="="/>
        <xsd:enumeration value="!=">
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumeration values for file size trigger comparison operator
    comparisonFileSizeTriggerType=">="
-->
<xsd:simpleType name="comparisonFileSizeTriggerType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="&gt;="/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the file size value pattern
    <fileSize comparison=">=" value="10|10B|10KB|10MB|10GB">file1</fileSize>
-->
<xsd:simpleType name="valueFileSizeTriggerType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0123456789]+([bB]|[kK][bB]|[mM][bB]|[gG][bB]|)"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the enumeration values for trigger logging enabled flag
    <trigger log="yes|no">
-->
<xsd:simpleType name="logEnabledType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="yes"/>
        <xsd:enumeration value="no"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
    Defines the reply type
    <reply QMGR="QMGR name" persistent="true">Queue Name</reply>
-->
<xsd:complexType name="replyType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="QMGR" type="xsd:string" use="required"/>
            <xsd:attribute name="persistent" type="persistenceType" use="optional"/>
            <xsd:attribute name="detailed" type="detailedType"
use="optional" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<!--
    Defines the accepted choices for the detailed attribute.

```

```

-->
<xsd:simpleType name="detailedType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="true"/>
    <xsd:enumeration value="false"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the priority type
  <transferset priority="1">
-->
<xsd:simpleType name="priorityType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0123456789]"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Define the job information element
  <job>
    <name>JOBNAME</name>
  </job>
-->
<xsd:complexType name="jobType">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines an action
  <action>
    <runCommand name="myCommand.sh" />
  </action>
-->
<xsd:complexType name="commandActionType">
  <xsd:choice>
    <xsd:element name="command" type="commandType" maxOccurs="1" minOccurs="0"/>
  </xsd:choice>
</xsd:complexType>
<!--
  Defines a command
  <command name="runme" successRC="0" maxReplyLength="1024">
    <argument>firstArg</argument>
    <argument>secondArg</argument>
  </command>
-->
<xsd:complexType name="commandType">
  <xsd:sequence>
    <xsd:element name="argument" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element name="target" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element name="property" type="propertyType" maxOccurs="unbounded"
minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="successRC" type="xsd:string" use="optional"/>
  <xsd:attribute name="retryCount" type="nonNegativeIntegerType" use="optional"/>
  <xsd:attribute name="retryWait" type="nonNegativeIntegerType" use="optional"/>
  <xsd:attribute name="type" type="callTypeType" use="optional"/>
  <xsd:attribute name="priority" type="commandPriorityType" use="optional"/>
  <xsd:attribute name="message" type="xsd:string" use="optional"/>
</xsd:complexType>
<!--
  Defines the enumeration values for the type of a command
  type="executable|antscript|jcl"
-->
<xsd:simpleType name="callTypeType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="executable"/>
    <xsd:enumeration value="antscript"/>
    <xsd:enumeration value="jcl"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the priority type for a command
  priority="5"
-->
<xsd:simpleType name="commandPriorityType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[123456789]"/>
  </xsd:restriction>
</xsd:simpleType>

```

```

<!--
  Defines the property type that is used as a child of commandType
  <property name="xxx" value="yyy"/>
-->
<xsd:complexType name="propertyType">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>
<!-- Defines a non-negative integer type -->
<xsd:simpleType name="nonNegativeIntegerType">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the transfer command specific reply information, to be included as part the
  general reply
  <transferReply>
    <preSourceData>
      <runCommandReply resultCode="0">
        <stdout>
          <line>the quick brown fox jumped over the lazy dog</line>
        </stdout>
        <stderr></stderr>
      </runCommandReply>
    </preSourceData>
  </transferReply>
-->
<xsd:complexType name="transferReplyType">
  <xsd:sequence>
    <xsd:element name="preSourceData" type="actionReplyType" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="postSourceData" type="actionReplyType" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="preDestinationData" type="actionReplyType" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="postDestinationData" type="actionReplyType" minOccurs="0"
maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Define the action reply type information
  <actionReply>
    <runCommandReply resultCode="1">
      <stdout></stdout>
      <stderr>
        <line>permission denied</line>
      </stderr>
    </runCommandReply>
  </actionReply>
-->
<xsd:complexType name="actionReplyType">
  <xsd:choice>
    <xsd:element name="runCommandReply" type="commandReplyType" maxOccurs="1"
minOccurs="0"/>
  </xsd:choice>
</xsd:complexType>
<!--
  Defines command specific reply information, to be included as part the general reply
  <commandReply resultCode="0">
    <stdout>
      <line>first line of output text</line>
      <line>second line of output text</line>
    </stdout>
    <stderr>
      <line>line of error text</line>
    </stderr>
  </commandReply>
-->
<xsd:complexType name="commandReplyType">
  <xsd:sequence>
    <xsd:element name="stdout" type="textLinesType" maxOccurs="1" minOccurs="1"/>
    <xsd:element name="stderr" type="textLinesType" maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="resultCode" type="xsd:int" use="required"/>
</xsd:complexType>
<!-- Defines type for lines of text -->
<xsd:complexType name="textLinesType">
  <xsd:sequence>
    <xsd:element name="line" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

```

```

<!--
general Defines the ping agent command specific reply information, to be included as part the
reply
<pingAgentReply resultCode="0">
  <agentVersion>Build level: f000-20090408-1200</agentVersion>
</pingAgentReply>
-->
<xsd:complexType name="pingAgentReplyType">
  <xsd:sequence>
    <xsd:element name="agentVersion" type="xsd:string" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines sequence of exit elements
  <exit ...
  <exit ...
-->
<xsd:complexType name="exitGroupType">
  <xsd:sequence>
    <xsd:element name="exit" type="exitType" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines the outcome of calling a command
  <command ...
  <callResult ...
-->
<xsd:complexType name="callGroupType">
  <xsd:sequence>
    <xsd:element name="command" type="commandType" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="callResult" type="callResultType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
<!--
  Defines either the successful call of a command, or a failed attempt to call a command
  <callResultType outcome="success|failure|error" retries="X">
    <result ... />
  </callResultType>
-->
<xsd:complexType name="callResultType">
  <xsd:sequence>
    <xsd:element name="result" type="resultType" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="returnCode" type="xsd:integer" use="optional"/>
  <xsd:attribute name="retries" type="xsd:integer" use="optional"/>
  <xsd:attribute name="outcome" type="outcomeType" use="required"/>
</xsd:complexType>
<!--
  Defines the information recorded for the successful call of a command
  <result...>
    <stdout...
    <stderr...
    <error...
  </result...>
-->
<xsd:complexType name="resultType">
  <xsd:sequence>
    <xsd:element name="stdout" type="outputType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="stderr" type="outputType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="error" type="xsd:string" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="returnCode" type="xsd:integer" use="optional"/>
  <xsd:attribute name="outcome" type="outcomeType" use="required"/>
  <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
</xsd:complexType>
<!-- Enumeration of call outcomes - success, failure or error -->
<xsd:simpleType name="outcomeType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="success"/>
    <xsd:enumeration value="failure"/>
    <xsd:enumeration value="error"/>
  </xsd:restriction>
</xsd:simpleType>
<!--
  Defines the information recorded for each line of standard output / standard error
  generated by calling a program
  <line>line 1</line>
  <line>line 2</line>
  etc.
-->
<xsd:complexType name="outputType">
  <xsd:sequence>

```

```

        <xsd:element name="line" type="xsd:string" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<!--
    Defines the information recorded for an unsuccessful program call.
-->
<xsd:complexType name="callFailedType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string"/>
    </xsd:simpleContent>
</xsd:complexType>
<!--
    Defines the exit type; records the transfer exit class name and a status message
    <exit name="class com.example.exit.StartExit">
        <status ...
    </exit>
-->
<xsd:complexType name="exitType">
    <xsd:sequence>
        <xsd:element name="status" type="exitStatusType" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
<!--
    Defines exit status to record whether exit voted to proceed or cancel transfer.
    <status resultCode="proceed">
        <supplement>go ahead</supplement>
    </status>
-->
<xsd:complexType name="exitStatusType">
    <xsd:sequence>
        <xsd:element name="supplement" type="xsd:string" maxOccurs="unbounded"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="resultCode" type="exitResultEnumType" use="optional"/>
</xsd:complexType>
<!--
    Defines the enumeration for transfer exit result values.
    <status resultCode="proceed">
-->
<xsd:simpleType name="exitResultEnumType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="proceed"/>
        <xsd:enumeration value="cancelTransfer"/>
        <xsd:enumeration value="cancelTask"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

**Note:** From IBM MQ 9.0, Managed File Transfer does not support the Web Gateway or web agents.

### Related concepts

[“XML message formats used by MFT” on page 2613](#)

Managed File Transfer uses messages in XML format for a number of purposes: to command an agent; to log information about the monitors, schedules, and transfers; and to define information used for configuration. The logical structure of the XML formats used for these purposes described by XML schema.

## IBM MQ utilities on z/OS reference

Reference information about the syntax, and usage of the various IBM MQ utility programs.

## IBM MQ utilities on z/OS by category

Use this topic as a reference to the different categories of utilities.

This topic introduces the IBM MQ utility programs that are provided to help you perform various administrative tasks. The utility programs are described in the subsequent sections:

[The IBM MQ CSQUTIL utility program: Managing page sets](#)

[The IBM MQ CSQUTIL utility program: Issuing commands](#)

[The IBM MQ CSQUTIL utility program: Managing queues](#)

The IBM MQ CSQJU003 Change log inventory utility

The remaining IBM MQ utilities summarizes what you can do with these utilities.

*Table 378. The IBM MQ CSQUTIL utility program: Managing page sets*

<b>Purpose</b>	<b>Function</b>	<b>See topic</b>
Format VSAM data sets as IBM MQ page sets.	FORMAT	<a href="#">“Formatting page sets (FORMAT) on z/OS” on page 2742</a>
Control recovery processing used for IBM MQ page sets.	FORMAT	<a href="#">“Formatting page sets (FORMAT) on z/OS” on page 2742</a>
Extract page set information.	PAGEINFO	<a href="#">“Page set information (PAGEINFO) on z/OS” on page 2746</a>
Copy IBM MQ page sets.	COPYPAGE	<a href="#">“Expanding a page set (COPYPAGE) on z/OS” on page 2747</a>
Copy IBM MQ page sets and reset the log information.	RESETPAGE	<a href="#">“Copying a page set and resetting the log (RESETPAGE) on z/OS” on page 2749</a>

*Table 379. The IBM MQ CSQUTIL utility program: Issuing commands*

<b>Purpose</b>	<b>Function</b>	<b>See topic</b>
Issue IBM MQ commands.	COMMAND	<a href="#">“Using the COMMAND function of CSQUTIL on z/OS” on page 2750</a>
Produce a set of DEFINE, ALTER or DELETE commands for objects.	COMMAND	<a href="#">Making a list of DEFINE commands</a>
Produce a client channel definition file.	COMMAND	<a href="#">Making a client channel definition file</a>
Produce a set of DEFINE commands for objects (offline).	SDEFS	<a href="#">“Producing a list of IBM MQ define commands (SDEFS) on z/OS” on page 2757</a>



Table 380. The IBM MQ CSQUTIL utility program: Managing queues

Purpose	Function	See topic
Copy contents of a queue to a data set.	COPY	<a href="#">“Copying queues into a data set while the queue manager is running (COPY) on z/OS” on page 2761</a>
Copy contents of a queue to a data set (offline).	SCOPY	<a href="#">“Copying queues into a data set while the queue manager is not running (SCOPY) on z/OS” on page 2763</a>
Delete contents of a queue.	EMPTY	<a href="#">“Emptying a queue of all messages (EMPTY) on z/OS” on page 2766</a>
Restore contents of a queue.	LOAD	<a href="#">“Restoring messages from a data set to a queue (LOAD) on z/OS” on page 2767</a>

Table 381. The IBM MQ CSQJU003 Change log inventory utility

Purpose	Function	See topic
Add active or archive log data sets.	NEWLOG	<a href="#">“Adding information about a data set to the BSDS (NEWLOG) on z/OS” on page 2775</a>
Delete active or archive log data sets.	DELETE	<a href="#">“Deleting information about a data set from the BSDS (DELETE) on z/OS” on page 2777</a>
Supply passwords for archive logs.	ARCHIVE	<a href="#">“Supplying a password for archive log data sets (ARCHIVE) on z/OS” on page 2778</a>
Control the next restart of the queue manager.	CRESTART	<a href="#">“Controlling the next restart (CRESTART) on z/OS” on page 2778</a>

Table 381. The IBM MQ CSQJU003 Change log inventory utility (continued)

Purpose	Function	See topic
Set checkpoint records.	CHECKPT	<a href="#">“Setting checkpoint records (CHECKPT) on z/OS” on page 2779</a>
Update the highest written log RBA.	HIGHRBA	<a href="#">“Updating the highest written log RBA (HIGHRBA) on z/OS” on page 2780</a>

Table 382. The remaining IBM MQ utilities

Name	Purpose	See topic
<b>CSQJU004</b> (Print log map utility)	List information about the log.	<a href="#">“The print log map utility (CSQJU004) on z/OS” on page 2781</a>
<b>CSQ1LOGP</b> (Log print utility)	Print the log. Extract log records into sequential files.	<a href="#">“The log print utility (CSQ1LOGP) on z/OS” on page 2782</a>
<b>CSQ5PQSG</b> ( IBM MQ table update utility)	Add and remove queue sharing group and queue manager entries in the IBM MQ tables held in the shared Db2 data-sharing group.	<a href="#">“The queue sharing group utility (CSQ5PQSG) on z/OS” on page 2793</a>
<b>CSQJUFMT</b> (Active log preformat utility)	Preformat log data sets Preformat Shared Message Data Sets (SMDS)	<a href="#">“The active log preformat utility (CSQJUFMT) on z/OS” on page 2797</a>
<b>CSQUDLQH</b> (Dead-letter queue handler utility)	Process messages on the dead-letter queue.	<a href="#">“The dead-letter queue handler utility (CSQUDLQH) on z/OS” on page 2798</a>
<b>CSQUCVX</b> (Data conversion exit utility)	Generate data conversion exit routines.	Writing a data-conversion exit program for IBM MQ for z/OS
<b>CSQUDSPM</b> (Display queue manager utility)	Display information about queue managers. The equivalent function on <a href="#">Multiplatforms</a> is <b>dspmq</b> .	<a href="#">“Display queue manager information utility (CSQUDSPM)” on page 2810</a>

These utilities are located in the `th1qua1.SCSQAUTH` or `th1qua1.SCSQLOAD` IBM MQ load libraries. Concatenate the appropriate IBM MQ language load library `th1qua1.SCSQANLx` (where `x` is the language letter) in the STEPLIB with the `th1qua1.SCSQAUTH` and `th1qua1.SCSQLOAD`.

**Note:** Failing to do so leads to unpredictable results (including issues with queue manager initialization).

Depending upon the utility work that your enterprise undertakes, you should use versions of the libraries at the same level as your queue managers where possible.

The utility control statements are available only in U.S. English. In some cases, the Db2 library `db2qua1.SDSNLOAD` is also needed.

## IBM MQ utility program (CSQUTIL) on z/OS

The CSQUTIL utility program is provided with IBM MQ to help you to perform backup, restoration, and reorganization tasks, and to issue IBM MQ commands.

Through this utility program, you can invoke functions in these groups:

### Page set management

These functions enable you to manage IBM MQ page sets. You can format data sets as page sets, change the recovery processing performed against page sets, extract page set information, increase the size of page sets and reset the log information contained in a page set. The page set must not belong to a queue manager that is currently running.

### Command management

These functions enable you to:

- Issue commands to IBM MQ
- Produce a list of DEFINE, ALTER, or DELETE commands for your IBM MQ objects

### Queue management

These functions enable you to back up and restore queues and page sets, copy queues and page sets to another queue manager, reset your queue manager, or to migrate from one queue manager to another.

Specifically, you can:

- Copy messages from a queue to a data set
- Delete messages from a queue
- Restore previously copied messages to their appropriate queues

The scope of these functions can be either:

- A *queue*, in which case the function operates on all messages in the specified queue.
- A *page set*, in which case the function operates on all the messages, in all the queues, on the specified page set.

Use these functions only for your own queues; do not use them for system queues (those with names beginning SYSTEM).

All the page set management functions, and some of the other functions, operate while the queue manager is not running, so you do not need any special authorization other than the appropriate access to the page set data sets. For the functions that operate while the queue manager is running, CSQUTIL runs as an ordinary z/OS batch IBM MQ program, issuing commands through the command server, and using the IBM MQ API to access queues.

You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*), to use the IBM MQ DISPLAY commands, and to use the IBM MQ API to access any queues that you want to manage. See the usage notes for each function for more information.



**Attention:** If you use CSQUTIL to define a channel, and the connection name contains two parts (the host name and port number) you must enclose the host name and port number within single

quotation marks to maintain the limit on the number of permissible parameters. Similarly, if your connection name consists of an IP address and port number, you must enclose these parameters within single quotation marks.

## Invoking the IBM MQ utility program on z/OS

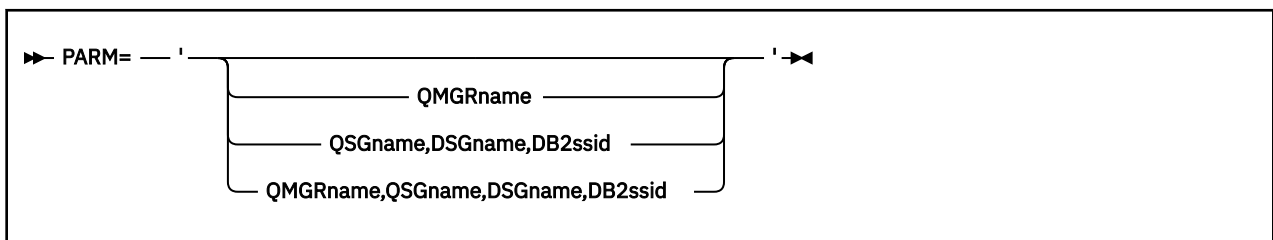
Use this topic to understand how to invoke CSQUTIL, the format of its parameters, and its return codes.

The CSQUTIL utility program runs as a z/OS batch program, below the 16 MB storage line. Specify the resources that the utility is to work with in the PARM parameter of the EXEC statement of the JCL.

```
// EXEC PGM=CSQUTIL, PARM=
```

Figure 11. How to invoke the CSQUTIL utility program

where PARM= expands to:



- [PARM parameters](#)
- [Return codes](#)

## PARM parameters

### QMGRname

Specifies the 1- to 4-character name of the queue manager or queue sharing group to which CSQUTIL is to connect.

If you specify the name of a queue sharing group, CSQUTIL connects to any queue manager in that group

### QSGname

Specifies the 1- to 4-character name of the queue sharing group from which CSQUTIL is to extract definitions.

### DSGname

Specifies the 8-character name of the Db2 data-sharing group from which CSQUTIL is to extract definitions.

### db2ssid

Specifies the 4-character name, or group attach name, of the Db2 database subsystem to which CSQUTIL is to attach for stand-alone functions.

### Which PARM parameters do you need?

Figure 11 on page 2740 shows that you can specify one of four options on the PARM statement. The option you specify depends on the function you need to implement, as follows:

- Use PARM= (or omit it all together) if you are using only offline functions, and not QSGDISP(GROUP) or QSGDISP(SHARED).
- Use PARM= ' QMGRname ' only if you intend to use functions that require the queue manager to be running, such as COPY and COMMAND.

- Use `PARM= ' QSGname, DSGname, db2ssid '` if you intend to use the SDEFS function with either `QSGDISP(GROUP)` or `QSGDISP(SHARED)` specified. This is because CSQUTIL requires access to Db2 to perform the SDEFS function in this situation.
- Use `PARM= ' QMGRname, QSGname, DSGname, db2ssid '` if you intend to combine the previous two functions in one CSQUTIL job.

If you specify a queue manager name as blanks, CSQUTIL uses the name of the default queue manager specified for z/OS batch programs in CSQBDEFV. The utility then uses this queue manager for the whole job step. When the utility connects to the queue manager, the authorization of the "signed-on user name" is checked to see which functions the invocation is allowed to use.

You specify the functions required by statements in the SYSIN data set according to these rules:

- The data set must have a record length of 80.
- Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
- Records with an asterisk ( `*` ) in column 1 are interpreted as comments and are ignored.
- Blank records are ignored.
- Each statement must start on a new line.
- A trailing - means continue from column 1 of the next record.
- A trailing + means continue from the first non-blank column of the next record.
- The keywords of statements are not case-sensitive. However, some arguments, such as queue name, are case sensitive.

The utility statements refer to the default or explicitly named DDnames for input and output. Your job can use the COPY and LOAD functions repeatedly and process different page sets or queues during a single run of the utility.

All output messages are sent to the SYSPRINT data set, which must have a record format of VBA and a record length of 125.

While running, CSQUTIL uses temporary dynamic queues with names of the form `SYSTEM.CSQUTIL.*`

## Return codes

When you are using the COMMAND verb to issue MQSC commands, you must use FAILURE(CONTINUE) so any failure in the commands that are issued give a non-zero return code. The default is FAILURE(IGNORE) and the return code from the command is always zero.

When CSQUTIL returns to the operating system, the return code can be:

- 0** All functions completed successfully.
- 4** Some functions completed successfully, some did not, or forced a sync point.
- 8** All the attempted functions failed.
- 12** No functions attempted; there was a syntax error in the statements or the expected data sets were missing.

In most cases, if a function fails or is forced to take a sync point, no further functions are attempted. In this case, the message CSQU147I replaces the normal completion message CSQU148I.

See the usage notes for each function for more information about success or failure.

## Syncpoints

The queue management functions used when the queue manager is running operate within a syncpoint so that, if a function fails, its effects can be backed out. The queue manager attribute, MAXUMSGS, specifies the maximum number of messages that a task can get or put within a single unit of recovery.

The utility issues an MQCMIT call when the MAXUMSGS limit is reached and issues the warning message CSQU087I. If the utility later fails, the changes already committed are not backed out.

Do not just rerun the utility to correct the problem or you might get duplicate messages on your queues.

Instead, use the current depth of the queue to work out, from the utility output, which messages have not been backed out. Then determine the most appropriate course of action. For example, if the function is LOAD, you can empty the queue and start again, or you can choose to accept duplicate messages on the queues.

To avoid such difficulties if the function fails, there are two options:

1. 1. Temporarily increase the value of MAXUMSGS to be greater than the number of messages in the:

- Queue, if you are working with a single queue.
- Longest queue in the page set, if you are working with an entire page set.

Use the `DISPLAY QSTATUS` command to find out the value of the CURDEPTH attribute, which is the current depth of the queue.

To find out the value of MAXUMSGS, use the `DISPLAY QMGR MAXUMSGS` command.

Then rerun the command, and after the utility has successfully run change MAXUMSGS back to what it was before.

**Note:** This approach is simpler but having a large number of messages in a single unit of work can incur a high CPU cost.

2. Use the utility to LOAD the messages to a temporary queue.

Note that you can delete the temporary queue in the event of failure and the job rerun.

Then use the MQSC MOVE command to move the messages from the temporary queue to the target queue. For example:

```
MOVE QL(tempq) TOQLLOCAL(targetq) TYPE(ADD)
```

Once the command has completed successfully, you can delete the temporary queue.

This approach takes longer, but moves the messages in a number of small units of work so is more efficient in terms of CPU cost.

## **Monitoring the progress of the IBM MQ utility program on z/OS**

You can monitor the progress of the CSQUTIL program by monitoring statements output to SYSPRINT.

To record the progress of CSQUTIL, every SYSIN statement is echoed to SYSPRINT.

The utility first checks the syntax of the statements in the SYSIN. The requested functions are started only if all the statements are syntactically correct.

Messages giving a commentary on the progress of each function are sent to SYSPRINT. When the processing of the utility is complete, statistics are printed with an indication of how the functions completed.

## **Formatting page sets (FORMAT) on z/OS**

You can use the CSQUTIL program to format page sets.

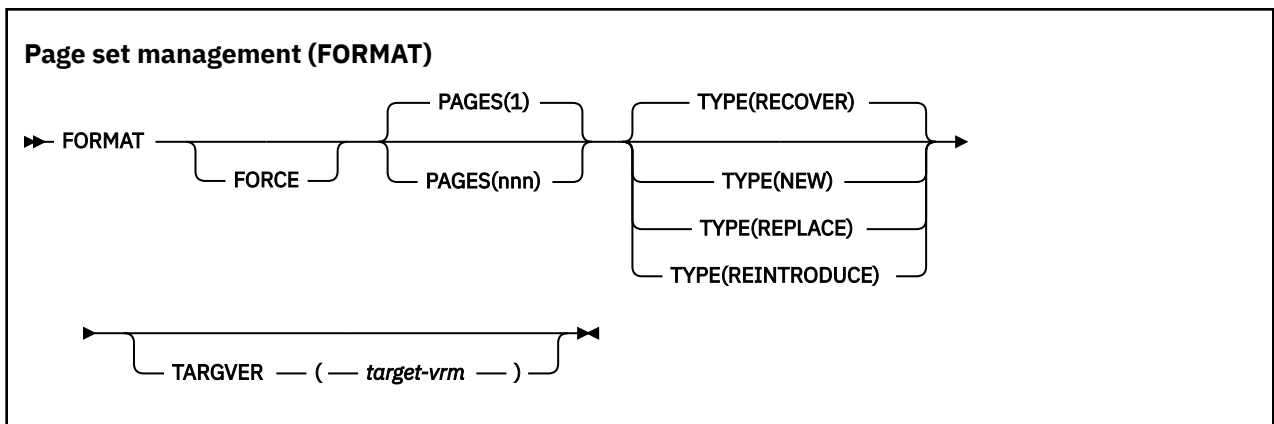
Use the FORMAT function to format page sets on all data sets specified by DDnames CSQP0000 through CSQP0099. In this way, you can format up to 100 page sets in a single invocation of the utility program. Use the FORCE keyword to reuse existing data sets.

You can also use the `FORMAT` function to change the recovery processing that is performed against page sets when the queue manager starts, using the `TYPE` keyword. This can assist in changing or recovering page sets, or reintroducing page sets that have been offline or suspended.

In summary, to:

- Reinststate a page set with no data, use `FORMAT` with the `TYPE(NEW)` option
- Reinststate a page set with old data, use `FORMAT` with the `TYPE(REPLACE)` option
- Reinststate a page set with old data made up-to-date, do not use `FORMAT` but start the queue manager with a backed-up copy of the page set
- Reintroduce a page set that was offline when the queue manager was backwards migrated using the `START QMGR BACKMIG` command, use `FORMAT` with the `TYPE(REINTRODUCE)` option.

Page sets have identifiers (PSIDs, in the range 00 through 99) which are established by the DDnames used for the data sets in the queue manager started task procedure; DDname `CSQP00nn` specifies the page set with identifier `nn`. The DDnames you use for the `FORMAT` function do not have to correspond to those used in the queue manager started task procedure, and do not therefore have any significance regarding page set identifiers.



- [Keywords and parameters](#)
- [Example](#)
- [Usage notes](#)

## Keywords and parameters

### FORCE

Specifies that existing data sets are to be reused without having to delete and redefine them first. You must define any page sets you want to reuse with the `REUSE` attribute in the `AMS DEFINE CLUSTER` statement.

See the [Optional Parameters](#) section of the `z/OS DEFINE CLUSTER` command for more information on `REUSE`.

The following code is an example on how you set `REUSE`:

```

//IDCAMS EXEC PGM=IDCAMS,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ALTER VICY.MQOM.PSID04 REUSE
/*

```

To undo the `REUSE` option, use the `ALTER` attribute to change the `REUSE` parameter to `NOREUSE`.

The `FORCE` keyword is not valid if `TYPE(REPLACE)` is specified.

**PAGES (nnn)**

Specifies the minimum number of pages to format in each page set. This enables a data set that spans more than one volume to be formatted.

Formatting of the data set is always done in whole space allocations, as specified as primary or secondary quantities when the data set is defined. The number of space allocations formatted is the minimum necessary to provide the requested number of pages; if there is insufficient data set space available, as many extents as can be obtained are formatted. If an existing page set is being reused (with the FORCE keyword), the whole page set is formatted, if that is larger.

The number of pages must be in the range 1 through 16 777 213 (because the maximum page set size is 64 GB (gigabytes)). The default is 1.

The PAGES keyword is not valid if TYPE(REPLACE) is specified.

**TYPE**

Specifies the type of recovery processing that is performed against queue manager page sets. Values are:

**RECOVER**

Use RECOVER for a data set that is to be a new page set for a queue manager (that is, to have a PSID which was never been used before).

This is the default.

The data set is formatted, and any messages or other data are erased. If a DDname is added to the queue manager's started task procedure for the new PSID that specifies this data set, it will be recognized as a new page set when the queue manager is restarted.

If such a data set was used as a page set with a PSID that has been used before, on restart the queue manager attempts to recover all queues and their messages that use storage classes that reference the page set from the time the page set was first used. This may make restart a lengthy process, and is unlikely to be what is wanted.

**NEW**

Use NEW for a data set that is to be a page set with a PSID that has been used before for a queue manager and with data that can be discarded, to restart a failed queue manager quickly or to reintroduce the page set after it has been offline or suspended.

The data set is formatted, and any messages or other data are erased. When the queue manager is restarted, with a DDname for the old PSID that specifies this data set, it does not recover the page set but treats it as if it has been newly added to the queue manager, and any historical information about it is discarded. All queues that use storage classes referencing this page set are cleared of all messages, in a similar fashion to the way that nonpersistent messages are cleared during restart processing. This means that there will be no effect on restart time.

**REPLACE**

Use REPLACE for a data set with a PSID that has been used before for a queue manager and with data that is known to be consistent and up to date, to reintroduce the page set after being offline or suspended.

The data set is not formatted, and any messages or other data are preserved. When the queue manager is restarted with a DDname for the PSID that specifies this data set, it does not recover the page set but treats it as if it has never been offline, or suspended, and any historical information about it is retained. All queues that use storage classes that reference the page set keep their messages. This means that there will be no effect on restart time.

This option will only be successful if the page set is in a consistent state; that is, on its last use the queue manager was terminated normally by a STOP QMGR MODE(FORCE) or MODE(QUIESCE) command.



## REINTRODUCE

Use REINTRODUCE for a data set that was offline when the queue manager was backward migrated using the START QMGR BACKMIG command, and you want to reintroduce the page set with old data made up-to-date, or if you want to reintroduce the data set with old data.

If you want to reintroduce the data set with old data, you also need to subsequently run FORMAT TYPE(REPLACE) CSQUTIL against the data set.

The data set is migrated to the version specified by the TARGVER keyword, and the data is not changed.

TYPE(REINTRODUCE) is not valid for page set zero.

The version of CSQUTIL must match the version of queue manager that last used the IBM MQ page set when TYPE(REINTRODUCE) is used. The command fails if migration is not allowed to the version specified by the TARGVER keyword.

You can also use TYPE(NEW) or TYPE(RECOVER) with CSQUTIL, at the migration target version to reintroduce the page set at that version.

## TARGVER

Specifies the target version for the formatted page set

### *target-**vr**m*

The version, release and modification number for the target version for TYPE(REINTRODUCE), for example 910.

This must be an LTS release, otherwise a [CSQU104E](#) message is output, and the utility exits with a return code of 12.



**Attention:** TARGVER is not valid if you do not also specify TYPE(REINTRODUCE).

## Example

Figure 12 on page 2745 illustrates how the FORMAT command is invoked from CSQUTIL. In this example, two page sets, referenced by CSQP0000 and CSQP0003, are formatted by CSQUTIL.

```
//FORMAT EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0000 DD DISP=OLD,DSN=pageset.dsname0
//CSQP0003 DD DISP=OLD,DSN=pageset.dsname3
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
FORMAT
/*
```

Figure 12. Sample JCL for the FORMAT function of CSQUTIL

Figure 13 on page 2745 illustrates how the FORMAT command with the TYPE option is invoked from CSQUTIL. In this example, the page set referenced by CSQP0003 is formatted by CSQUTIL.

```
//FORMAT EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0003 DD DISP=OLD,DSN=page set.dsname3
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
FORMAT TYPE(RECOVER)
/*
```

Figure 13. Sample JCL for the FORMAT function of CSQUTIL with the TYPE option

## Usage notes

1. You cannot format page sets that belong to a queue manager that is still running.
2. When you use FORMAT, it is not necessary to specify a queue manager name.
3. If you use TYPE(REPLACE), recovery logs starting from when the page set was first used with the queue manager, or from when the page set was last formatted, must be available.
4. If you use data set names in which the queue manager name is a high-level qualifier, you can more easily identify which page sets are used by which queue manager, if more than one queue manager is defined.
5. Any update to a resource due to the resolution of an incomplete unit of work, where the update relates to a page on a page set that has been formatted with TYPE(REPLACE) or TYPE(NEW), is not honored. The update to the resource is lost.
6. If there is an error when formatting a page set, it does not prevent other page sets from being formatted, although the FORMAT function is considered to have failed.
7. Failure of this function does not prevent other CSQUTIL functions being attempted.

## Page set information (PAGEINFO) on z/OS

Use the PAGEINFO function to extract page set information from one or more page sets, specified by DDnames in the range CSQP0000 through CSQP0099, for the source data sets from which page set information is required.

### Page set management (PAGEINFO)

►► PAGEINFO ◄◄

## Keywords and parameters

There are no keywords or parameters.

## Example

In Figure 14 on page 2746, page set information is required from two existing page sets.

```
//PAGEINFO EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0001 DD DISP=OLD,DSN=page set.existing.name1
//CSQP0006 DD DISP=OLD,DSN=page set.existing.name6
//SYSPRINT DD SYSOUT=*
//SYSIN DD
* Extract page set information for 2 existing page sets (CSQS0001 and CSQS0006)
PAGEINFO
/*
```

Figure 14. Sample JCL showing the use of the PAGEINFO function

where:

### CSQP0001, CSQP0006

Are the DDnames of the source data sets from which you want to extract page set information.

Information returned from PAGEINFO might include:

- Page set number
- Number of pages in a page set
- Queue manager associated with a page set

- Utility status information
- Page set recovery RBA for each page set
- System recovery RBA for all the page sets reported on by the PAGEINFO function

## Usage notes

1. You cannot use PAGEINFO on the page sets of a queue manager that is running.
2. Failure of this function does not prevent other CSQUTIL functions from being attempted.
3. If you attempt to use the PAGEINFO function after the queue manager has terminated abnormally, the page sets might not have been closed properly. If a page set has not been closed properly, you cannot successfully run the PAGEINFO function against it. To avoid this problem, run the AMS VERIFY command before using the PAGEINFO function. The AMS VERIFY command might produce error messages. However, it does close the page sets properly so that the PAGEINFO function can complete successfully.

For more information about the AMS [VERIFY](#) command, see the *z/OS DFSMS Access Method Services for VSAM* manual.

4. The system recovery RBA relates only to those page sets processed; it does not relate to the whole queue manager unless all the page sets for the queue manager are included. If the page sets are from more than one queue manager, no system recovery RBA can be determined.

## Expanding a page set (COPYPAGE) on z/OS

Use the COPYPAGE function to copy one or more page sets to a larger page set.

**Note:** The COPYPAGE function is only used for *expanding* page sets. It is not used for making backup copies of page sets. If you want to do this, use AMS REPRO as described in [How to back up and recover page sets](#). When you have used the COPYPAGE function, the page sets cannot be used by a queue manager with a different name, so do not rename your queue manager.

Use the COPYPAGE function to copy one or more page sets to a larger page set. All queues and messages on the page set are copied. If you copy page set zero, all the IBM MQ object definitions are also copied. Each page set is copied to a destination data set that must be formatted as a page set. Copying to a smaller page set is not supported.

If you use this function, you must modify the page set definition in the started task procedure to reflect the change of the name of the data set on which the new page set resides.

To use the COPYPAGE function, define DDnames in the range CSQS0000 through CSQS0099 for the source data sets, and define DDnames for the target data sets from CSQT0000 through CSQT0099.

For more information, see [Managing page sets](#).

### Page set management (COPYPAGE)

➤ COPYPAGE ➤

## Keywords and parameters

There are no keywords or parameters.

## Example

In [Sample JCL showing the use of the COPYPAGE function](#), two existing page sets are copied onto two new page sets. The procedure for this is:

1. Set up the required DDnames, where:

### **CSQP0005, CSQP0006**

Identify the destination data sets. These DDnames are used by the FORMAT function.

## CSQS0005, CSQS0006

Identify the source data sets containing the two page sets you want to copy.

## CSQT0005, CSQT0006

Identify the destination data sets (page sets), but this time for the COPYPAGE function.

2. Format the destination data sets, referenced by DDnames CSQP0005 and CSQP0006, as page sets using the FORMAT function.
3. Copy the two existing page sets onto the new page sets using the COPYPAGE function.

```
//JOB LIB DD DISP=SHR,DSN=ANTZ.MQ.&VER..&LVL..OUT.SCSQANLE
// DD DISP=SHR,DSN=ANTZ.MQ.&VER..&LVL..OUT.SCSQAUTH
//*
//S1 EXEC PGM=IDCAMS
//* Delete any prior attempt, then allocate a new larger page set
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE 'VICY.MQ38.PAGE01.NEW' CLUSTER
DEFINE CLUSTER (NAME('VICY.MQ38.PAGE01.NEW') +
MODEL('VICY.MQ38.PAGE01') +
DATA CLAS(EXTENDED) +
LINEAR CYLINDERS(100,50))
//*
//MQMUTIL EXEC PGM=CSQUTIL,PARM='',REGION=4M
//* CSQUTIL
//* FORMAT acts on DDNAME like CSQPnnnn
//* optional, FORMAT PAGES(nnn) to force allocation and format of
//* secondary extents.
//* COPYPAGE copies from source, CSQSnnnn
//* to target, CSQTnnnn
//SYSPRINT DD SYSOUT=*
//CSQP0001 DD DISP=SHR,DSN=VICY.MQ38.PAGE01.NEW
//CSQS0001 DD DISP=SHR,DSN=VICY.MQ38.PAGE01
//CSQT0001 DD DISP=SHR,DSN=VICY.MQ38.PAGE01.NEW
//SYSIN DD *
FORMAT
COPYPAGE
//*
//RENAME EXEC PGM=IDCAMS
//* the cluster and data components must be renamed independently
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ALTER 'VICY.MQ38.PAGE01' NEWNAME('VICY.MQ38.PAGE01.OLD')
ALTER 'VICY.MQ38.PAGE01.DATA' +
NEWNAME('VICY.MQ38.PAGE01.OLD.DATA')
ALTER 'VICY.MQ38.PAGE01.NEW' +
NEWNAME('VICY.MQ38.PAGE01')
ALTER 'VICY.MQ38.PAGE01.NEW.DATA' +
NEWNAME('VICY.MQ38.PAGE01.DATA')
/*
```

Figure 15. Sample JCL showing the use of the COPYPAGE function

## Usage notes

1. You cannot use COPYPAGE on page sets of a queue manager that is running.
2. Using COPYPAGE involves stopping the queue manager. This results in the loss of nonpersistent messages.
3. Before you use COPYPAGE, the new data sets must be preformatted as page sets. To do this, use the FORMAT function, as shown in [Figure 15 on page 2748](#).
4. Ensure that the new (destination) data sets are larger than the old (source) data sets.
5. You cannot change the page set identifier (PSID) associated with a page set. For example, you cannot 'make' page set 03 become page set 05.
6. Failure of this function does not prevent other CSQUTIL functions from being attempted.
7. If you attempt to use the COPYPAGE function after the queue manager has terminated abnormally, the page sets might not have been closed properly. If a page set has not been closed properly, you cannot successfully run the COPYPAGE function against it.

To avoid this problem, run the AMS VERIFY command before using the COPYPAGE function. The AMS VERIFY command might produce error messages. However, it does close the page sets properly, so that the COPYPAGE function can complete successfully.

For more information about the AMS [VERIFY](#) command, see the *z/OS DFSMS Access Method Services Commands* manual.

8. See [Defining a page set to be larger than 4 GB](#) for information on using the EXTENDED attribute on the **DATACLAS** parameter.

## **Copying a page set and resetting the log (RESETPAGE) on z/OS**

The RESETPAGE function is like the COPYPAGE function except that it also resets the log information in the new page sets.

RESETPAGE lets you restart the queue manager from a known, valid set of page sets, even if the corresponding log data sets have been corrupted.

The source page sets for RESETPAGE must be in a consistent state. They must be either:

- Page sets that have been through a successful queue manager shutdown using the IBM MQ command STOP QMGR.
- Copies of page sets that have been through a successful stop.

The RESETPAGE function must not be run against copies of page sets made using fuzzy backup (see [Method 2: Fuzzy backup](#)), or against page sets that are from a queue manager that has terminated abnormally.

RESETPAGE either:

- Copies page sets on all data sets referenced by DDnames CSQS0000 through CSQS0099 to new data sets referenced by DDnames CSQT0000 through CSQT0099. If you use this function, modify the page set definition in the started task procedure to reflect the change of the name of the data set on which the new page set resides.
- Resets the log information in the page set referenced by DDnames CSQP0000 through CSQP0099.

For more information, see [Managing page sets](#).

### **Using the RESETPAGE function**

You can use the RESETPAGE function to update a set of consistent page sets so that they can be used with a set of new (clean) BSDS and log data sets to start the queue manager. You only have to use the RESETPAGE function if both copies of the log have been lost or damaged; you can restart from backup copies of page sets (and accept the resulting loss of data from the time the copies were made), or from your existing page sets.

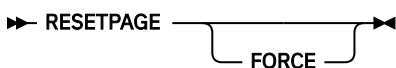
In this situation, use the RESETPAGE function on **all** the page sets of the affected queue manager. You must also create new BSDS and log data sets.

**Note:** Do not use the RESETPAGE function on a subset of the page sets known to IBM MQ.

If you run the RESETPAGE function against any page sets, but do not provide clean BSDS and log data sets for the queue manager, IBM MQ attempts to recover the logs from RBA zero, and treats the page sets as empty. For example, the following messages are produced if you attempt to use the RESETPAGE function to generate page sets zero, 1, 2, and 3 without providing a clean set of BSDS and log data sets:

```
CSQI021I +CSQ1 CSQIECUR PAGE SET 0 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 1 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 2 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 3 IS EMPTY. MEDIA RECOVERY STARTED
```

#### **Page set management (RESETPAGE)**



## Keywords and parameters

### FORCE

Specifies that the page sets specified by DDnames CSQP0000 through CSQP00nn are to be reset in place.

If FORCE is not specified, the page sets specified by DDnames CSQS0000 through CSQS00nn are copied to new page sets specified by DDnames CSQT0000 through CSQT00nn. This is the default.

You should take a copy of the page sets first. See [backing up page sets](#) for sample JCL to perform this operation.

### Example

An existing page set, referenced by DDname CSQS0007, is copied to a new data set referenced by DDname CSQT0007. The new data set, which is also referenced by DDname CSQP0007, is already formatted as a page set before the RESETPAGE function is called.

```
//RESTPAGE EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//CSQP0007 DD DISP=OLD,DSN=pageset.newname7
//CSQS0007 DD DISP=OLD,DSN=pageset.oldname7
//CSQT0007 DD DISP=OLD,DSN=pageset.newname7
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
* Format new data set, CSQP0007, as page set
FORMAT
* Copy page set CSQS0007 to CSQT0007 and reset it
RESETPAGE
/*
```

Figure 16. Sample JCL showing the use of the RESETPAGE function

## Usage notes

1. Do not use the RESETPAGE function against page sets after the queue manager has terminated abnormally. Page sets from a queue manager that terminated abnormally will probably contain inconsistent data; using RESETPAGE on page sets in this state leads to data integrity problems.
2. You cannot use RESETPAGE on page sets belonging to a queue manager that is running.
3. Before you use RESETPAGE, the new data sets must be pre-formatted as page sets. To do this, use the FORMAT function, as shown in [Figure 16 on page 2750](#).
4. Ensure that the new (destination) data sets are larger than the old (source) data sets.
5. You cannot change the page set identifier (PSID) associated with a page set. For example, you cannot 'make' page set 03 become page set 05.
6. Failure of this function does not prevent other CSQUTIL functions from being attempted.

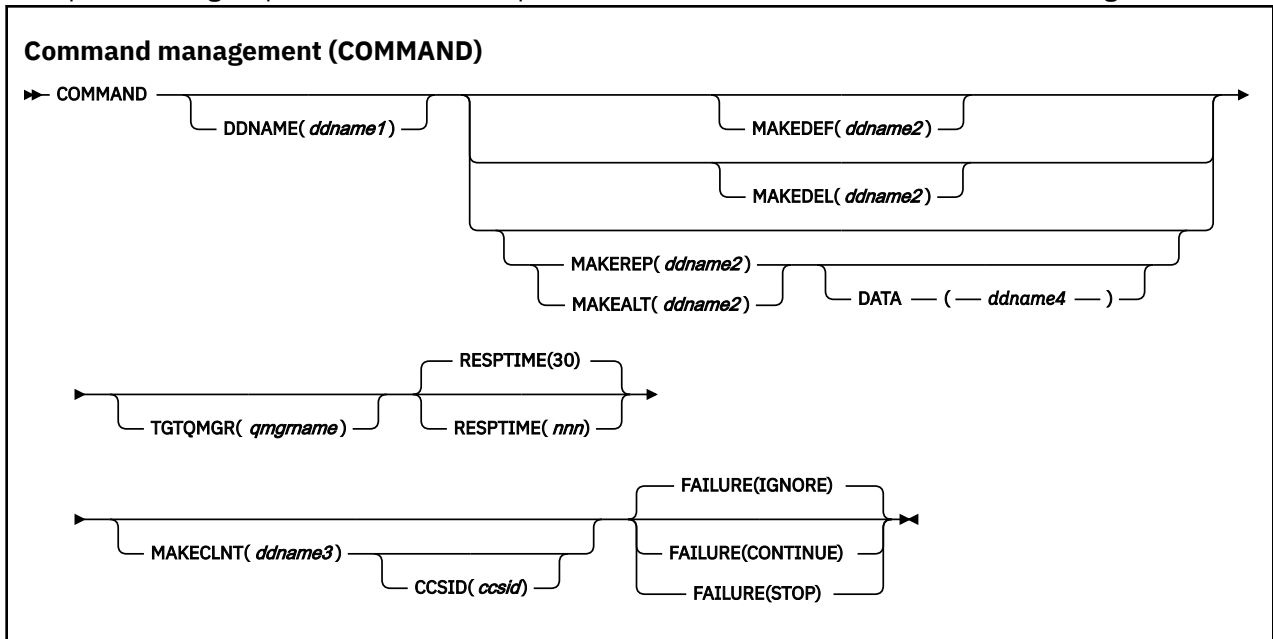
## Using the COMMAND function of CSQUTIL on z/OS

You can use the COMMAND function of CSQUTIL to direct commands to the queue manager.

Use the COMMAND function to:

1. Pass commands from an input data set to the queue manager.
2. Produce a list of DEFINE commands that describe the objects in a queue manager. The commands can be used to keep a record of the object definitions or to regenerate all or part of a queue manager's objects as part of a migration from one queue manager to another.
3. Produce a list of commands to change or delete a set of objects in a queue manager.
4. Make a client channel definition file.

The queue manager specified in the PARM parameter of the EXEC statement must be running.



- [Keywords and parameters](#)
- [Examples](#)
- [Usage notes for CSQUTIL COMMAND](#)

If you use **FAILURE (IGNORE)** the job step always obtains return code 0.

If you use **FAILURE (STOP)** or **FAILURE (CONTINUE)** the job step obtains return code 8 if there were any non zero return codes from the statements.

You should use **FAILURE (STOP)** or **FAILURE (CONTINUE)** to report any errors in the definitions.

## Keywords and parameters

### DDNAME(*ddname1*)

Specifies that the commands are to be read from a named input data set. If this keyword is omitted, the default DDname, CSQUCMD, is used.

*ddname1* specifies the DDname that identifies the input data set from which commands are to be read.

### MAKEDEF(*ddname2*), MAKEDEL(*ddname2*), MAKEREP(*ddname2*), MAKEALT(*ddname2*)

Specify that commands are to be generated from any DISPLAY object commands in the input data set.

The commands that are generated are:

#### MAKEDEF

DEFINE NOREPLACE, with all the attributes and values returned by the DISPLAY commands. For the queue manager object, an ALTER command is generated with all the attributes and values. For channel authentication records, a SET command is generated.

Both CSQUTIL SDEFS and the CSQUTIL COMMAND with the MAKEDEF option can be used to produce a set of MQSC commands to re-create the objects currently defined in the queue manager.

The difference between the two is that CSQUTIL COMMAND must be run against an active queue manager and is most appropriate for regular backup of object definitions, whereas CSQUTIL SDEFS can be used to re-create definitions for a queue manager that is not currently running. This makes the CSQUTIL SDEFS option more appropriate for recovery scenarios.

**MAKEDEL**

DELETE. For local queues, NOPURGE is used. For channel authentication records, a SET command with ACTION(REMOVE) is used

**MAKEREP**

DEFINE REPLACE, with any keywords and values from the data set specified by the DATA keyword. For channel authentication records, a SET command with ACTION(REPLACE) is used.

**MAKEALT**

ALTER, with any keywords and values from the data set specified by the DATA keyword. For channel authentication records, a SET command with ACTION(REPLACE) is used.

Only one of these keywords may be specified. If these keywords are omitted, no commands are generated.

*ddname2* specifies the DDname that identifies the output data set in which the DEFINE, DELETE or ALTER commands are to be stored. The data set should be RECFM=FB, LRECL=80. This data set can then be used as input for a later invocation of the COMMAND function or it can be incorporated into the initialization data sets CSQINP1 and CSQINP2.

**DATA(*ddname4*)**

*ddname4* specifies a data set from which command keywords and values are to be read, and appended to each command generated for MAKEREP or MAKEALT.

**TGTQMGR(*qmgrname*)**

Specifies the name of the z/OS queue manager where you want the commands to be performed. This option is not supported for use with queue managers on distributed platforms. You can specify a target queue manager that is not the one you connect to. In this case, you would normally specify the name of a remote queue manager object that provides a queue manager alias definition (the name is used as the *ObjectQMgrName* when opening the command input queue). To do this, you must have suitable queues and channels set up to access the remote queue manager.

The default is that commands are performed on the queue manager to which you are connected, as specified in the PARM field of the EXEC statement.

**RESPTIME(*nnn*)**


Specifies the time in seconds to wait for a response to each command, in the range 5 through 999.

The default is 30 seconds.

**MAKECLNT(*ddname3*)**

Specifies that a client channel definition file is generated from any DISPLAY CHANNEL commands in the input data set that return information about client-connection channels, and any DISPLAY AUTHINFO commands that return information about authentication information objects for which the LDAPUSER and LDAPPWD attributes are not set.

If this keyword is omitted, no file is generated.

**Important:**  From IBM MQ 9.1, the MAKECLNT attribute is deprecated.

*ddname3* specifies the DDname that identifies the output data set in which the generated file is to be stored; the data set should be RECFM=U, LRECL=6144. The file can then be downloaded as binary data to the client machine by a suitable file transfer program.

**CCSID(*ccsid*)**

Specifies the coded character set identifier (CCSID) that is to be used for the data in a client channel definition file. The value must be in the range 1 through 65535; the default is 437. You can only specify CCSID if you also specify MAKECLNT.

**Note:** IBM MQ assumes that the data is to be in ASCII, and that the encoding for numeric data is to be MQENC\_INTEGER\_REVERSED.



## FAILURE

Specifies what action to take if an IBM MQ command that is issued fails to execute successfully. Values are:

## IGNORE

Ignore the failure; continue reading and issuing commands, and treat the COMMAND function as being successful. This is the default.

## CONTINUE

Read and issue any remaining commands in the input data set, but treat the COMMAND function as being unsuccessful.

## STOP

Do not read or issue any more commands, and treat the COMMAND function as being unsuccessful.

## Examples

This section gives examples of using the COMMAND function for the following:

- [“Issuing commands” on page 2753](#)
- [“Making a list of DEFINE commands” on page 2753](#)
- [“Making a list of ALTER commands” on page 2754](#)
- [“Making a client channel definition file” on page 2755](#)

### Issuing commands

In [Figure 17 on page 2753](#), the data sets referenced by DDnames CSQUCMD and OTHER contain sets of commands. The first COMMAND statement takes commands from the default input data set MY.COMMANDS(COMMAND1) and passes them to the queue manager. The second COMMAND statement takes commands from the input data set MY.COMMANDS(OTHER1), which is referenced by DDname OTHER, and passes them to the queue manager.

```
//COMMAND EXEC PGM=CSQUTIL,PARM='CSQ1'  
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE  
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH  
//CSQUCMD DD DSN=MY.COMMANDS(COMMAND1),DISP=SHR  
//OTHER DD DSN=MY.COMMANDS(OTHER1),DISP=SHR  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
* THE NEXT STATEMENT CAUSES COMMANDS TO BE READ FROM CSQUCMD DDNAME  
COMMAND  
* THE NEXT SET OF COMMANDS WILL COME FROM 'OTHER' DDNAME  
COMMAND DDNAME(OTHER)  
* THE NEXT STATEMENT CAUSES COMMANDS TO BE READ FROM CSQUCMD  
* DDNAME AND ISSUED ON QUEUE MANAGER CSQ2 WITH A RESPONSE TIME  
* OF 10 SECONDS  
COMMAND TGTQMR(CSQ2) RESPTIME(10)  
/*
```

*Figure 17. Sample JCL for issuing IBM MQ commands using CSQUTIL*

### Making a list of DEFINE commands

In [Figure 18 on page 2754](#), the data set referenced by DDname CMDINP contains a set of DISPLAY commands. These DISPLAY commands specify generic names for each object type (except the queue manager itself). If you run these commands, a list is produced containing all the IBM MQ objects. In these DISPLAY commands, the ALL keyword is specified to ensure that all the attributes of all the objects are included in the list, and that all queue sharing group dispositions are included.

**Note:** Failing to issue DISPLAY STGCLASS as the first command can result in a set of definitions that will not be successfully processed by the queue manager, as STGCLASS definitions must be defined

before the associated queue objects are defined. MAKEDEFS generate output based on the order of the input DISPLAY commands.

The MAKEDEF keyword causes this list to be converted into a corresponding set of DEFINE NOREPLACE commands (ALTER for the queue manager). These commands are put into a data set referenced by the **ddname2** parameter of the MAKEDEF keyword, that is, OUTPUT1. If you run this set of commands, IBM MQ regenerates all the object definitions in the queue manager.

```
//QDEFS EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTPUT1 DD DISP=OLD,DSN=MY.COMMANDS(DEFS)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(CMDINP) MAKEDEF(OUTPUT1)
/*
//CMDINP DD *
DISPLAY STGCLASS(*) ALL QSGDISP(QMGR)
DISPLAY STGCLASS(*) ALL QSGDISP(GROUP)
DISPLAY CFSTRUCT(*) ALL

DISPLAY QUEUE(*) ALL QSGDISP(QMGR)
DISPLAY QUEUE(*) ALL QSGDISP(GROUP)
DISPLAY QUEUE(*) ALL QSGDISP(SHARED)
DISPLAY TOPIC(*) ALL QSGDISP(QMGR)
DISPLAY TOPIC(*) ALL QSGDISP(GROUP)
DISPLAY NAMELIST(*) ALL QSGDISP(QMGR)
DISPLAY NAMELIST(*) ALL QSGDISP(GROUP)
DISPLAY PROCESS(*) ALL QSGDISP(QMGR)
DISPLAY PROCESS(*) ALL QSGDISP(GROUP)
DISPLAY CHANNEL(*) ALL QSGDISP(QMGR)
DISPLAY CHANNEL(*) ALL QSGDISP(GROUP)
DISPLAY AUTHINFO(*) ALL QSGDISP(QMGR)
DISPLAY AUTHINFO(*) ALL QSGDISP(GROUP)
DISPLAY CHLAUTH('*') ALL
DIS SUB(*) SUBTYPE(ADMIN) ALL DISTYPE(DEFINED)

DISPLAY QMGR ALL

/*
```

Figure 18. Sample JCL for using the MAKEDEF option of the COMMAND function

### Making a list of ALTER commands

In Figure 19 on page 2755, the data set referenced by DDname CMDINP contains a DISPLAY command that will produce a list of all local queues with names beginning "ABC".

The MAKEALT keyword causes this list to be converted into a corresponding set of ALTER commands, each of which includes the data from the data set referenced by DDname CMDALT. These commands are put into a data set referenced by the ddname2 parameter of the MAKEALT keyword, that is, OUTPUTA. If you run this set of commands, all the local queues with names beginning "ABC" will be disabled for PUT and GET.

```

//QALTS EXEC PGM=CSQUTIL,PARM='CSQ1 '
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTPUTA DD DISP=OLD,DSN=MY.COMMANDS(ALTS)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(CMDINP) MAKEALT(OUTPUTA) DATA(CMDALT)
/*
//CMDINP DD *
DISPLAY QLOCAL(ABC*)
/*
//CMDALT DD *
PUT(DISABLED) +
GET(DISABLED)
/*

```

Figure 19. Sample JCL for using the MAKEALT option of the COMMAND function

### Making a client channel definition file

In Figure 20 on page 2755, the data set referenced by DDname CMDCHL contains a DISPLAY CHANNEL command and a DISPLAY AUTHINFO command. The DISPLAY commands specify a generic name and the ALL keyword is specified to ensure that all the attributes are included.

The MAKECLNT keyword converts these attributes into a corresponding set of client channel definitions. These are put into a data set referenced by the *ddname3* parameter of the MAKECLNT keyword, that is, OUTCLNT, which is ready to be downloaded to the client machine.

```

//CLIENT EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTCLNT DD DISP=OLD,DSN=MY.CLIENTS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(CMDCHL) MAKECLNT(OUTCLNT)
/*
//CMDCHL DD *
DISPLAY CHANNEL(*) ALL TYPE(CLNTCONN)
DISPLAY AUTHINFO(*) ALL
/*

```

Figure 20. Sample JCL for using the MAKECLNT option of the COMMAND function

### Usage notes for CSQUTIL COMMAND

1. The rules for specifying commands in the input data set are the same as for the initialization data sets:

- The data set must have a record length of 80.
- Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
- Records with an asterisk (\*) in column 1 are interpreted as comments and are ignored.
- Blank records are ignored.
- Each command must start on a new record.
- A trailing - means continue from column 1 of the next record.
- A trailing + means continue from the first non-blank column of the next record.
- The maximum number of characters permitted in a command is 32 762.

With the additional rule:

- A semicolon (;) can be used to terminate a command; the remaining data in the record is ignored.

See [Running MQSC commands from text files](#) for more information about the rules for building IBM MQ commands.

2. The output from a [“DISPLAY QMGR \(display queue manager settings\)”](#) on page 765 command contains all the queue manager attributes. Using the **DISPLAY QMGR** command as part of MAKEDEF might generate an ALTER command that cannot be issued before the channel initiator is active.

Since setting PSCLUS(DISABLED) can only be done if the channel initiator is active, it might be necessary to modify the resulting ALTER command so that it does not attempt to set PSCLUS(DISABLED) until the channel initiator is active.

3. If you specify the MAKEDEF keyword:

- In the input data set, the DISPLAY commands for objects must contain the ALL parameter so that the complete definition of each object is produced. See [Figure 18 on page 2754](#).
- To obtain a complete definition, you must DISPLAY the following:
  - queues
  - topic
  - namelists
  - process definitions
  - channels
  - storage classes
  - authentication information objects
  - CF structures
  - channel authentication records
  - queue manager

**Note:** DEFINE commands are not generated for any local queues that can be identified as dynamic, or for channels that were defined automatically.

- Do not specify the same MAKEDEF data set for more than one COMMAND function, unless its DD statement specifies a sequential data set with DISP=MOD.

4. If you specify the MAKEREP, MAKEALT, or MAKEDEL keywords:

- In the input data set, include DISPLAY commands that select the set of objects for which you want to generate commands.
- For MAKEREP and MAKEALT, the data (if any) from the data set specified by the DATA keyword is appended to each generated command, exactly as entered. The format of the data set and the rules for specifying command data are the same as for the command input data set. Because the same data is appended to each command, if you want to process several sets of objects, you will need to use several separate COMMAND functions, each with a different DATA data set.
- Commands are not generated for channels that were defined automatically.

5. If you specify the MAKEDEF, MAKEREP, MAKEALT, or MAKEDEL keywords, commands are generated only for objects reported by the target queue manager (as specified by the TGTQMGR keyword or defaulted), even if CMDSCOPE is used in the DISPLAY commands. To generate commands for several queue managers in a queue sharing group, use a separate COMMAND function for each.

In a queue sharing group, queues, processes, channels, storage classes and authentication information objects should each have two DISPLAY commands, one with QSGDISP(QMGR) and one with QSGDISP(GROUP). Queues should have a third with QSGDISP(SHARED). It is not necessary to specify QSGDISP(COPY) because the required commands will be generated automatically when the commands for objects with QSGDISP(GROUP) are issued.

6. Do not specify the same MAKEDEF, MAKEREP, MAKEALT, or MAKEDEL data set for more than one COMMAND function, unless its DD statement specifies a sequential data set with DISP=MOD.

7. If you specify the MAKECLNT keyword:

- In the input data set, the display commands for channels and authentication information objects must contain the ALL parameter so that the complete definition of each channel and authentication information object is produced.
  - If the DISPLAY commands return information for a particular channel more than once, only the last set of information is used.
  - Do not specify the same client definition file data set for more than one COMMAND function, unless its DD statement specifies a sequential data set with DISP=MOD.
8. The results of DISPLAY commands used in conjunction with MAKEDEF, MAKEREP, MAKEALT, MAKEDEL or MAKECLNT are also sent to SYSPRINT.
  9. If you specify the FAILURE keyword, a command is determined to be a success or failure according to the codes returned in message CSQN205I. If the return code is 00000000 and the reason code is 00000000 or 00000004, it is a success; for all other values it is a failure.
  10. The COMMAND function is determined to be a success only if both:
    - All the commands in the input data set are read and issued and get a response from IBM MQ, regardless of whether the response indicates successful execution of the command or not.
    - Every command issued executes successfully, if FAILURE(CONTINUE) or FAILURE(STOP) is specified.

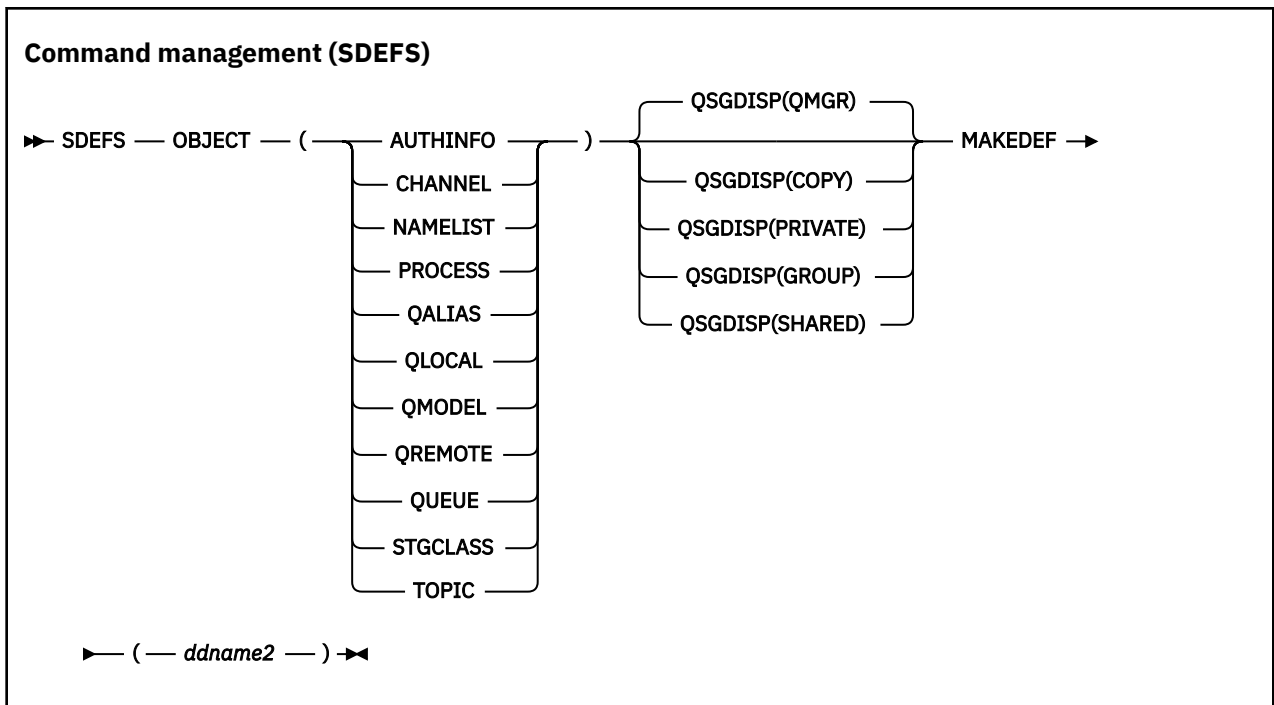
If COMMAND fails, no further CSQUTIL functions are attempted.
  11. You need the necessary authority to use command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*) and to use the IBM MQ commands that you want to issue.

## **Producing a list of IBM MQ define commands (SDEFS) on z/OS**

You can use the SDEFS function of CSQUTIL to produce a list of DEFINE commands describing the objects in your queue manager or queue sharing group.

Both CSQUTIL SDEFS and the CSQUTIL COMMAND with the MAKEDEF option can be used to produce a set of MQSC commands to re-create the objects currently defined in the queue manager.

The difference between the two is that CSQUTIL COMMAND must be run against an active queue manager and is most appropriate for regular backup of object definitions, whereas CSQUTIL SDEFS can be used to re-create definitions for a queue manager that is not currently running. This makes the CSQUTIL SDEFS option more appropriate for recovery scenarios.



- [Keywords and parameters](#)
- [Examples](#)
- [Usage notes](#)

## Keywords and parameters

### OBJECT

Specifies the type of object to be listed.

A value of QUEUE lists queues of all types, as if you had specified QALIAS, QLOCAL, QMODEL and QREMOTE.

### QSGDISP

Specifies from where the object definition information is obtained. Depending on how the object has been defined, this information is either:

- On the page set zero referred to by the CSQP0000 DD statement, or
- In a Db2 shared repository.

Permitted values are shown in [Table 383](#) on page 2758.

<i>Table 383. SDEFS QSGDISP parameters and their actions</i>	
<b>QSGDISP parameter</b>	<b>What the SDEFS utility does</b>
QMGR	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1) Only objects defined with QSGDISP(QMGR) are included.
COPY	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1) Only objects defined with QSGDISP(COPY) are included.

<i>Table 383. SDEFS QSGDISP parameters and their actions (continued)</i>	
<b>QSGDISP parameter</b>	<b>What the SDEFS utility does</b>
PRIVATE	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1) Both QSGDISP(QMGR) and QSGDISP(COPY) objects are included.
GROUP	Creates DEFINE statements for the specified object type from definitions held on Db2 resource definition tables for the specified queue sharing group. Only objects defined with QSGDISP(GROUP) are included. No CSQP0000 DD statement is required; the Db2 subsystem specified at object definition is accessed. The Db2 library db2qual.SDSNLOAD is required.
SHARED	Creates DEFINE statements for all local queues defined with QSGDISP(SHARED) by accessing the Db2 resource definition table for the specified queue sharing group. This parameter is permitted only with OBJECT(QLOCAL) or OBJECT(Queue). No CSQP0000 DD statement is required; the Db2 subsystem specified at object definition is accessed. The Db2 library db2qual.SDSNLOAD is required.

**Notes:**

1. Because only page set zero is accessed, you must ensure that the queue manager is not running.

**MAKEDEF( ddname2 )**

Specifies that define commands generated for the object are to be placed in the output data set identified by the DDname. The data set should be RECFM=FB, LRECL=80. This data set can then be used as input for a later invocation of the COMMAND function or it can be incorporated into the initialization data sets CSQINP1 and CSQINP2.

The commands generated are DEFINE NOREPLACE, with all the attributes and values for the object.

**Note:** DEFINE commands are not generated for any local queues that can be identified as dynamic, or for channels that were defined automatically.

**Examples**

```
//SDEFS EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=th1qual.SCSQANLE
// DD DISP=SHR,DSN=th1qual.SCSQAUTH
//CSQP0000 DD DISP=OLD,DSN=pageset.dsname0
//OUTPUT1 DD DISP=OLD,DSN=MY.COMMANDS(DEFS)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SDEFS OBJECT(Queue) MAKEDEF(OUTPUT1)
/*
```

*Figure 21. Sample JCL for the SDEFS function of CSQUTIL*

```

//SDEFS EXEC PGM=CSQUTIL,PARM='Qsgname,Dsgname,Db2name'
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
// DD DISP=SHR,DSN=db2qua1.SDSNLOAD
//OUTPUT1 DD DISP=OLD,DSN=MY.COMMANDS(DEFS)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SDEFS OBJECT(QLOCAL) QSGDISP(SHARED) MAKEDEF(OUTPUT1)
/*

```

Figure 22. Sample JCL for the SDEFS function of CSQUTIL for objects in the Db2 shared repository

```

//CSQUTIL JOB CLASS=A,MSGCLASS=H,NOTIFY=&SYSUID,REGION=0M
//PS00 EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQAUTH
// DD DISP=SHR,DSN=thlqua1.SCSQANLE
//CSQP0000 DD DISP=OLD,DSN=pageset.dsname0
//OUTPUT1 DD DISP=OLD,DSN=MY.COMMANDS(CHANNEL)
//OUTPUT2 DD DISP=OLD,DSN=MY.COMMANDS(AUTHINFO)
//OUTPUT3 DD DISP=OLD,DSN=MY.COMMANDS(NAMELIST)
//OUTPUT4 DD DISP=OLD,DSN=MY.COMMANDS(PROCESS)
//OUTPUT5 DD DISP=OLD,DSN=MY.COMMANDS(QALIAS)
//OUTPUT6 DD DISP=OLD,DSN=MY.COMMANDS(QLOCAL)
//OUTPUT7 DD DISP=OLD,DSN=MY.COMMANDS(QMODEL)
//OUTPUT8 DD DISP=OLD,DSN=MY.COMMANDS(QREMOTE)
//OUTPUT9 DD DISP=OLD,DSN=MY.COMMANDS(QQUEUE)
//OUTPUT0 DD DISP=OLD,DSN=MY.COMMANDS(STGCLASS)
//OUTPUTA DD DISP=OLD,DSN=MY.COMMANDS(TOPIC)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SDEFS OBJECT(CHANNEL) MAKEDEF(OUTPUT1)
SDEFS OBJECT(AUTHINFO) MAKEDEF(OUTPUT2)
SDEFS OBJECT(NAMELIST) MAKEDEF(OUTPUT3)
SDEFS OBJECT(PROCESS) MAKEDEF(OUTPUT4)
SDEFS OBJECT(QALIAS) MAKEDEF(OUTPUT5)
SDEFS OBJECT(QLOCAL) MAKEDEF(OUTPUT6)
SDEFS OBJECT(QMODEL) MAKEDEF(OUTPUT7)
SDEFS OBJECT(QREMOTE) MAKEDEF(OUTPUT8)
SDEFS OBJECT(QQUEUE) MAKEDEF(OUTPUT9)
SDEFS OBJECT(STGCLASS) MAKEDEF(OUTPUT0)
SDEFS OBJECT(TOPIC) MAKEDEF(OUTPUTA)
/*

```

Figure 23. Sample JCL for the SDEFS function of CSQUTIL, when recovering all objects from a valid page set zero

## Usage notes

1. For local definitions, do not use SDEFS for a queue manager that is running because results will be unpredictable. You can avoid doing this accidentally by using DISP=OLD in the CSQP0000 DD statement. For shared or group queue definitions, this does not matter because the information is derived from Db2.
2. When you use SDEFS for local queues you do not need to specify a queue manager name. However, for shared and group queue definitions, a queue manager name is required to access Db2.
3. To use the SDEFS function more than once in a job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.
4. If the SDEFS function fails, no further CSQUTIL functions are attempted.
5. The SDEFS function does not support the CHLAUTH, SUB, CFSTRUCT or QMGR objects. To back these objects up, use the [CSQUTIL COMMAND](#) function.

## Related concepts

[“IBM MQ utility program \(CSQUTIL\) on z/OS” on page 2739](#)





## DEFTYPE

Specifies whether to copy dynamic queues:

### ALL

Copy all queues; this is the default.

### PREDEFINED

Do not include dynamic queues; this is the same set of queues that are selected by the COMMAND and SDEFS functions with the MAKEDEF parameter.

## DDNAME(ddname)

Specifies that the messages are to be copied to a named data set. If this keyword is omitted, the default DDname, CSQUOUT, is used. The keyword DDname can be abbreviated to DD.

*ddname* specifies the DDname of the destination data set, which is used to store the messages. The record format of this data set must be variable block spanned (VBS).

## Example

```
//COPY EXEC PGM=CSQUTIL,PARM='CSQ1',REGION=0M
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTPUTA DD DSN=SAMPLE.UTILITY.COPYA,DISP=(NEW,CATLG),
// SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
// DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQUOUT DD DSN=SAMPLE.UTILITY.COPY3,DISP=(NEW,CATLG),
// SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
// DCB=(RECFM=VBS,BLKSIZE=23200)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
* COPY WHOLE PAGE SET TO 'CSQUOUT'
COPY PSID(03)
* COPY ONE QUEUE TO 'OUTPUT'
COPY QUEUE(ABC123A) DDNAME(OUTPUTA)
/*
```

Figure 24. Sample JCL for the CSQUTIL COPY functions

## Usage notes

1. The queues involved must not be in use when the function is started.
2. If you want to operate on a range of page sets, repeat the COPY function for each page set.
3. The function operates only on local queues.
4. A COPY PSID function is considered successful only if it successfully copies all the queues on the page set.
5. If you try to copy an empty queue (either explicitly by COPY QUEUE or because there are one or more empty queues on a page set that you are copying), data indicating this is written to the sequential data set, and the copy is considered to be a success. However, if you attempt to copy a nonexistent queue, or a page set containing no queues, the COPY function fails, and no data is written to the data set.
6. If COPY fails, no further CSQUTIL functions are attempted.
7. To use the COPY function more than once in the job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.
8. You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*), to use the DISPLAY QUEUE and DISPLAY STGCLASS MQSC commands, and to open the queues that you want to copy with the MQOO\_INPUT\_EXCLUSIVE and MQOO\_BROWSE options.
9. For the **REGION** parameter, a value of 0M means that the job is allowed to have the amount of storage it needs. However, if a job tries to acquire too much storage, it might impact other jobs in the system.

You must ideally look to limit the REGION size and specify an absolute maximum value that the job is allowed to acquire.

## **z/OS** Copying queues into a data set while the queue manager is not running (SCOPY) on z/OS

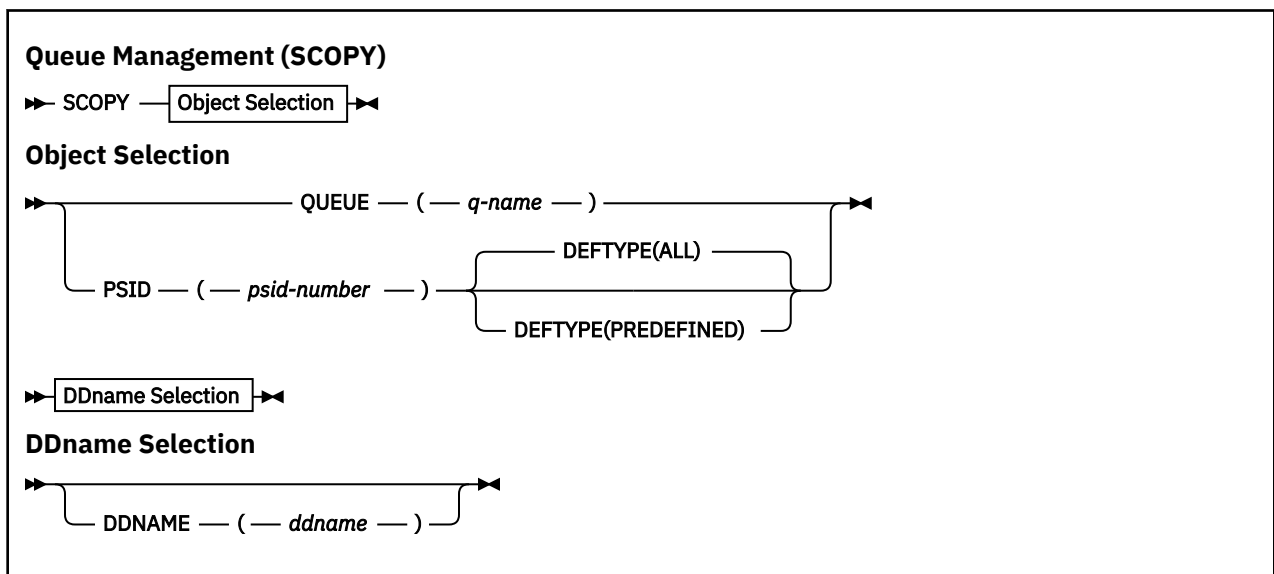
You can use the SCOPY function of CSQUTIL to copy queued messages to a sequential data set when the queue manager is not running, without destroying any messages in the original queues.

The scope of the SCOPY function is determined by the keyword that you specify in the first parameter. You can either copy all the messages from a named queue, or all the messages from all the queues on a named page set.

Use the complementary function, LOAD, to restore the messages to their queues.

To use the SCOPY function, DDname CSQP0000 must specify the data set with page set zero for the subsystem required.

**Note:** The SCOPY function does not operate on shared queues.



- [Keywords and parameters](#)
- [Example](#)
- [Usage notes](#)

### Keywords and parameters

#### **QUEUE(*q-name*)**

Specifies that messages in the named queue are to be copied. The keyword QUEUE can be abbreviated to Q.

*q-name* specifies the name of the queue to be copied. This name is case-sensitive.

DDname CSQP00 *nn* must specify the data set with page set *nn* for the subsystem required, where *nn* is the number of the page set where the queue resides.

#### **PSID(*psid-number*)**

Specifies that all the messages in all the queues in the specified page set are to be copied.

*psid-number* is the page set identifier, which specifies the page set to be used. This identifier is a two-digit integer (whole number) representing a single page set.

DDname CSQP00 *psid-number* must specify the data set with the required page set for the subsystem required.

## DEFTYPE

Specifies whether to copy dynamic queues:

### ALL

Copy all queues; this is the default.

### PREDEFINED

Do not include dynamic queues; this is the same set of queues that are selected by the COMMAND and SDEFS functions with the MAKEDEF parameter.

This parameter is only valid if you specify PSID.

## DDNAME(ddname)

Specifies that the messages are to be copied to a named data set. If this keyword is omitted, the default DDname, CSQUOUT, is used. The keyword DDname can be abbreviated to DD.

*ddname* specifies the DDname of the destination data set, which is used to store the messages. The record format of this data set must be variable block spanned (VBS).

Do not specify the same DDname on more than one SCOPY statement, unless its DD statement specifies a sequential data set with DISP=MOD.

## Example

```
//SCOPY EXEC PGM=CSQUTIL,REGION=0M
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUT
//OUTPUTA DD DSN=SAMPLE.UTILITY.COPYA,DISP=(NEW,CATLG),
// SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
// DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQUOUT DD DSN=SAMPLE.UTILITY.COPY3,DISP=(NEW,CATLG),
// SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
// DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQP0000 DD DISP=OLD,DSN=pageset.dsname0
//CSQP0003 DD DISP=OLD,DSN=pageset.dsname3
//CSQP0006 DD DISP=OLD,DSN=pageset.dsname6
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
* COPY WHOLE PAGE SET TO 'CSQUOUT'
SCOPY PSID(03)
* COPY ONE QUEUE TO 'OUTPUT' - QUEUE IS ON PAGE SET 6
SCOPY QUEUE(ABC123A) DDNAME(OUTPUTA)
/*
```

Figure 25. Sample JCL for the CSQUTIL SCOPY functions

## Usage notes

1. Do not use SCOPY for a queue manager that is running because results are unpredictable. You can avoid doing this accidentally by using DISP=OLD in the page set DD statement.
2. When you use SCOPY, you do not need to specify a queue manager name.
3. If you want to operate on a range of page sets, repeat the SCOPY function for each page set.
4. The function operates only on local queues and only for persistent messages.
5. An SCOPY PSID function is considered successful only if it successfully copies all the queues on the page set. If an empty queue is processed, data indicating this is written to the sequential data set. If the page set has no queues, the SCOPY function fails, and no data is written to the data set.
6. If you try to copy an empty queue explicitly by SCOPY QUEUE, data indicating this is written to the sequential data set, and the copy is considered to be a success. However, if you attempt to copy a nonexistent queue, the SCOPY function fails, and no data is written to the data set.
7. If the SCOPY function fails, no further CSQUTIL functions are attempted.

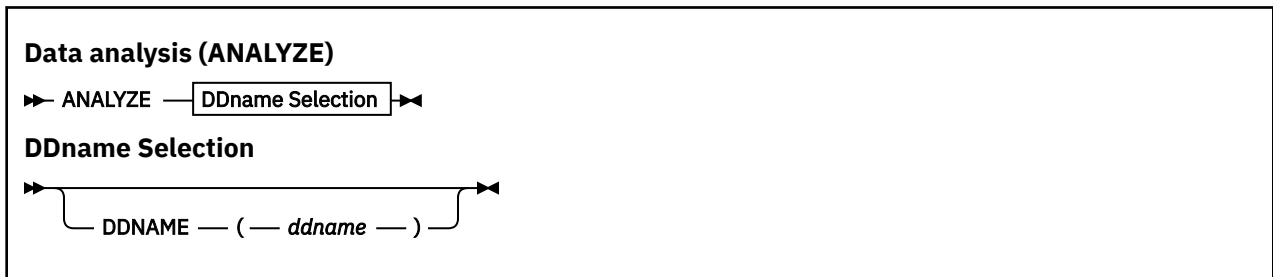
8. To use the SCOPY function more than once in the job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.
9. For the **REGION** parameter, a value of 0M means that the job is allowed to have the amount of storage it needs. However, if a job tries to acquire too much storage, it might impact other jobs in the system. You must ideally look to limit the REGION size and specify an absolute maximum value that the job is allowed to acquire.

## Analyzing the queue data copied to a data set by COPY or SCOPY using ANALYZE on z/OS

Use this topic to understand analyzing the queue data copied to a data set by COPY or SCOPY.

This function reads and analyzes a data set (created using COPY or SCOPY), and for each queue, displays:

- queue name
- number of messages for the queue
- total length of the messages



- [“Keywords and parameters” on page 2765](#)
- [“Example” on page 2765](#)
- [“Usage notes” on page 2765](#)

### Keywords and parameters

#### DDNAME(*ddname*)

Specifies the data set to be processed. This keyword can be abbreviated to DD.

*ddname* specifies the DDname that identifies the destination data set of a prior COPY or SCOPY operation. This name is not case sensitive, and can be up to eight characters long.

### Example

```
//LOAD EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTPUTA DD DSN=MY.UTILITY.OUTPUTA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ANALYZE DDNAME(OUTPUTA)
```

Figure 26. Sample JCL for the CSQUTIL ANALYZE function

### Usage notes

1. If you omit DDname(*ddname*) the default DDname, CSQUINP, is used.

## Emptying a queue of all messages (EMPTY) on z/OS

You can use the EMPTY function of CSQUTIL to delete all messages from a named queue or all the queues on a page set.

The queue manager must be running. The scope of the function is determined by the keyword that you specify in the first parameter.

Use this function with care. Only delete messages of which copies have already been made.

**Note:** See “Syncpoints” on page 2741 for information about how to avoid problems with duplicate messages if this function fails.



- [Keywords and parameters](#)
- [Example](#)
- [Usage notes](#)

### Keywords and parameters

You must specify the scope of the EMPTY function. Choose one of these:

#### **QUEUE(*q-name*)**

Specifies that messages are to be deleted from a named queue. This keyword can be abbreviated to Q.

*q-name* specifies the name of the queue from which messages are to be deleted. This name is case sensitive.

#### **PSID(*psid-number*)**

Specifies that all the messages are to be deleted from all queues in the named page set.

*psid-number* specifies the page-set identifier. This identifier is a two-digit integer (whole number) representing a single page set.

### Example

```
//EMPTY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
EMPTY QUEUE (SPARE)
EMPTY PSID(66)
/*
```

Figure 27. Sample JCL for the CSQUTIL EMPTY function

### Usage notes

1. The queues involved must not be in use when the function is invoked.

2. This function operates only on local queues.
3. If you want to operate on a range of page sets, repeat the EMPTY function for each page set.
4. You cannot empty the system-command input queue (SYSTEM.COMMAND.INPUT).
5. An EMPTY PSID function is considered successful only if it successfully empties all the queues on the page set.
6. If you empty a queue that is already empty (either explicitly by EMPTY QUEUE or because there are one or more empty queues on a page set that you are emptying), the EMPTY function is considered to be a success. However, if you attempt to empty a nonexistent queue, or a page set containing no queues, the EMPTY function fails.
7. If EMPTY fails or is forced to take a syncpoint, no further CSQUTIL functions are attempted.
8. You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*), to use the DISPLAY QUEUE and DISPLAY STGCLASS MQSC commands, and to use the IBM MQ API to get messages from the queues that you want to empty.

### Related concepts

[“Invoking the IBM MQ utility program on z/OS” on page 2740](#)

Use this topic to understand how to invoke CSQUTIL, the format of its parameters, and its return codes.

## Restoring messages from a data set to a queue (LOAD) on z/OS

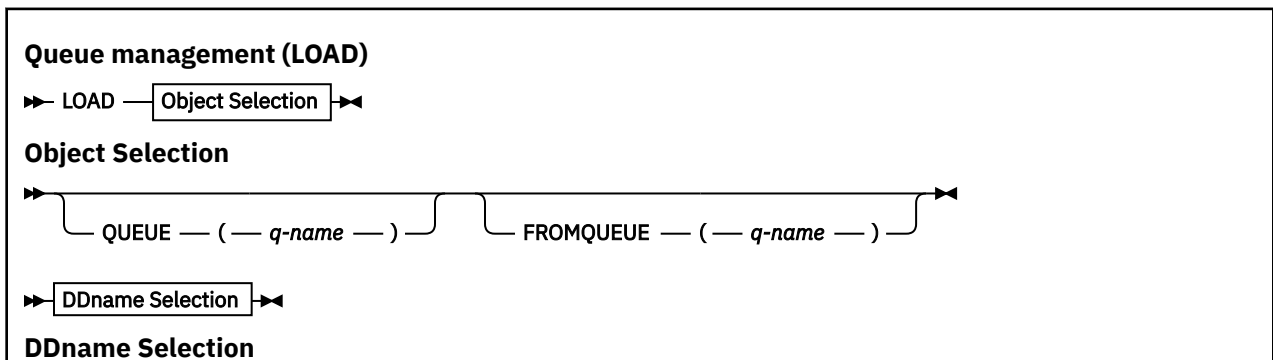
The LOAD function of CSQUTIL is complementary to the COPY or SCOPY function. LOAD restores messages from the destination data set of an earlier COPY or SCOPY operation. The queue manager must be running.

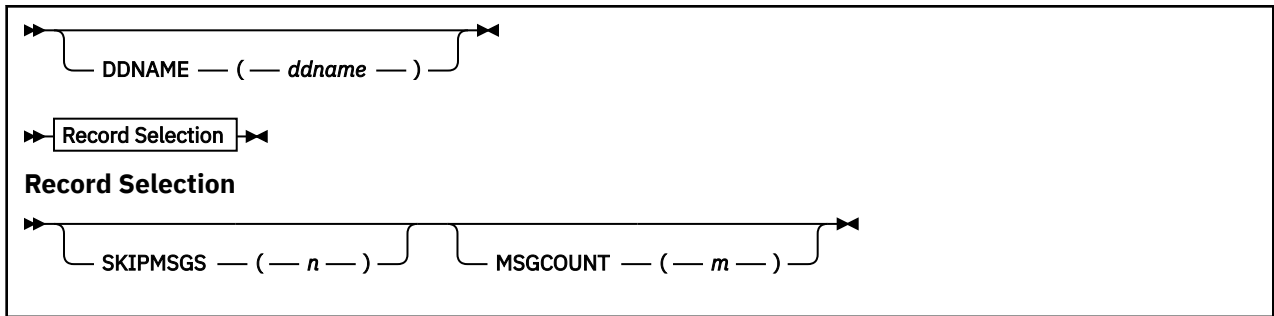
The data set can contain messages from one queue only if it was created by COPY or SCOPY QUEUE, or from a number of queues if it was created by COPY PSID or several successive COPY or SCOPY QUEUE operations. Messages are restored to queues with the same name as those from which they were copied. You can specify that the first or only queue is loaded to a queue with a different name. (This would normally be used with a data set created with a single COPY queue operation to restore the messages to a queue with a different name.)

### Notes:

1. See [“Syncpoints” on page 2741](#) for information about how to avoid problems with duplicate messages if this function fails.
2. An alternative approach to the LOAD function is to use the [“dmpmqmsg \(queue load and unload\)” on page 65](#) utility which is more flexible in many cases.

Messages are restored to queues with the same name as those from which they were copied. You can specify that the first or only queue is loaded to a queue with a different name using the **QUEUE** parameter. (This would normally be used with a data set created with a single COPY queue operation to restore the messages to a queue with a different name.) For a data set containing multiple queues, the first queue to be processed can be specified using the **FROMQUEUE** parameter. Messages are restored to this queue and all subsequent queues in the data set.





- [Keywords and parameters](#)
- [Example](#)
- [Usage notes](#)

## Keywords and parameters

### QUEUE(*q-name*)

This parameter specifies that the messages from the first or only queue on the destination data set of a prior COPY or SCOPY operation are loaded to a named queue. Messages from any subsequent queues are loaded to queues with the same names as those they came from. The keyword QUEUE can be abbreviated to Q.

*q-name* specifies the name of the queue to which the messages are to be loaded. This name is case sensitive. It must not be a model queue.

### FROMQUEUE(*q\_name*)

Specifies the name of the first queue to process on the destination data set of a prior COPY or SCOPY operation. Messages from this queue and any subsequent queues on the data set are loaded to queues with the same names as those that they came from. If this parameter is removed, the LOAD function starts with the first queue on the data set and processes all queues. The keyword FROMQUEUE can be abbreviated to FROMQ.

### DDNAME(*ddname*)

Specifies that messages are loaded from a named data set. This keyword can be abbreviated to DD.

*ddname* specifies the **DDNAME** that identifies the destination data set of a prior COPY or SCOPY operation, from which the messages are to be loaded. This name is not case sensitive, and can be up to 8 characters long.

If you omit **DDNAME** (*ddname*) the default **DDNAME**, CSQUINP, is used.

### SKIPMSGS(*n*)

Specifies that the first *n* messages in the sequential data set are to be skipped before commencing the load of the queue.

If you omit SKIPMSGS(*n*) no messages are skipped; the load starts at the first message.

### MSGCOUNT(*m*)

Specifies that only *m* messages are read from the data set and loaded to the queue.

If you omit MSGCOUNT(*m*) the number of messages read is unlimited.



## Example

```
//LOAD EXEC PGM=CSQUTIL,PARM=('CSQ1'),REGION=0M
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTPUTA DD DSN=MY.UTILITY.OUTPUTA,DISP=SHR
//CSQUINP DD DSN=MY.UTILITY.COPYA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
LOAD QUEUE(ABC123) DDNAME(OUTPUTA)
LOAD QUEUE(TOQ) FROMQUEUE(QUEUEA) SKIPMSGS(55)
/*
```

Figure 28. Sample JCL for the CSQUTIL LOAD function

### Note:

REGION - A value of 0M means that the job is allowed to have the amount of storage it needs. However, if a job tries to acquire too much storage, it might impact other jobs in the system. You must ideally look to limit the REGION size and specify an absolute maximum value that the job is allowed to acquire.

LOAD QUEUE(ABC123) DDNAME(OUTPUTA) - Reloads all queues from the input data set MY.UTILITY.OUTPUTA. The names of the queues loaded are the same as the queue names from which the data was copied, apart from the first queue on the data set which is reloaded to queue ABC123.

LOAD QUEUE(TOQ) FROMQUEUE(QUEUEA) SKIPMSGS(55) - Reloads all queues from the input data set MY.UTILITY.COPYA, starting from queue QUEUEA. The names of the queues loaded are the same as the queue names from which the data was copied, apart from the first queue QUEUEA, which is reloaded to queue TOQ. In processing the messages in QUEUEA, the first 55 messages are ignored, and loading starts from the 56th message.

### Usage notes

1. To use the LOAD function, the queues or page sets involved must not be in use when the function is invoked.
2. If the data set contains multiple queues, the LOAD function is considered successful only if it successfully loads all the queues on the data set. (or all those following the starting queue specified with FROMQUEUE, if this is set).
3. If LOAD fails, or is forced to take a syncpoint, no further CSQUTIL functions are attempted.
4. CSQUTIL uses MQPMO\_SET\_ALL\_CONTEXT to ensure that the message descriptor fields remain the same as the original copy. It therefore needs an access of CONTROL in the CONTEXT profile of the queue. For full details, see [Profiles for context security](#).

## Restoring messages from a data set to a queue (SLOAD) on z/OS

The SLOAD function of CSQUTIL is complementary to the COPY or SCOPY function. SLOAD restores messages from the destination data set of an earlier COPY or SCOPY operation. SLOAD processes a single queue.

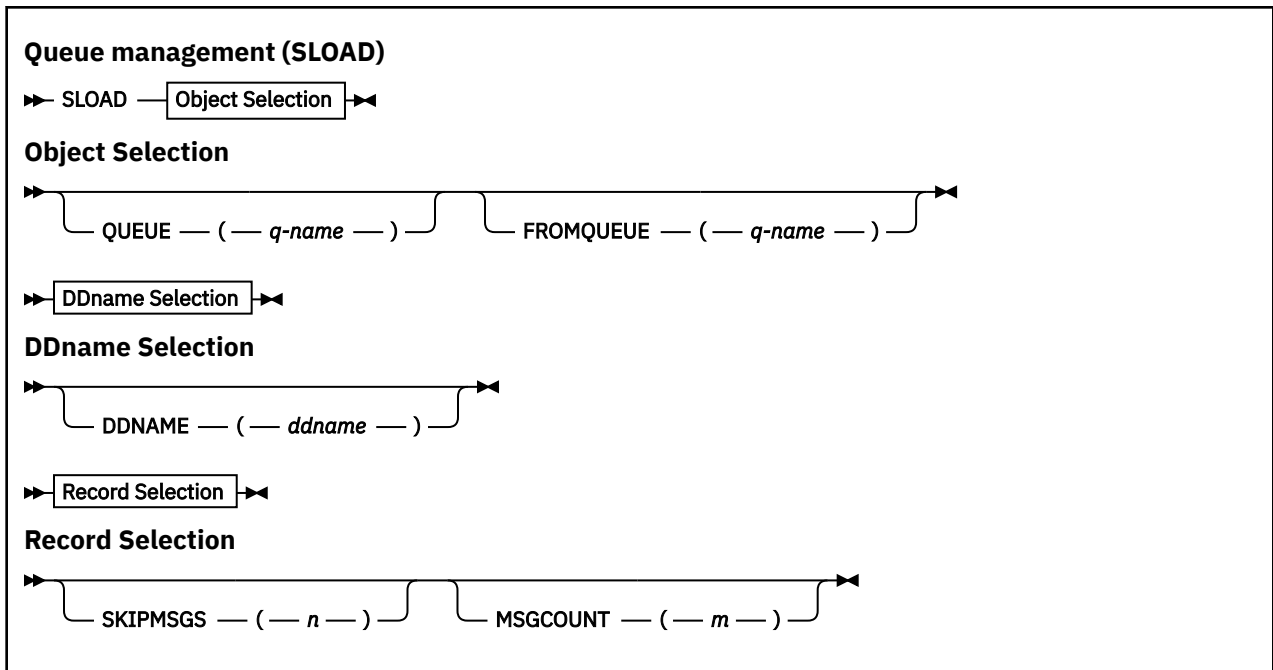
To use SLOAD the queue manager must be running.

If the data set was created by COPY or SCOPY QUEUE it contains messages from one queue only. If the data set was created by COPY PSID or several successive COPY or SCOPY QUEUE operations, it might contain messages from a number of queues.

By default, SLOAD processes the first queue on the data set. You can specify a particular queue to process using the **FROMQUEUE** parameter.

By default, messages are restored to a queue with the same name as the one from which it was copied. You can specify that the queue is loaded to a queue with a different name using the **QUEUE** parameter.

**Note:** See “Syncpoints” on page 2741 for information about how to avoid problems with duplicate messages if this function fails.



- “Keywords and parameters” on page 2770
- “Example” on page 2771
- “Usage notes” on page 2771

## Keywords and parameters

### QUEUE(*q-name*)

This parameter specifies that the messages from the first or only queue on the destination data set of a prior COPY or SCOPY operation are to be loaded to a named queue. The keyword QUEUE can be abbreviated to Q.

*q-name* specifies the name of the queue to which the messages are to be loaded. This name is case sensitive. It must not be a model queue.

### FROMQUEUE(*q-name*)

Specifies the name of the queue to process. If this parameter is omitted, the first queue is processed. The keyword FROMQUEUE can be abbreviated to FROMQ.

*q-name* specifies the name of the queue to be processed. This name is case sensitive.

### DDNAME(*ddname*)

Specifies that messages are to be loaded from a named data set. This keyword can be abbreviated to DD.

*ddname* specifies the **DDNAME** that identifies the destination data set of a prior COPY or SCOPY operation, from which the messages are to be loaded. This name is not case sensitive, and can be up to 8 characters long.

If you omit **DDNAME** (*ddname*) the default **DDNAME**, CSQUINP, is used.

### SKIPMSGS(*n*)

Specifies that the first *n* messages in the sequential data set are to be skipped before commencing the load of the queue.

If you omit SKIPMSGS(*n*) no messages are skipped; the load starts at the first message.



- [Keywords and parameters](#)
- [Example](#)
- [Usage notes](#)

## Keywords and parameters

### CHANNEL (*channel name*)

Specifies the name of a cluster-sender channel, or a generic channel name.

If a generic channel name is specified each cluster-sender channel that matches the generic name is processed.

If a single asterisk is specified all cluster-sender channels are processed.

### MOVEMSGS

Specifies whether messages queued for the channel should be moved from the old transmission queue to the new transmission queue during the switching process. Values are:

#### YES

Messages are moved from the old transmission queue to the new transmission queue. This is the default.

#### NO

Messages are not moved from the old transmission queue to the new transmission queue. If this option is selected it is the responsibility of the system programmer to resolve any messages for the channel on the old transmission queue after the switch has completed.

### STATUS

Display the switching status for matching cluster-sender channels. If this keyword is not specified the command switches the transmission queue for stopped or inactive cluster-sender channels that require switching.

## Examples

Figure 1 illustrates how the SWITCH function can be used to query the switching status of all cluster-sender channels whose names match the generic name CLUSTER.\*.

```
//SWITCH EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SWITCH CHANNEL(CLUSTER.*) STATUS
/*
```

Figure 30. Sample JCL for querying the switching status of cluster-sender channels using the CSQUTIL SWITCH function

Figure 2 illustrates how the SWITCH function can be used to switch the transmission queue for the cluster-sender channel CLUSTER.TO.QM1.

```
//SWITCH EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SWITCH CHANNEL(CLUSTER.TO.QM1)
/*
```

Figure 31. Sample JCL for switching the transmission queue associated with a cluster-sender channel using the CSQUTIL SWITCH function

## Usage notes

1. The channel initiator must be running to initiate a switch of transmission queue for cluster-sender channels.
2. The transmission queue associated with a cluster-sender channel can only be switched if the channel is STOPPED or INACTIVE.
3. You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*)
4. You need the necessary authority to issue the START CHANNEL command.
5. To initiate a switch of transmission queue for a cluster-sender channel, you also need command resource authority for the channel.

## Related tasks

Clustering: [Switching cluster transmission queues](#)

z/OS

## The change log inventory utility (CSQJU003) on z/OS

The IBM MQ change log inventory utility runs as a z/OS batch job to change the bootstrap data set (BSDS).

Through this utility, you can invoke these functions:

### NEWLOG

Add active or archive log data sets.

### DELETE

Delete active or archive log data sets.

### ARCHIVE

Supply passwords for archive logs.

### CRESTART

Control the next restart of IBM MQ.

### CHECKPT

Set checkpoint records.

### HIGHRBA

Update the highest written log RBA.

Only run this utility when IBM MQ is stopped. This is because the active log data sets named in the BSDS are dynamically added for exclusive use to IBM MQ and remain allocated exclusively to IBM MQ until it terminates. You can add new active log data sets to an active queue manager with the [“DEFINE LOG \(define a new active log\) on z/OS”](#) on page 534 command.

The DEFINE LOG command can be used to update a BSDS of any version. However, you must use the CSQJUCNV utility to convert the BSDS from version 1 to version 2. A version 1 BSDS has space for up to 31 active log data sets in each log copy ring, whereas a version 2, or higher, BSDS has space for up to 310 active log data sets in each log copy ring.

## Invoking the CSQJU003 utility on z/OS

Use this topic to understand how to invoke the CSQJU003 utility.

The utility runs as a z/OS batch program. [Figure 32 on page 2774](#) gives an example of the JCL required.

```
//JU003 EXEC PGM=CSQJU003
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=629
//SYSUT1 DD DISP=SHR,DSN=bsds.dsname
//SYSIN DD *
NEWLOG DSNNAME=CSQREPAL.A0001187,COPY1VOL=CSQV04,UNIT=SYSDA,
STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=YES,PASSWORD=PASSWRD
/*
```

Figure 32. Sample JCL to invoke the CSQJU003 utility

### Data definition (DD) statements

CSQJU003 requires DD statements with these DDnames:

#### **SYSUT1**

This statement is required; it names the BSDS.

#### **SYSUT2**

This statement is required if you use dual BSDSs; it names the second copy of the BSDS.

#### **Dual BSDSs and CSQJU003**

Each time you run the CSQJU003 utility, the BSDS time stamp field is updated with the current system time. If you run CSQJU003 separately for each copy of a dual copy BSDS, the time stamp fields are not synchronized, so the queue manager fails at startup, issuing error message CSQJ120E. Therefore, if CSQJU003 is used to update dual copy BSDSs, both BSDSs must be updated within a single run of CSQJU003.

#### **SYSPRINT**

This statement is required; it names a data set for print output. The logical record length (LRECL) is 125. The block size (BLKSIZE) must be 629.

#### **SYSIN**

This statement is required; it names the input data set for statements that specify what the utility is to do. The logical record length (LRECL) is 80.

You can use more than one statement of each type. In each statement, separate the operation name (NEWLOG, DELETE, ARCHIVE, CRESTART) from the first parameter by one or more blanks. You can use parameters in any order; separate them by commas with no blanks. Do not split a parameter description across two SYSIN records.

A statement containing an asterisk (\*) in column 1 is considered to be a comment, and is ignored. However, it appears in the output listing. To include a comment or sequence number in a SYSIN record, separate it from the last comma by a blank. When a blank follows a comma, the rest of the record is ignored.

### Multiple statement operation

When running CSQJU003, a significant error in any statement causes the control statements for the statement in error and all following statements to be skipped. Therefore, BSDS updates cannot occur for any operation specified in the statement in error, or any following statements. However, all the remaining statements are checked for syntax errors.



## Keywords and parameters

### **DSNAME= *dsname***

Names a log data set.

*dsname* can be up to 44 characters long.

### **PASSWORD= *password***

Assigns a password to the data set. It is stored in the BSDS and later used in any access to the active or archive log data sets.

The password is a data set password, and should follow standard VSAM convention: 1 through 8 alphanumeric characters (A through Z, 0 through 9) or special characters (& \* + - . ; ' /).

You should use an ESM such as RACF to provide your data set security requirements.

### **COPY1**

Makes the data set an active log copy-1 data set.

### **COPY2**

Makes the data set an active log copy-2 data set.

### **STARTRBA= *startrba***

Gives the log RBA (relative byte address within the log) of the beginning of the replacement active log data set or the archive log data set volume specified by DSNAME.

*startrba* is a hexadecimal number of up to 16 characters. The value must end with 000. If you use fewer than 16 characters, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

The value of STARTRBA must be a multiple of 4096. (The hexadecimal value must end in 000.)

A value higher than FFFFFFFF000 cannot be specified for a version 1 format BSDS.

### **ENDRBA= *endrba***

Gives the log RBA (relative byte address within the log) of the end of the replacement active log data set or the archive log data set volume specified by DSNAME.

*endrba* is a hexadecimal number of up to 16 characters. The value must end with FFF. If you use fewer than 16 characters, leading zeros are added.

A value higher than FFFFFFFFFF cannot be specified for a version 1 format BSDS.

### **STARTIME= *starttime***

Start time of the RBA in the BSDS. This is an optional field. The time stamp format (with valid values in parentheses) is yyyydddhhmssst, where:

#### **yyyy**

Indicates the year (1993 through 2099)

#### **ddd**

Indicates the day of the year (1 through 365; 366 in leap years)

#### **hh**

Indicates the hour (zero through 23)

#### **mm**

Indicates the minutes (zero through 59)

#### **ss**

Indicates the seconds (zero through 59)

#### **t**

Indicates tenths of a second

If fewer than 14 digits are specified for the STARTIME and ENDTIME parameter, trailing zeros are added.

STARTRBA is required when STARTIME is specified.



**ENDTIME= *endtime***

End time of the RBA in the BSDS. This is an optional field. For time stamp format, see the STARTIME option. The ENDTIME value must be greater than or equal to the value of STARTIME.

**STRTLRSN= *strtlrsn***

Gives the LRSN (logical record sequence number) of the first complete log record on the new archive data set.

*strtlrsn* is a hexadecimal number of up to 12 characters. If you use fewer than 12 characters, leading zeros are added.

**ENDLRSN= *endlrsn***

Gives the LRSN (logical record sequence number) of the last log record on the new archive data set.

*endlrsn* is a hexadecimal number of up to 12 characters. If you use fewer than 12 characters, leading zeros are added.

**COPY1VOL= *vol-id***

The volume serial of the copy-1 archive log data set named after DSNAME.

**COPY2VOL= *vol-id***

The volume serial of the copy-2 archive log data set named after DSNAME.

**UNIT= *unit-id***

The device type of the archive log data set named after DSNAME.

**CATALOG**

Specifies whether the archive log data set is cataloged:

**NO**

The archive log data set is not cataloged. All subsequent allocations of the data set are made using the unit and volume information specified on the function. This is the default.

**YES**

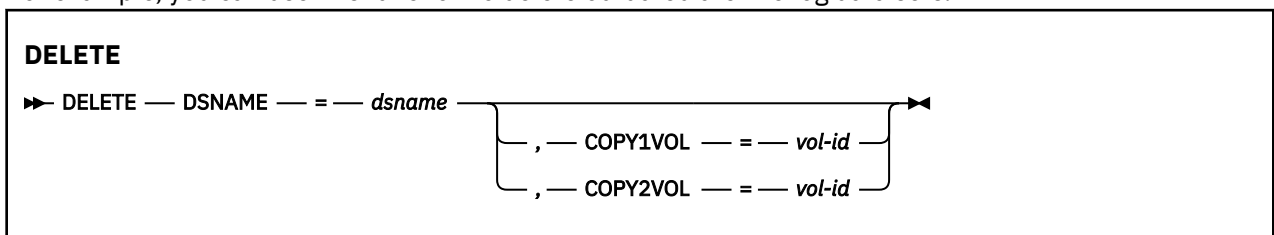
The archive log data set is cataloged. A flag is set in the BSDS indicating this, and all subsequent allocations of the data set are made using the catalog.

IBM MQ requires that all archive log data sets on DASD be cataloged. Select CATALOG=YES if the archive log data set is on DASD.

## **Deleting information about a data set from the BSDS (DELETE) on z/OS**

You can use the DELETE function of CSQJU003 to delete all information about a specified log data set or data set volume from the bootstrap data sets.

For example, you can use this function to delete outdated archive log data sets.

**Keywords and parameters****DSNAME= *dsname***

Specifies the name of the log data set.

*dsname* can be up to 44 characters long.

**COPY1VOL= *vol-id***

The volume serial number of the copy-1 archive log data set named after DSNAME.

## **COPY2VOL= vol-id**

The volume serial number of the copy-2 archive log data set named after DSNAME.

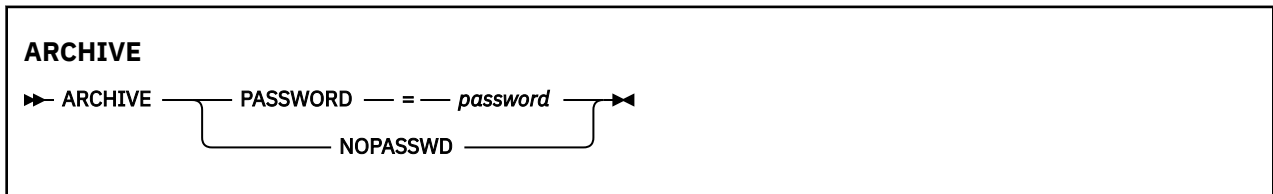
## **z/OS** Supplying a password for archive log data sets (ARCHIVE) on z/OS

You can use the ARCHIVE function of CSQJU003 to assign a password to all archive data sets created after this operation.

This password is added to the z/OS password data set each time a new archive log data set is created.

Use the NOPASSWD keyword to remove the password protection for all archives created after the archive operation.

**Note:** Typically, use an external security manager (ESM), such as RACF, if you want to implement security on any IBM MQ data sets.



## **Keywords and parameters**

### **PASSWORD= password**

Specifies that a password is to be assigned to the archive log data sets.

*password* specifies the password, which is a data set password and it must follow the standard VSAM convention; that is, 1 through 8 alphanumeric characters (A through Z, 0 through 9) or special characters (& \* + - . ; ' /).

### **NOPASSWD**

Specifies that archive password protection is not to be active for all archives created after this operation. No other keyword can be used with NOPASSWD.

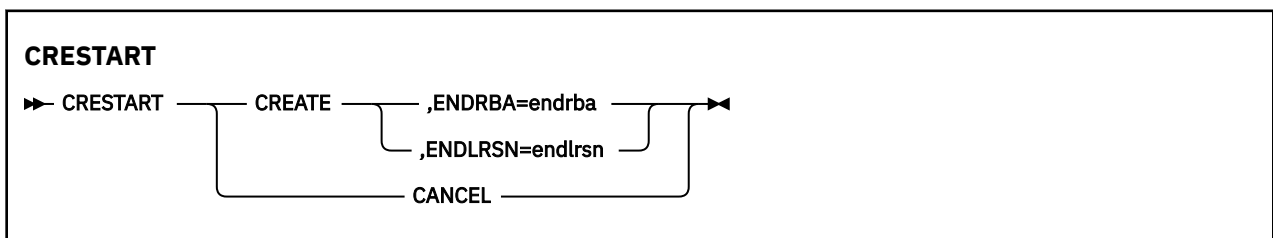
## **z/OS** Controlling the next restart (CRESTART) on z/OS

You can use the CRESTART function of CSQJU003 to control the next restart of the queue manager, either by creating a new conditional restart control record or by canceling the one currently active.

These records limit the scope of the log data used during restart (truncating the log, in effect) . Any existing conditional restart control record governs every restart until one of these events occurs:

- A restart operation completes
- A CRESTART CANCEL is issued
- A new conditional restart control record is created

**Attention: This can override IBM MQ efforts to maintain data in a consistent state.** Only use this function when implementing the disaster recovery process described in [Recovering a single queue manager at an alternative site](#) and [Recovering a queue sharing group at the alternative site](#), or under the guidance of IBM service.



## Keywords and parameters

### CREATE

Creates a new conditional restart control record. When the new record is created, the previous control record becomes inactive.

### CANCEL

Makes the currently active conditional restart control record inactive. The record remains in the BSDS as historical information.

No other keyword can be used with CANCEL.

### ENDRBA= *endrba*

Gives the last RBA of the log to be used during restart (the point at which the log is to be truncated), and the starting RBA of the next active log to be written after restart. Any log information in the bootstrap data set and the active logs, with an RBA greater than *endrba*, is discarded.

*endrba* is a hexadecimal number of up to 16 digits. If you use fewer than 16 digits, leading zeros are added.

The value of ENDRBA must be a multiple of 4096. (The hexadecimal value must end in 000.)

A value higher than FFFFFFFF000 cannot be specified for a version 1 format BSDS.

### ENDLRSN= *endlrsn*

Gives the LRSN of the last log record to be used during restart (the point at which the log is to be truncated). Any log information in the bootstrap data set and the active logs with an LRSN greater than *endlrsn* is discarded.

## Setting checkpoint records (CHECKPT) on z/OS

You can use the CHECKPT function of CSQJU003 to add or delete a record in the BSDS checkpoint queue.

Use the STARTRBA and ENDRBA keywords to add a record, or the STARTRBA and CANCEL keywords to delete a record.

**Attention: This can override IBM MQ efforts to maintain data in a consistent state.** Only use this function when implementing the disaster recovery process described in [Recovering a single queue manager at an alternative site](#) and [Recovering a queue sharing group at the alternative site](#), or under the guidance of IBM service.

### CHECKPT

```
▶▶ CHECKPT — STARTRBA — = — startrba →  
  
    , — ENDRBA — = — offlrba — , — TIME — = — time ▶▶  
    , — CANCEL —
```

## Keywords and parameters

### STARTRBA= *startrba*

Indicates the start checkpoint log record.

*startrba* is a hexadecimal number of up to 16 digits. If you use fewer than 16 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

A value higher than FFFFFFFFFF cannot be specified for a version 1 format BSDS.

### ENDRBA= *endrba*

Indicates the end checkpoint log record corresponding to the start checkpoint record.

*endrba* is a hexadecimal number of up to 16 digits. If you use fewer than 16 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

A value higher than FFFFFFFFFF cannot be specified for a version 1 format BSDS.

**TIME= *time***

Gives the time the start checkpoint record was written. The time stamp format (with valid values in parentheses) is *yyydddhhmmsst*, where:

**yyyy**

Indicates the year (1993 through 2099)

**ddd**

Indicates the day of the year (1 through 365; 366 in leap years)

**hh**

Indicates the hour (zero through 23)

**mm**

Indicates the minutes (zero through 59)

**ss**

Indicates the seconds (zero through 59)

**t**

Indicates tenths of a second

If fewer than 14 digits are specified for the TIME parameter, trailing zeros are added.

**CANCEL**

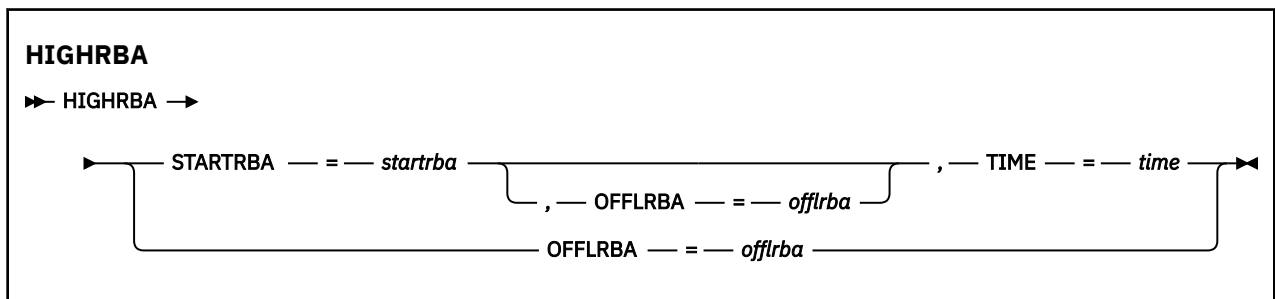
Deletes the checkpoint queue record containing a starting RBA that matches the RBA specified by STARTRBA.

**z/OS Updating the highest written log RBA (HIGHRBA) on z/OS**

You can use the HIGHRBA function of CSQJU003 to update the highest written log RBA recorded in the BSDS for either the active or archive log data sets.

Use the STARTRBA keyword to update the active log, and the OFFLRBA keyword to update the archive log.

**Attention: This can override IBM MQ efforts to maintain data in a consistent state.** Only use this function when implementing the disaster recovery process described in [Recovering a single queue manager at an alternative site](#), or under the guidance of IBM service personnel.



**Keywords and parameters**

**STARTRBA= *startrba***

Indicates the log RBA of the highest written log record in the active log data set.

*startrba* is a hexadecimal number of up to 16 digits. If you use fewer than 16 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

A value higher than FFFFFFFFFF cannot be specified for a version 1 format BSDS.

**TIME= *time***

Specifies when the log record with the highest RBA was written to the log. The time stamp format (with valid values in parentheses) is *yyydddhhmmsst*, where:

**yyyy**

Indicates the year (1993 through 2099)

**ddd**

Indicates the day of the year (1 through 365; 366 in leap years)

**hh**

Indicates the hour (zero through 23)

**mm**

Indicates the minutes (zero through 59)

**ss**

Indicates the seconds (zero through 59)

**t**

Indicates tenths of a second

If fewer than 14 digits are specified for the TIME parameter, trailing zeros are added.

**OFFLRBA= *offlrba***

Specifies the highest offloaded RBA in the archive log.

*offlrba* is a hexadecimal number of up to 16 digits. If you use fewer than 16 digits, leading zeros are added. The value must end with hexadecimal 'FFF'.

A value higher than FFFFFFFFFFFF cannot be specified for a version 1 format BSDS.

**z/OS****The print log map utility (CSQJU004) on z/OS**

CSQJU004 is the batch utility program used to print log data information from the BSDS.

The IBM MQ print log map utility runs as a z/OS batch program to list the following information:

- The BSDS version
- Log data set name and log RBA association for both copies of all active and archive log data sets
- Active log data sets available for new log data
- Contents of the queue of checkpoint records in the bootstrap data set (BSDS)
- Contents of the quiesce history record
- System and utility time stamps
- Passwords for the active and archive log data sets, if provided

You can run the CSQJU004 program regardless of whether the queue manager is running. However, if the queue manager is running, consistent results from the utility can be ensured only if both the utility and the queue manager are running under control of the same z/OS system.

For further information, see

- [Invoking the CSQJU004 utility](#)
- [Data definition statements required for the CSQJU004 utility](#)

To use this utility, the user ID of the job must have the requisite security authorization, or, if the BSDS is password protected, the appropriate VSAM password for the data set.

**Invoking the CSQJU004 utility**

The following example shows the JCL used to invoke the CSQJU004 utility:

```
//JU004 EXEC PGM=CSQJU004
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=bsds.dsname
```

Figure 33. Sample JCL to invoke the CSQJU004 utility

The EXEC statement can use an optional parameter TIME(RAW) which changes the way timestamps are formatted.

```
//JU004 EXEC PGM=CSQJU004,PARM='TIME(RAW)'
```

This parameter causes timestamps to be formatted without applying timezone or leap second offsets for the formatting system. You can use this mode of operation when formatting a BSDS created at a remote site, or before a daylight saving time change, for example. The default, no parameter specified, is to format timestamps using the current formatting system's timezone and leap second corrections.

Formatted times affected by this parameter are:

- highest RBA written
- archive log command times
- checkpoint times
- conditional restart record times

#### Data definition statements

The CSQJU004 utility requires DD statements with the following DDnames:

##### **SYSUT1**

This statement is required to specify and allocate the bootstrap data set. If the BSDS must be shared with a concurrently running queue manager subsystem, use DISP=SHR on the DD statement.

##### **SYSPRINT**

This statement is required to specify a data set or print spool class for print output. The logical record length (LRECL) is 125 and the record format (RECFM) is VBA.

[Finding out what the BSDS contains](#) describes the output.

z/OS

## The log print utility (CSQ1LOGP) on z/OS

Use this utility to print information contained in the IBM MQ log data sets or the BSDS.

- [Invoking the CSQ1LOGP utility](#)
- [Input control parameters](#)
- [Usage notes](#)
- [The EXTRACT function](#)
  - [Example of processing EXTRACT data](#)
- [CSQ1LOGP output](#)
  - [Detail report](#)
  - [Record layouts for the output data sets](#)

### Invoking the CSQ1LOGP utility

You run the IBM MQ log print utility as a z/OS batch program. You can specify:

- A bootstrap data set (BSDS)
- Active log data sets (with no BSDS)
- Archive log data sets (with no BSDS)

Sample JCL to invoke the CSQ1LOGP utility is shown in [Figure 34 on page 2784](#), [Figure 35 on page 2784](#), [Figure 36 on page 2784](#) and [Figure 37 on page 2784](#).

These data definition statements must be provided:

#### **SYSPRINT**

All error messages, exception conditions, and the detail report are written to this data set. The logical record length (LRECL) is 131.

#### **SYSIN**

Input selection criteria can be specified in this data set. See [“Input control parameters” on page 2785](#) for more information.

The logical record length (LRECL) must be 80, but only columns 1 through 72 are significant; columns 73 through 80 are ignored. At most 50 records can be used. Records with an asterisk (\*) in column 1 are interpreted as comments and are ignored.

#### **SYSSUMRY**

If a summary report is requested, by specifying the parameter **SUMMARY ( YES )** or **SUMMARY ( ONLY )**, the output is written to this data set. The logical record length (LRECL) is 131.

#### **BSDS**

Name of the bootstrap data set (BSDS).

#### **ACTIVE<sub>n</sub>**

Name of an active log data set you want to print (n=number).

#### **ARCHIVE**

Name of an archive log data set you want to print.

If you specify the keyword **EXTRACT ( YES )**, provide one or more of the following DD statements, depending on what types of data you want to extract. Do not specify an LRECL, as it is set internally by the utility. These DDs are the required DCB parameters for the output data set.

#### **CSQBACK**

This data set contains persistent messages written to the log by units of work that were rolled back during the log range specified.

#### **CSQCMT**

This data set contains persistent messages written to the log by units of work that were committed during the log range specified.

#### **CSQBOTH**

This data set contains persistent messages written to the log by units of work that were either committed or rolled back during the log range specified.

#### **CSQINFLT**

This data set contains persistent messages written to the log by units of work that remained in flight during the log range specified.

#### **CSQOBS**

This data set contains information about object alterations that occurred during the log range specified.

For each DD statement, the record format (RECFM) is VB, the logical record length (LRECL) is 32756, and the block size (BLKSIZE) must be 32760.

If you are processing active log data sets, the utility runs even if IBM MQ is running, if the BSDS and active log data sets are defined by using at least SHAREOPTIONS(2 3).

```

V 9.3.4
//PRTLOG EXEC PGM=CSQ1LOGP
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQLOAD
//BSDS DD DSN=qmgr.bsds.dsname,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN DD *
* select records for page set 3. Produce both summary and detail reports
PAGESET(3)
SUMMARY(YES)
/*

```

Figure 34. Sample JCL to invoke the CSQ1LOGP utility using a BSDS

```

//PRTLOG EXEC PGM=CSQ1LOGP
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQLOAD
//ACTIVE1 DD DSN=qmgr.logcopy1.ds01,DISP=SHR
//ACTIVE2 DD DSN=qmgr.logcopy1.ds02,DISP=SHR
//ACTIVE3 DD DSN=qmgr.logcopy1.ds03,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN DD *
insert your input control statements here, for example:
Urid(urid1)
Urid(urid2)
/*

```

Figure 35. Sample JCL to invoke the CSQ1LOGP utility using active log data sets

```

//PRTLOG EXEC PGM=CSQ1LOGP
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQLOAD
//ARCHIVE DD DSN=qmgr.archive1.ds01,DISP=SHR
// DD DSN=qmgr.archive1.ds02,DISP=SHR
// DD DSN=qmgr.archive1.ds03,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN DD *
insert your input control statements here
/*

```

Figure 36. Sample JCL to invoke the CSQ1LOGP utility using archive log data sets

```

V 9.3.4
//PRTLOG EXEC PGM=CSQ1LOGP
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQLOAD
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
...
//CSQBACK DD DSN=backout.dataset,DISP=(NEW,CATLG)
//CSQCMT DD DSN=commit.dataset,DISP=(NEW,CATLG)
//CSQBOTH DD DSN=both.dataset,DISP=(NEW,CATLG)
//CSQINFLT DD DSN=inflight.dataset,DISP=(NEW,CATLG)
//CSQOBS DD DSN=objects.dataset,DISP=(NEW,CATLG)

```

Figure 37. Sample JCL showing additional statements for the EXTRACT keyword

**Note:** **V 9.3.4** thlqual.SCSQAUTH DD is required in STEPLIB for EXTRACT.



```

//STEP1 EXEC PGM=CSQ1LOGP,REGION=0M
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
//          DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//          DD DISP=SHR,DSN=thlqua1.SCSQLOAD
//ARCHIVE DD DISP=SHR,DSN=xxx.yyy.A0030620
//          DD DISP=SHR,DSN=xxx.yyy.A0030621
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//CSQCMT DD DSN=xxx.MSGS.COMMIT,
// DISP=(NEW,CATLG),SPACE=(CYL,(1,10),RLSE),UNIT=SYSDA
//SYSIN DD *
EXTRACT(YES) SUMMARY(NO)
URID(xxxxxxxxxxxxxx)
/*
//STEP2 EXEC PGM=CSQ4LOGS,PARM=('MQST REPLAY'),REGION=0M
//STEPLIB DD DSN=thlqua1.SCSQANLE,DISP=SHR
//          DD DSN=thlqua1.SCSQAUTH,DISP=SHR
//          DD DSN=thlqua1.SCSQLOAD,DISP=SHR
//FILEIN DD DSN=xxx.MSGS.COMMIT,DISP=SHR
//SYSDBOU DD SYSOUT=*
//SYSABOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
/*

```

Figure 38. Sample JCL to retrieve recently deleted messages from the archive log

Note that **xxx.MSGS.COMMIT** is consistent in both steps.

The EXEC statement can use an optional parameter **TIME(RAW)** which changes the way timestamps are formatted.

```
//PRTLOG EXEC PGM=CSQ1LOGP,PARM='TIME(RAW)'
```

This causes timestamps to be formatted without applying timezone or leap second offsets for the formatting system. You can use this mode of operation when formatting log data created at a remote site, or before a daylight saving time change, for example.

If no parameter is specified, the default behavior is to format timestamps using the timezone and leap second corrections of the system doing the formatting.

Formatted times affected by this parameter are those associated with:

- checkpoint time
- restart time
- UR start time

## Input control parameters

The keywords that you can use in the SYSIN data set are described in the following list.

You can specify various selection criteria to limit the log records that are processed. These are:

- log range, using RBASTART-RBAEND or LRSNSTART-LRSNEND
- page sets, using PAGESET
- units of recovery, using URID
- record contents, using DATA
- resource manager, using RM

Different types of selection criteria can be combined; only records meeting all the criteria are processed.

**LRSNSTART (hexadecimal-constant)**

Specifies the logical record sequence number (LRSN) from which to begin processing. You cannot use this keyword together with RBASTART. Use this keyword only if your queue manager is in a queue sharing group.

LRSN values are always greater than A0000000000; this value is used as the start value if a lower value is specified.

You can also use the forms STARTLRSN or STRTLRSN or LRSNSTRT. Specify this keyword only once.

**LRSNEND (hexadecimal-constant)**

Specifies the logical record sequence number (LRSN) of the last record to be scanned. The default is FFFFFFFFFF (the end of the data sets). You can use this keyword only with LRSNSTART.

You can also use the form ENDLRSN.

Specify this keyword only once.

**RBASTART (hexadecimal-constant)**

Specifies the log RBA from which to begin processing. You cannot use this keyword together with LRSNSTART.

You can also use the forms STARTRBA or ST. Specify this keyword only once.

**RBAEND (hexadecimal-constant)**

Specifies the last valid log RBA that is to be processed. If this keyword is omitted, processing continues to the end of the log (FFFFFFFFFFFF if 6 byte RBAs are in use, or FFFFFFFFFFFFFFFF if 8 byte RBAs are in use). You can use this keyword only with RBASTART.

You can also use the forms ENDRBA or EN. Specify this keyword only once.

**PAGESET (decimal-integer)**

Specifies a page set identifier. The number must be in the range 00 through 99. You can specify a maximum of 10 PAGESET keywords. If PAGESET keywords are specified, only log records associated with the page sets you specify are processed.

**URID (hexadecimal-constant)**

Specifies a hexadecimal unit of recovery identifier. Changes to data occur in the context of an IBM MQ unit of recovery. A unit of recovery is identified on the log by a BEGIN UR record. The log RBA of that BEGIN UR record is the URID value you must use. If you know the URID for a particular UR that you are interested in, you can limit the extraction of information from the log to that URID.

The hexadecimal constant can consist of 1 through 16 characters (8 bytes), and leading zeros are not required.

You can specify a maximum of 10 URID keywords.

**DATA (hexadecimal-string)**

Specifies a data string in hexadecimal.

The string can consist of 2 through 48 characters (24 bytes), and must have an even number of characters.

You can specify a maximum of 10 DATA keywords.

If multiple DATA keywords are specified, only log records that contain at least one of the strings are processed.

**Note:** Though you can use the DATA and EXTRACT parameters together, it is difficult to reliably derive meaning from the output, unless you have a good understanding of the internal implementation of IBM MQ. This is because only the low level individual log records that contain the requested DATA are processed so you do not extract the full output that is logically associated with the data, only the records where that DATA sequence actually appears. For example you might get only records associated with putting messages and not with getting messages, or you might get only the first part of the data for long messages because the rest of the data is in other log records that do not contain the requested DATA string.

**RM (resource\_manager)**

Specifies a particular resource manager. Only records associated with this resource manager are processed. Valid values for this keyword are:

**RECOVERY**

Recovery log manager

**DATA**

Data manager

**BUFFER**

Buffer manager

**IMSBRIDGE**

IMS bridge

**SUMMARY (YES|NO|ONLY)**

Specifies whether a summary report is to be produced or not:

**YES**

Produce a summary report in addition to the detail report.

**NO**

Do not produce a summary report.

**ONLY**

Produce only a summary report (no detail report).

The default is NO.

**EXTRACT (YES|NO)**

Specifying EXTRACT(YES) causes each log record that meets the input selection criteria to be written to the appropriate output file, as explained on page [“The EXTRACT function” on page 2788](#). The default is NO.

**Note:** Though you can use the DATA and EXTRACT parameters together, it is difficult to reliably derive meaning from the output, unless you have a good understanding of the internal implementation of IBM MQ. This is because only the low level individual log records that contain the requested DATA are processed so you do not extract the full output that is logically associated with the data, only the records where that DATA sequence actually appears. For example you might get only records associated with putting messages and not with getting messages, or you might get only the first part of the data for long messages because the rest of the data is in other log records that do not contain the requested DATA string.

**V 9.3.4 QMCCSID (decimal-integer)**

Specifies the Coded Character Set Identifier to use as the queue manager CCSID. This should match the CCSID of the queue manager to which the logs belong.

It is ignored unless EXTRACT(YES) is also used, and is used with EXTRACT(YES) to process messages with message properties. The default is 500.

**V 9.3.4 EXV6COMPAT (YES|NO)**

Specifying EXV6COMPAT(YES) causes extracted messages that were put under PROPCTL(V6COMPAT) and that have had additional message properties added to be written to the appropriate output file with csvevb of MQPUTV6C. See [“The EXTRACT function” on page 2788](#) for more information.

It is ignored unless EXTRACT(YES) is also used. The default is NO.

**DECOMPRESS (YES|NO)**

Specifies whether any compressed log records will be expanded:

**YES**

Any compressed log records will be expanded before a Search, Print or Extract function is performed.

**NO**

Any compressed log records will not be expanded before a Search or Print function is performed. Do not use DECOMPRESS(NO) with the Extract function.

The default is YES.

## Usage notes

1. If your queue manager is in a queue sharing group, you can specify the log range required by either LRSNSTART (optionally with LRSNEND) or RBASTART (optionally with RBAEND). You cannot mix LRSN and RBA specifications.  
  
If you need to coordinate the log information from the different queue managers in the queue sharing group, use LRSN specifications. Note that processing logs simultaneously from different queue managers in a queue sharing group is not supported.
2. If your queue manager is not in a queue sharing group, you cannot use LRSN specifications; you must use RBA specifications.
3. If you are using a BSDS, RBASTART or LRSNSTART must be specified.
4. CSQ1LOGP starts its processing on the first record containing an LRSN or RBA value greater than or equal to the value specified on LRSNSTART or RBASTART.
5. Normally you are only interested in the most recent additions to the log. Take care to choose a suitable value for the start of the log range, and do not use the defaults. Otherwise, you create an enormous amount of data, most of which is of no interest to you.

## The EXTRACT function

Typical uses of the EXTRACT parameter are to:

- Review which persistent messages were put to or got from a queue and whether the request was committed. This allows messages to be replayed.
- Review persistent messages that were put or got, but the request was backed out.
- Display which applications backed out rather than committed.
- Discover the volume of persistent data processed by queues, to identify the high use queues.
- Identify which applications set object attributes.
- Re-create object definitions for recovery purposes after a major failure, for private queues only.

When CSQ1LOGP with the EXTRACT parameter set is run against a log data set it processes all records in the data set, or all those within a specified range. Processing is as follows:

1. When a commit request is found, if the CSQCMT ddname is present then the data is written to this data set. If the CSQBOTH ddname is present the data is also written to this data set.
2. When a backout request is found, if the CSQBACK ddname is present then the data is written to this data set. If the CSQBOTH ddname is present the data is also written to this data set.
3. When changes to objects are detected, the information is written to the data set identified by the CSQOBS ddname.
4. When the last record has been processed, information about remaining units of work is written to the data set identified by the CSQINFLT ddname.

If you do not want to collect one or more of these classes of information, then omit the appropriate DD statements.

### Example of processing EXTRACT data

The following job uses DFSORT facilities to process the file of committed records to add up the number of bytes put to each queue.

```

//TOOLRUN EXEC PGM=ICETOOL,REGION=1024K
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//TOOLIN DD *
SORT FROM(IN) TO(TEMP1) USING(CTL1)
DISPLAY FROM(TEMP1) LIST(OUT1) ON(5,48,CH) ON(53,4,BI)
/*
//CTL1 DD *
* SELECT THE RECORDS WHICH WERE PUT
  INCLUDE COND=(180,5,CH,EQ,C'MQPUT')
* SORT BY QUEUE NAME
  SORT FIELDS=(112,48,CH,A)
* ONLY COPY THE QUEUE NAME AND SIZE OF USER DATA TO OUTPUT REC
  OUTREC FIELDS=(1,4,112,48,104,4)
* ADD UP THE NUMBER OF BYTES PROCESSED
* SUM FIELDS=(104,4,FI)
/*
//IN DD DISP=SHR,DSN=commit.dataset
//TEMP1 DD DISP=(NEW,DELETE),DSN=&TEMP1,SPACE=(CYL,(10,10))
//OUT1 DD SYSOUT=*

```

Figure 39. Accumulating bytes put to each queue

This produces output in the following format:

BA1	3605616
BA10	3572328
BA2	3612624
BA3	3579336
BA4	3572328
BA5	3491736
BA6	3574080
BA7	3532032
BA8	3577584
BA9	3539040
SYSTEM.ADMIN.CHANNEL.EVENT	186120
SYSTEM.ADMIN.QMGR.EVENT	384
SYSTEM.CHANNEL.SYNCQ	46488312

The following table lists the samples that are provided to allow you to print and interpret the data generated when EXTRACT(YES) is used

Table 384. Samples for printing and interpreting EXTRACT data	
Sample	Description
thlqual.SCSQLOAD(CSQ4LOGS)	Sample C program to: <ul style="list-style-type: none"> <li>Report on the unit of work activity, and on activity that defines (that is, creates) or alters an object.</li> <li>Optionally, replay messages.</li> </ul>
thlqual.SCSQC37S(CSQ4LOGS)	Source for sample C program
thlqual.SCSQC370(CSQ4LOGD)	C header file to map records generated when using the EXTRACT(YES) function of CSQ1LOGP

Table 384. Samples for printing and interpreting EXTRACT data (continued)

Sample	Description
thlqual.SCSQPROC(CSQ4LOGJ)	Sample JCL to run program CSQ4LOGS

**Note:** You should not run the CSQ4LOGS program from an APF authorized library. In some circumstances you receive an abend code if you do so.

## CSQ1LOGP output

### Detail report

The detail report begins by echoing the input selection criteria specified by SYSIN, and then prints each valid log record encountered. Definitions of keywords in the detail report are as follows:

**RM**

Resource manager that wrote the log record.

**TYPE**

Type of log record.

**URID**

BEGIN UR for this unit of recovery, see the previous description.

**LRID**

Logical record identifier in the form: AAAAAAAA .BBBBBBCC where:

**AAAAAAA**

Is the page set number.

**BBBBBB**

Is the relative page number in the page set.

**CC**

Is the relative record number on the page.

**LRSN**

Logical record sequence number (LRSN) of the log record scanned.

**SUBTYPE**

Subtype of the log record type.

**CHANGE LENGTH**

Length of the logged change.

**CHANGE OFFSET**

Start position of the change.

**BACKWARD CHAIN**

Pointer to the previous page.

**FORWARD CHAIN**

Pointer to the next page.

**RECORD LENGTH**

Length of the inserted record.

### Record layouts for the output data sets

The data sets produced when the EXTRACT keyword is specified contains information about persistent messages. Messages are identified by their queue name and an eight character key. Once a message has been got, the key can be reused by another message, so it is important to ensure that time sequence is maintained. In the records are times. A time stamp can be extracted only from a Begin-UR record or from an MQPUT request. Thus if there is only a long running transaction which is getting messages, the times when the gets occurred are the time the transaction started (the Begin-UR record). If there are many short units of work, or many messages being put, the time is reasonably accurate (within milliseconds). Otherwise the times become less and less accurate.

**Note:** There is a 4 byte Record Descriptor Word at the front of each record because the files are Variable Blocked format. The first data byte of a variable-length record has relative position 5 and the first 4 bytes contain the record descriptor word. The field names correspond to those in the C header file CSQ4LOGD in thlqual.SCSQC370.

The information in the data sets has the following layout:

<i>Table 385. Record layout for the output data set</i>					
<b>Offset (Dec)</b>	<b>Offset (Hex)</b>	<b>Type</b>	<b>Length</b>	<b>Name</b>	<b>Description</b>
0	0	Character	21	csrecord date	The approximate time the log was written, in the format yyyy.ddd hh:mm:ss.thm
21	15	Character	7	cstimedelta	Approximate time difference in milliseconds from the start of the unit of work. Right-aligned and padded with blanks.
28	1C	64-bit integer	8	dtodout	Estimated time that the log record was created, in STCK format.
36	24	Character	8	csurid	Queue manager-specific unique identifier of the unit of work that created the log record.
44	2C	Character	12	cscorrelator	Thread correlation identifier
56	38	Character	8	csauth	Authorization identifier (Userid associated with unit of work)
64	40	64-bit integer	8	dtime	Time that the unit of work was started, in STCK format
72	48	Character	8	csresource	Resource name
80	50	Character	8	cscnty	Connection type: one of BATCH, RRSBATCH, IMS, CICS, CHIN, or nulls for an internal task
88	58	Character	8	cscnid	Connection ID of thread that created this unit of work
96	60	Character	3	csstatus	Unit of work type: BUR for begin or CP for checkpoint information
99	63	Integer	4	ldatalen	Length of the message data (if any)
103	67	Character	4	csqmgrname	Name of queue manager
107	6B	Character	48	csqueue name	Name of queue, for get, put, or expired messages. This field can be question marks. Question marks appear when it is not possible to determine the user ID associated with the entry. This typically happens when the begin_ur record or the checkpoint record from which you might get the URID is not in the log range specified in the job, nor on the log data sets used.
155	9B	Character	12	cssqdmcp	Shared queue message key. Blank if not a shared queue
167	A7	Character	8	cscdmcp	Non-shared queue message key. Blank if a shared queue.

Table 385. Record layout for the output data set (continued)

Offset (Dec)	Offset (Hex)	Type	Length	Name	Description
175	AF	Character	8	csverb	<p>Activity:</p> <p><b>ALTER</b> the object was changed</p> <p><b>DEFINE</b> the object was created</p> <p><b>MQGET</b> the message was got</p> <p><b>MQPUT</b> the message was put</p> <p><b>V9.3.4 MQPUTPRP</b> the message was put with properties. See Note “1” on page 2793.</p> <p><b>V9.3.4 MQPUTV6C</b> the message was put with PROPCTL(V6COMPAT) and additional properties were added, and the EXV6COMPAT(YES) option was specified. See Note “2” on page 2793.</p> <p><b>EXPIRE</b> the message expired</p> <p><b>ABORT2</b> the message was backed out</p> <p><b>PHASE1</b> the first phase of two-phase commit</p> <p><b>PHASE2</b> the second phase of two-phase commit, or the only phase of one phase commit</p>
183	B7	Character	1	cscmitstatus	<p>Status of unit of work:</p> <p><b>B</b> backed out</p> <p><b>C</b> committed</p> <p><b>I</b> inflight</p>
184	B8	Character	1	csshunt	<p>Shunted indicator:</p> <p><b>S</b> shunted record</p> <p><b>N</b> not shunted</p>
185	B9	Character	8	cslogrba	RBA of log record
193	C1	Character	8	csshuntrba	RBA of shunted log record



Table 385. Record layout for the output data set (continued)					
Offset (Dec)	Offset (Hex)	Type	Length	Name	Description
201	C9	Character	1	csuowscope	UOW scope in hexadecimal: <b>01</b> local <b>02</b> shared
202	CA	Integer	4	lsegment	The segment number of the data, starting from 1.
206	CE		Variable		Data part
206	CE	Character	1	csbora	If csverb is ALTER, indicates whether the data is the 'before' or 'after' copy of the object. <b>B</b> before <b>A</b> after
207	CF	Character	Variable	csvardata	Message or object data. Length as given in ldataen. Message data is the MQMD followed by the body of the message. <b>V 9.3.4</b> If the message has message properties, these are represented as an MQRFH2 header chained into the body of the message following the MQMD, and any MQMDE, MQXQH, and MQDLH header. See Note “1” on page 2793.

**Notes:** **V 9.3.4**

1. MQPUTPRP was used by versions prior to IBM MQ 9.3.4 for put of a message with message properties. The properties were in the message data in an internal format that made replay challenging.
2. Message replay needs to remove the MQRFH2 header added by log extract, and convert it to a message handle before putting the message, to preserve the PROPCTL(V6COMPAT) behavior.  
The CSQ4LOGS replay sample does this.

## **z/OS** The queue sharing group utility (CSQ5PQSG) on z/OS

You can use the CSQ5PQSG utility program to add queue sharing group and queue manager definitions to the IBM MQ Db2 tables, and to remove them.

The CSQ5PQSG utility can also be used to verify the consistency of Db2 object definitions for queue manager, CF structure, and shared queue objects, within a queue sharing group.

- [Invoking the queue sharing group utility](#)
- [Syntax, keywords, and parameters](#)
- [Example](#)

## Invoking the queue sharing group utility

Figure 40 on page 2794 shows an example of the JCL used to invoke the CSQ5PQSG utility.

```
//S001 EXEC PGM=CSQ5PQSG,REGION=4M,  
//      PARM='function,function parameters'  
//STEPLIB DD DSN=th1qua1.SCSQANLE,DISP=SHR  
//         DD DSN=th1qua1.SCSQAUTH,DISP=SHR  
//         DD DSN=db2qua1.SDSNLOAD,DISP=SHR  
//SYSPRINT DD SYSOUT=*
```

Figure 40. Sample JCL to invoke the CSQ5PQSG utility

### Data definition statements

The CSQ5PQSG utility requires data definition statements with the following DDname:

#### SYSPRINT

This statement is required; it names the data set for print output. The logical record length (LRECL) is 125.

### Syntax, keywords, and parameters

#### Queue sharing group utility

```
► PARM=' ——— ADD QMGR — ,qmgr-name,qsg-name,dsg-name,DB2-ssid ——— ' —►  
——— ADD QSG — ,qsg-name,dsg-name,DB2-ssid ———  
——— REMOVE QMGR — ,qmgr-name,qsg-name,dsg-name,DB2-ssid ———  
——— REMOVE QSG — ,qsg-name,dsg-name,DB2-ssid ———  
——— MIGRATE DSG — ,dsg-name,DB2-ssid ———  
——— MIGRATE QSG — ,qsg-name,dsg-name,DB2-ssid ———  
——— FORCE QMGR — ,qmgr-name,qsg-name,dsg-name,DB2-ssid ———  
——— VERIFY QSG — ,qsg-name,dsg-name,DB2-ssid ———
```

A queue sharing group name (*qsg-name*) can have up to 4 characters, comprising uppercase A-Z, 0-9, \$, #, @. It must not start with a numeric. For implementation reasons, names of less than 4 characters are padded internally with @ symbols, so do not use names ending in @.

The queue sharing group name must be different from any of the queue manager names within the queue sharing group.

#### PARM

This field contains the function request followed by the function-specific parameters. These are described in the following text:

#### ADD QMGR

Add a queue manager record into the CSQ.ADMIN\_B\_QMGR table. This operation completes successfully only if all the following conditions are met:

- A corresponding queue sharing group record exists in the CSQ.ADMIN\_B\_QSG table.
- The queue manager entry does not exist in the CSQ.ADMIN\_B\_QMGR table as the member of a different queue sharing group.
- There is no member entry in the XCF group with a different QMGR number value than the one created by the utility when you add a record to the CSQ.ADMIN\_B\_QMGR table.

Note that it does not matter whether the queue manager being added is active or inactive when the ADD QMGR function is being performed.

If there are members in the XCF group without the corresponding entries in the Db2 table, you can use the utility to add them. Add queue managers in the order that is indicated by the CSQU524I messages that are issued by the queue sharing group utility (CSQ5PQSG) when it is run with the **VERIFY QSG** parameter.

If a queue manager exists in Db2 table CSQ.ADMIN\_B\_QMGR, but is missing from MVS XCF group, you can run this utility to restore the appropriate XCF group entry, as indicated by CSQ5010E message.

***qmgr-name***

The queue manager name

***qsg-name***

The queue sharing group name

***dsg-name***

The Db2 data-sharing group name

***DB2-ssid***

The Db2 subsystem ID

**ADD QSG**

Add a queue sharing group record into the CSQ.ADMIN\_B\_QSG table.

***qsg-name***

The queue sharing group name

***dsg-name***

The Db2 data-sharing group name

***DB2-ssid***

The Db2 subsystem ID

**REMOVE QMGR**

Remove a queue manager record from the CSQ.ADMIN\_B\_QMGR table. This only completes successfully if the queue manager has either never been started, or terminated normally from its last execution.

***qmgr-name***

The queue manager name

***qsg-name***

The queue sharing group name

***dsg-name***

The Db2 data-sharing group name

***DB2-ssid***

The Db2 subsystem ID

**REMOVE QSG**

Remove a queue sharing group record from the CSQ.ADMIN\_B\_QSG table. This only completes successfully if no queue managers are defined to the queue sharing group.

***qsg-name***

The queue sharing group name

***dsg-name***

The Db2 data-sharing group name

***DB2-ssid***

The Db2 subsystem ID

**MIGRATE DSG**

Verify that all the queue managers in the data-sharing group are at a version that is compatible with IBM MQ 9.3.

***dsg-name***

The Db2 data-sharing group name

***DB2-ssid***

The Db2 subsystem ID

This function does not do the migration, which involves several steps.

**MIGRATE QSG**

Verify that all the queue managers in the data-sharing group are at a version that is compatible with IBM MQ 9.3.

The MIGRATE QSG and MIGRATE DSG functions perform the same function. The only difference is in the scope of the processing. MIGRATE QSG works on a single queue sharing group only, MIGRATE DSG works on all queue sharing groups that are defined within the data-sharing group.

***qsg-name***

The queue sharing group name

***dsg-name***

The Db2 data-sharing group name

***DB2-ssid***

The Db2 subsystem ID

This function does not do the migration, which involves several steps.

**FORCE QMGR**

Remove a queue manager record from the CSQ.ADMIN\_B\_QMGR table, even if the queue manager has terminated abnormally.

Use the **FORCE** option, rather than **REMOVE**, to remove the last queue manager in a queue sharing group.

**Attention:** This can override IBM MQ efforts to maintain data in a consistent state. Only use this function when you cannot carry out the procedure for removing a queue manager from a queue sharing group on page [Removing a queue manager from a queue sharing group](#).

***qmgr-name***

The queue manager name

***qsg-name***

The queue sharing group name

***dsg-name***

The Db2 data-sharing group name

***DB2-ssid***

The Db2 subsystem ID

**VERIFY QSG**

Validate the consistency of the Db2 object definitions for queue manager, CF structure, and shared queue objects, within the queue sharing group.

***qsg-name***

The queue sharing group name

***dsg-name***

The Db2 data-sharing group name

***DB2-ssid***

The Db2 subsystem ID

## Example

The following sample JCL adds an entry for queue manager QM01 into queue sharing group QSG1. It specifies a connection to Db2 subsystem DB2A, which is a member of Db2 data-sharing group DSN510PG.

```
//S001 EXEC PGM=CSQ5PQSG,REGION=4M,  
//      PARM='ADD QMGR,QM01,QSG1,DSN510PG,DB2A'  
//STEPLIB DD DSN=th1qua1.SCSQANLE,DISP=SHR  
//        DD DSN=th1qua1.SCSQAUTH,DISP=SHR  
//        DD DSN=db2qua1.SDSNLOAD,DISP=SHR  
//SYSPRINT DD SYSOUT=*
```

Figure 41. Using the queue sharing group utility to add a queue manager into a queue sharing group

z/OS

## The active log preformat utility (CSQJUFMT) on z/OS

You can use the CSQJUFMT utility to format active log data sets before they are used by a queue manager.

If the active log data sets are preformatted by the utility, log write performance is improved on the queue manager's first pass through the active logs. If the utility is not used, the queue manager must format each log control interval at log write time before it is used. On the second and subsequent passes through the active log data sets, the log control intervals already contain data, so need no further formatting, and no performance benefit accrues.

### Invoking the CSQJUFMT utility

You can only run the CSQJUFMT program before starting the queue manager that use the logs.

**Note:** Do not use this utility to format a log data set after the queue manager has started, or data will be lost.

```
EXEC PGM=CSQJUFMT
```

Each step running the CQJUFMT utility formats a single active log data set. Add additional CSQJUFMT steps for each active log being created.



**Attention:** JCL limits the number of steps in a single job to 255. If you are formatting more than 255 active log data sets, you will need to run multiple jobs.

These DD statements should be provided:

#### **SYSPRINT**

This statement is required to specify a data set or print spool class for print output.

#### **SYSUT1**

This statement identifies the log data set to be preformatted.

```

//JOB LIB DD DISP=SHR, DSN=thlqual.SCSQANLE
// DD DISP=SHR, DSN=thlqual.SCSQAUTH
// *
//JUFMT11 EXEC PGM=CSQJUFMT
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=OLD, DSN=h1q.LOGCOPY1.DS01
// *
//JUFMT21 EXEC PGM=CSQJUFMT
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=OLD, DSN=h1q.LOGCOPY2.DS01

```

Figure 42. Example of the JCL used to invoke the CSQJUFMT utility

Sample JCL is supplied in thlqual.SCSQPROC (CSQ4LFMT) for preformatting a newly defined dual log data set. It contains two steps, one step to format each of the copies of the log data set.

## z/OS The dead-letter queue handler utility (CSQUDLQH) on z/OS

You can use the default dead-letter utility (CSQUDLQH) to handle message written to the dead-letter queue.

A *dead-letter queue* (DLQ) is a holding queue for messages that cannot be delivered to their destination queues. Every queue manager in a network can have an associated DLQ.

Queue managers, message channel agents, and applications can put messages on the DLQ. All messages on the DLQ can be prefixed with a *dead-letter header* structure, MQDLH. Messages put on the DLQ by a queue manager or by a message channel agent always have a dead-letter header; ensure that applications putting messages on the DLQ also supply a dead-letter header structure. The *Reason* field of the MQDLH structure contains a reason code that identifies why the message is on the DLQ.

Implement a routine that runs regularly to process messages on the DLQ. Such a routine is called a *dead-letter queue handler*. IBM MQ supplies a default *dead-letter queue handler* (DLQ handler) called CSQUDLQH. A user-written *rules table* supplies instructions to the DLQ handler, for processing messages on the DLQ. That is, the DLQ handler matches messages on the DLQ against entries in the rules table. When a DLQ message matches an entry in the rules table, the DLQ handler performs the action associated with that entry.

## z/OS Invoking the DLQ handler on z/OS

Use this topic to understand how to invoke the CSQUDLQH utility program, and its data definition statements.

The CSQUDLQH utility program runs as a z/OS batch program. Specify the name of the dead-letter queue that you want to process and the queue manager on which it resides. You can do this in one of the following two ways (in these examples, the dead-letter queue is called CSQ1.DEAD.QUEUE and the queue manager is called CSQ1):

1. The names can be specified as positional parameters in the PARM parameter of the EXEC statement within the submitted JCL, for example:

```

//READQ EXEC PGM=CSQUDLQH,
// PARM='CSQ1.DEAD.QUEUE CSQ1'

```

Figure 43. Specifying the queue manager and dead-letter queue names for the dead-letter queue handler in the JCL

2. The names can be specified in the rules table, for example:

```
INPUTQ(CSQ1.DEAD.QUEUE) INPUTQM(CSQ1)
```

*Figure 44. Specifying the queue manager and dead-letter queue names for the dead-letter queue handler in the rules table*

Any parameters that you specify in the PARM parameter override those in the rules table. If you specify only one parameter in the PARM statement, it is used as the name of the dead-letter queue. The rules table is taken from the SYSIN data set.

For further information on the keywords you can specify, to match and process pattern and action keywords, see “Rules (patterns and actions) on z/OS” on page 2801.

## Stopping the DLQ handler

The CSQUDLQH utility is stopped when any of the following conditions is true:

- The dead letter queue is empty for a specified amount of time as configured by the WAIT control data keyword.
- The dead letter queue is set to GET(DISABLED).
- The queue manager is quiesced.
- The CSQUDLQH job is cancelled.

Messages generated during the handling of the queue are written to the standard output when the CSQUDLQH utility ends in a controlled manner. If the handler is cancelled, it does not generate these messages.

## Data definition statements

CSQUDLQH requires DD statements with these DDnames:

### **SYSOUT**

This statement is required; it names the data set for print output. You can specify the logical record length (LRECL) and block size (BLKSIZE) for this output data set.

### **SYSIN**

This statement is required; it names the input data set containing the rules table that specifies what the utility is to do. The logical record length (LRECL) is 80.

## Sample JCL

```
//READQ EXEC PGM=CSQUDLQH,  
//      PARM='CSQ1.DEAD.QUEUE CSQ1'  
//STEPLIB DD DSN=thlqua1.SCSQAUTH,DISP=SHR  
//      DD DSN=thlqua1.SCSQLOAD,DISP=SHR  
//      DD DSN=thlqua1.SCSQANLE,DISP=SHR  
//SYSOUT DD SYSOUT=*  
//SYSIN DD *  
INPUTQM(CSQ2) INPUTQ('CSQ2.DEAD.QUEUE')  
ACTION(RETRY)  
/*
```

*Figure 45. Sample JCL to invoke the CSQUDLQH utility*

## The DLQ handler rules table on z/OS

The DLQ handler rules table defines how the DLQ handler is to process messages that arrive on the DLQ.

There are two types of entry in a rules table:

- The first entry in the table, which is optional, contains “Control data” on page 2800.
- All other entries in the table are *rules* for the DLQ handler to follow. Each rule consists of a *pattern* (a set of message characteristics) that a message is matched against, and an *action* to be taken when a message on the DLQ matches the specified pattern. There must be at least one rule in a rules table.

Each entry in the rules table comprises one or more keywords.

See “Rules table conventions on z/OS” on page 2804 for information about the syntax of the rules table.

See [Rules \(patterns and actions\)](#) for information about how the pattern-matching, and action keywords control the CSQUDLQH utility

### Control data

This section describes the keywords that you can include in a control-data entry in a DLQ handler rules table.

- All keywords are optional.
- If a control-data entry is included in the rules table, it must be the first entry in the table.
- The default value for a keyword, if any, is underlined>.
- The vertical line (|) separates alternatives. You can specify only one of these.

#### **INPUTQ (QueueName| ' ' (default) )**

Specifies the name of the DLQ that you want to process:

1. If you specify a queue name in the PARM parameter of the EXEC statement, this overrides any INPUTQ value in the rules table.
2. If you do not specify a queue name in the PARM parameter of the EXEC statement, the INPUTQ value in the rules table is used.
3. If you do not specify a queue name in the PARM parameter of the EXEC statement or the rules table, the dead-letter queue named *qmgr-name.DEAD.QUEUE* is used if it has been defined. If this queue does not exist, the program fails and returns error message CSQU224E, giving the reason code for the error.

#### **INPUTQM (QueueManagerName| ' ' (default) )**

Specifies the name of the queue manager that owns the DLQ named on the INPUTQ keyword.

1. If you specify a queue manager name in the PARM parameter of the EXEC statement, this overrides any INPUTQM value in the rules table.
2. If you do not specify a queue manager name in the PARM parameter of the EXEC statement, the INPUTQM value in the rules table is used.
3. If you do not specify a queue manager name in the PARM parameter of the EXEC statement or the rules table, the default queue manager is used (if one has been defined using CSQBDEFV). If not, the program fails and returns error message CSQU220E, giving the reason code for the error.

#### **RETRYINT (Interval|60 (default) )**

Specifies the interval, in seconds, at which the DLQ handler should attempt to reprocess messages on the DLQ that could not be processed at the first attempt, and for which repeated attempts have been requested. The DLQ handler reprocesses messages after it has first browsed to the end of the queue.

The default is 60 seconds.



**WAIT (YES (default) |NO|nnn)**

Specifies whether the DLQ handler should wait for further messages to arrive on the DLQ when it detects that there are no further messages that it can process.

**YES**

The DLQ handler waits indefinitely.

**NO**

The DLQ handler terminates when it detects that the DLQ is either empty or contains no messages that it can process.

**nnn**

The DLQ handler waits for *nnn* seconds for new work to arrive after it detects that the queue is either empty or contains no messages that it can process, before terminating.

Specify a value in the range 1 through 999 999.

Specify WAIT (YES) for busy DLQs, and WAIT (NO) or WAIT (*nnn*) for DLQs that have a low level of activity. If the DLQ handler is allowed to terminate, you can use triggering to invoke it when needed.

## **Rules (patterns and actions) on z/OS**

The DLQ handler is controlled with a series of pattern-matching and action keywords described here.

Figure 46 on page 2801 shows an example rule from a DLQ handler rules table.

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
ACTION (RETRY) RETRY (3)
```

Figure 46. An example rule from a DLQ handler rules table

This section describes the keywords that you can include in a rules table. It begins with a description of the pattern-matching keywords (those keywords against which messages on the DLQ are matched). It then describes the action keywords (those keywords that determine how the DLQ handler is to process a matching message).

- All keywords except ACTION are optional.
- The default value for a keyword, if any, is underlined. For most keywords, the default value is asterisk (\*), which matches any value.
- The vertical line (|) separates alternatives. You can specify only one of these keywords.

The keywords can be grouped as follows:

- The pattern-matching keywords
- The action keywords

### The pattern-matching keywords

The pattern-matching keywords, are described in the following table. You use these keywords to specify values against which messages on the DLQ are matched. All pattern-matching keywords are optional.

**APPLIDAT (ApplIdentityData|\* (default) )**

The *ApplIdentityData* value of the message on the DLQ, specified in the message descriptor, MQMD.

**APPLNAME (PutAppName|\* (default) )**

The name of the application that issued the MQPUT or MQPUT1 call, as specified in the *PutAppName* field of the message descriptor, MQMD, of the message on the DLQ.

**APPLTYPE (PutApplType|\* (default) )**

The *PutApplType* value specified in the message descriptor, MQMD, of the message on the DLQ.

**DESTQ (QueueName|\* (default) )**

The name of the message queue for which the message is destined.

**DESTQM (QueueManagerName|\* (default) )**

The queue manager name for the message queue for which the message is destined.

**FEEDBACK (Feedback|\* (default) )**

Describes the nature of the report when the *MsgType* value is MQMT\_REPORT.

You can use symbolic names. For example, you can use the symbolic name MQFB\_COA to identify those messages on the DLQ that require confirmation of their arrival on their destination queues. A few symbolic names are not accepted by the utility and lead to a syntax error. In these cases, you can use the corresponding numeric value.

**FORMAT (Format|\* (default) )**

The name that the sender of the message uses to describe the format of the message data.

**MSGTYPE (MsgType|\* (default) )**

The message type of the message on the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQMT\_REQUEST to identify those messages on the DLQ that require replies.

**PERSIST (Persistence|\* (default) )**

The persistence value of the message. (The persistence of a message determines whether it survives restarts of the queue manager.)

You can use symbolic names. For example, you can use the symbolic name MQPER\_PERSISTENT to identify those messages on the DLQ that are persistent.

**REASON (ReasonCode|\* (default) )**

The reason code that describes why the message was put to the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQRC\_Q\_FULL to identify those messages placed on the DLQ because their destination queues were full. A few symbolic names are not accepted by the utility and lead to a syntax error. In these cases, you can use the corresponding numeric value.

**REPLYQ (QueueName|\* (default) )**

The reply-to queue name specified in the message descriptor, MQMD, of the message on the DLQ.

**REPLYQM (QueueManagerName|\* (default) )**

The queue manager name of the reply-to queue specified in the REPLYQ keyword.

**USERID (UserIdentifier|\* (default) )**

The user ID of the user who originated the message on the DLQ, as specified in the message descriptor, MQMD.

## The action keywords

The action keywords are described in the following table. You use these keywords to describe how a matching message is processed.

**ACTION ( DISCARD | IGNORE | RETRY | FWD)**

The action taken for any message on the DLQ that matches the pattern defined in this rule.

**DISCARD**

Causes the message to be deleted from the DLQ.

**IGNORE**

Causes the message to be left on the DLQ.

**RETRY**

Causes the DLQ handler to try again to put the message on its destination queue.

**FWD**

Causes the DLQ handler to forward the message to the queue named on the FWDQ keyword.

You must specify the ACTION keyword. The number of attempts made to implement an action is governed by the RETRY keyword. The RETRYINT keyword of the control data controls the interval between attempts.

**CONVERT (YES (default) |NO)**

By default, this keyword is set to CONVERT(YES). When forwarding or retrying a message, the DLQ handler performs an MQGET with MQGMO\_CONVERT; that is, it converts the message data to the CCSID and encoding of the queue manager.

However, setting CONVERT(NO) forwards or retries the message without converting the message contents.

**FWDQ (QueueName |&DESTQ |&REPLYQ)**

The name of the message queue to which the message is forwarded when you select the ACTION keyword.

**QueueName**

This parameter is the name of a message queue. FWDQ(' ') is not valid.

**&DESTQ**

Takes the queue name from the *DestQName* field in the MQDLH structure.

**&REPLYQ**

Takes the name from the *ReplyToQ* field in the message descriptor, MQMD. You can specify REPLYQ (?\*) in the message pattern to avoid error messages, when a rule specifying FWDQ (&REPLYQ), matches a message with a blank *ReplyToQ* field.

**FWDQM (QueueManagerName|&DESTQM|&REPLYQM|' ' (default))**

The queue manager of the queue to which a message is forwarded.

**QueueManagerName**

This parameter defines the queue manager name for the queue to which the message is forwarded when you select the ACTION (FWD) keyword.

**&DESTQM**

Takes the queue manager name from the *DestQMgrName* field in the MQDLH structure.

**&REPLYQM**

Takes the name from the *ReplyToQMgr* field in the message descriptor, MQMD.

..

The local queue manager.

**HEADER (YES (default) |NO)**

Whether the MQDLH should remain on a message for which ACTION (FWD) is requested. By default, the MQDLH remains on the message. The HEADER keyword is not valid for actions other than FWD.

**PUTAUT (DEF (default) |CTX)**

The authority with which messages should be put by the DLQ handler:

**DEF**

Puts messages with the authority of the DLQ handler itself.

**CTX**

Causes the messages to be put with the authority of the user ID in the message context. You must be authorized to assume the identity of other users, if you specify PUTAUT (CTX).

**RETRY (RetryCount|1 (default) )**

The number of times that an action should be attempted (at the interval specified on the RETRYINT keyword of the control data). Specify a value in the range 1 through 999 999 999.

**Note:** The count of attempts made by the DLQ handler to implement any particular rule is specific to the current instance of the DLQ handler; the count does not persist across restarts. If you restart the DLQ handler, the count of attempts made to apply a rule is reset to zero.

## **Rules table conventions on z/OS**

Use this topic to understand the conventions used in the CSQUDLQH rule table.

The rules table must adhere to the following conventions regarding its syntax, structure, and contents:

- A rules table must contain at least one rule.
- Keywords can occur in any order.
- A keyword can be included once only in any rule.
- Keywords are not case-sensitive.
- A keyword and its parameter value can be separated from other keywords by at least one blank or comma.
- Any number of blanks can occur at the beginning or end of a rule, and between keywords, punctuation, and values.
- Each rule must begin on a new line.
- For reasons of portability, the significant length of a line should not be greater than 72 characters.
- Use the plus sign (+) as the last non-blank character on a line to indicate that the rule continues from the first non-blank character in the next line. Use the minus sign (-) as the last non-blank character on a line to indicate that the rule continues from the start of the next line. Continuation characters can occur within keywords and parameters.

For example:

```
APPLNAME('ABC+
D')
```

results in 'ABCD'.

```
APPLNAME('ABC-
D')
```

results in 'ABC D'.

- Comment lines, which begin with an asterisk (\*), can occur anywhere in the rules table.
- Blank lines are ignored.

Each entry in the DLQ handler rules table comprises one or more keywords and their associated parameters. The parameters must follow these syntax rules:

- Each parameter value must include at least one significant character. The delimiting quotation marks in following examples are not considered significant. For example, these parameters are valid:

**FORMAT('ABC')**

3 significant characters

**FORMAT(ABC)**

3 significant characters

**FORMAT('A')**

1 significant character

**FORMAT(A)**

1 significant character

**FORMAT('')**

1 significant character

These parameters are not valid because they contain no significant characters:

– FORMAT('')

– FORMAT( )

– FORMAT()

– FORMAT

- Wildcard characters are supported. You can use the question mark (?) instead of any single character, except a trailing blank. You can use the asterisk (\*) instead of zero or more adjacent characters. The asterisk (\*) and the question mark (?) are **always** interpreted as wildcard characters in parameter values.
- You cannot include wildcard characters in the parameters of these keywords: ACTION, HEADER, RETRY, FWDQ, FWDQM, and PUTAUT.
- Trailing blanks in parameter values, and in the corresponding fields in the message on the DLQ, are not significant when performing wildcard matches. However, leading and embedded blanks within strings in quotation marks are significant to wildcard matches.
- Numeric parameters cannot include the question mark (?) wildcard character. You can include the asterisk (\*) instead of an entire numeric parameter, but the asterisk cannot be included as part of a numeric parameter. For example, these are valid numeric parameters:

**MSGTYPE(2)**

Only reply messages are eligible

**MSGTYPE(\*)**

Any message type is eligible

**MSGTYPE(' \*')**

Any message type is eligible

However, MSGTYPE(' 2\* ') is not valid, because it includes an asterisk (\*) as part of a numeric parameter.

- Numeric parameters must be in the range zero through 999 999 999 unless otherwise stated. If the parameter value is in this range, it is accepted, even if it is not currently valid in the field to which the keyword relates. You can use symbolic names for numeric parameters.
- If a string value is shorter than the field in the MQDLH or MQMD to which the keyword relates, the value is padded with blanks to the length of the field. If the value, excluding asterisks, is longer than the field, an error is diagnosed. For example, these are all valid string values for an eight character field:

**'ABCDEFGH'**

8 characters

**'A\*C\*E\*G\*I'**

5 characters excluding asterisks

**'\*A\*C\*E\*G\*I\*K\*M\*O\*'**

8 characters excluding asterisks

- Strings that contain blanks, lowercase characters, or special characters other than period (.), forward slash (/), underscore (\_), and percent sign (%) must be enclosed in single quotation marks. Lowercase characters not enclosed in quotation marks are folded to uppercase. If the string includes a quotation, two single quotation marks must be used to denote both the beginning and the end of the quotation. When the length of the string is calculated, each occurrence of double quotation marks is counted as a single character.

## Processing the rules table on z/OS

Use this topic to understand how the CSQUDLQH utility processes the rules table.

The DLQ handler searches the rules table for a rule with a pattern that matches a message on the DLQ. The search begins with the first rule in the table, and continues sequentially through the table. When a rule with a matching pattern is found, the rules table attempts the action from that rule. The DLQ handler increments the retry count for a rule by 1 whenever it attempts to apply that rule. If the first attempt fails, the attempt is repeated until the count of attempts made matches the number specified on the RETRY keyword. If all attempts fail, the DLQ handler searches for the next matching rule in the table.

This process is repeated for subsequent matching rules until an action is successful. When each matching rule has been attempted the number of times specified on its `RETRY` keyword, and all attempts have failed, `ACTION (IGNORE)` is assumed. `ACTION (IGNORE)` is also assumed if no matching rule is found.

For further information, see [Ensuring that all DLQ messages are processed](#).

**Note:**

1. Matching rule patterns are sought only for messages on the DLQ that begin with an `MQDLH`. If the dead-letter queue handler encounters one or more messages that are not prefixed by an `MQDLH`, it issues an information message to report this. Messages that do not contain an `MQDLH` are not processed by the DLQ handler and remain on the dead-letter queue until dealt with by another method.
2. All pattern keywords can default, so that a rule can consist of an action only. Note, however, that action-only rules are applied to all messages on the queue that have `MQDLHs` and that have not already been processed in accordance with other rules in the table.
3. The rules table is validated when the DLQ handler starts, and errors flagged at that time. You can change the rules table at any time, but those changes do not come into effect until the DLQ handler is restarted.
4. The DLQ handler does not alter the content of messages, of the `MQDLH`, or of the message descriptor. The DLQ handler always puts messages to other queues with the message option `MQPMO_PASS_ALL_CONTEXT`.
5. Consecutive syntax errors in the rules table might not be recognized because the validation of the rules table is designed to eliminate the generation of repetitive errors.
6. The DLQ handler opens the DLQ with the `MQOO_INPUT_AS_Q_DEF` option.
7. Do not run applications that perform `MQGET` calls against the queue at the same time as the DLQ handler. This includes multiple instances of the DLQ handler. There is typically a one-to-one relationship between the dead-letter queue and the DLQ handler.

## Ensuring that all DLQ messages are processed

The DLQ handler keeps a record of all messages on the DLQ that have been seen but not removed. If you use the DLQ handler as a filter to extract a small subset of the messages from the DLQ, the DLQ handler still keeps a record of those messages on the DLQ that it did not process. Also, the DLQ handler cannot guarantee that new messages arriving on the DLQ will be seen, even if the DLQ is defined as first-in first-out (FIFO). Therefore, if the queue is not empty, the DLQ is periodically rescanned to check all messages. For these reasons, ensure that the DLQ contains as few messages as possible. If messages that cannot be discarded or forwarded to other queues (for whatever reason) are allowed to accumulate on the queue, the workload of the DLQ handler increases and the DLQ itself is in danger of filling up.

You can take specific measures to enable the DLQ handler to empty the DLQ. For example, do not use `ACTION (IGNORE)`, which leaves messages on the DLQ. (Remember that `ACTION (IGNORE)` is assumed for messages that are not explicitly addressed by other rules in the table.) Instead, for those messages that you would otherwise ignore, use an action that moves the messages to another queue. For example:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

Similarly, the final rule in the table should be a catchall to process messages that have not been addressed by earlier rules in the table. For example, the final rule in the table could be something like this:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

This forwards messages that fall through to the final rule in the table to the queue `REALLY.DEAD.QUEUE`, where they can be processed manually. If you do not have such a rule, messages are likely to remain on the DLQ indefinitely.

## An example DLQ handler rules table on z/OS

Use this topic as an example of the DLQ handler rules table.

Here is an example rules table that contains a single control-data entry and several rules:

```
*****
*           An example rules table for the CSQUDLQH utility           *
*****
* Control data entry
* -----
* If no queue manager name is supplied as an explicit parameter to CSQUDLQH,
* use the default queue manager.
* If no queue name is supplied as an explicit parameter to CSQUDLQH, use the
* DLQ defined for the queue manager.
*
inputqm(' ') inputq(' ')

* Rules
* -----

* The first check deals with attempted security violations.
* If a message was placed on the DLQ because the putter did not have the
* appropriate authority for the target queue, forward the message to a queue
* for manual inspection.

REASON(MQRC_NOT_AUTHORIZED) ACTION(FWD) +
FWDQ(DEADQ.MANUAL.SECURITY)

* The next set of rules with ACTION (RETRY) try to deliver the message to the
* intended destination.

* If a message is placed on the DLQ because its destination queue is full,
* attempt to forward the message to its destination queue. Make 5 attempts at
* approximately 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because there has been a problem starting the
* application by triggering, forward the message to another queue for manual
* inspection.

REASON(MQFB_APPL_CANNOT_BE_STARTED) ACTION(FWD) +
FWDQ(DEADQ.MANUAL.TRIGGER)

* If a message is placed on the DLQ because of a put inhibited condition, attempt
* to forward the message to its destination queue. Make 5 attempts at
* approximately 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation often send messages with incorrect addresses. When we find
* a request from the AAAA corporation, we return it to the DLQ (DEADQ) of the
* reply-to queue manager (&REPLYQM). The AAAA DLQ handler attempts to
* redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation requests that we try sending messages to queue manager
* BBB2 if queue manager BBB1 is unavailable.

DESTQM(BBB1) +
ACTION(FWD) FWDQ(&DESTQ) FWDQM(BBB2) HEADER(NO)

* The CCCC corporation is very security conscious, and believes that none of its
* messages will ever end up on one of our DLQs. If we do see a message from a
* CCCC queue manager on our DLQ, we send it to a special destination in the CCCC
* organization where the problem is investigated.

REPLYQM(CCCC.*) +
ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent risk being lost when a queue manager terminates.
* If an application is sending nonpersistent messages, it will be able to cope with
* the message being lost, so we can afford to discard the message.

PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)
```

\* For performance and efficiency reasons, we like to keep the number of messages on the DLQ small. If we receive a message that has not been processed by an earlier rule in the table, we assume that it requires manual intervention to resolve the problem.

\* Some problems are best solved at the node where the problem was detected, and others are best solved where the message originated. We do not have the message origin, but we can use the REPLYQM to identify a node that has some interest in this message. Attempt to put the message onto a manual intervention queue at the appropriate node. If this fails, put the message on the manual intervention queue at this node.

```
REPLYQM('?*') +  
  ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)  
  
ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

## z/OS The BSDS conversion utility (CSQJUCNV) on z/OS

You can use the CSQJUCNV BSDS conversion utility to convert a version 1 bootstrap data set (BSDS) to version 2. CSQJUCNV runs as a batch job.

A version 1 BSDS supports 6 byte log RBA (Relative Byte Address) values. A version 2 BSDS can be used by queue managers running IBM MQ 8.0.0, or later, and supports 8 byte log RBA values. For more information about the change from 6 byte to 8 byte log RBA, see [Larger log Relative Byte Address](#).

**V 9.3.0** From IBM MQ for z/OS 9.3, there is no Db2 interaction.

**V 9.3.0** **V 9.3.0** From IBM MQ for z/OS 9.2.5, queue managers are automatically created with a version 2 BSDS. However, the CSQJUCNV utility is still provided so that migrated queue managers with a version 1 BSDS can be converted to version 2 if required.

The converted BSDSs are written to new data sets. These new data sets must be allocated with similar attributes to the current BSDS before the utility is run, and must be empty. A version 2 BSDS contains more data than a version 1 BSDS, therefore, you must ensure that the new data sets are allocated with sufficient available space. The sample JCL in `thlqual.SCSQPROC(CSQ4BSDS)` contains the recommended values when defining a new BSDS.

The current BSDSs are not modified and can be used to start the queue manager, should the attempt to convert the BSDSs and restart the queue manager with the new BSDS be unsuccessful.

### Important:

1. Only run this utility when the queue manager that owns the BSDS is stopped.
  2. Do not attempt to start the queue manager with the new BSDS until the utility has completed successfully. If a queue manager is started with a BSDS that is the output of an unsuccessful or incomplete conversion, it terminates with reason code `00D10121`.
  3. To use this utility, your user ID of the job must have read and write access to both the old and new BSDSs.
- [“Invoking the CSQJUCNV utility” on page 2808](#)
  - [“Data definition \(DD\) statements” on page 2809](#)

### Invoking the CSQJUCNV utility

The utility runs as a z/OS batch program. Figure 1 shows an example of the JCL used to invoke the CSQJUCNV utility for a queue manager that is a member of a queue sharing group.

**V 9.3.0** **V 9.3.0**



```

//CONVERT EXEC PGM=CSQJUCNV,REGION=32M
//STEPLIB DD DSN=thlqual.SCSQAUTH,DISP=SHR
//          DD DSN=thlqual.SCSQANLE,DISP=SHR
//          DD DSN=db2qual.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=h1q.BSDS01,DISP=SHR
//SYSUT2 DD DSN=h1q.BSDS02,DISP=SHR
//SYSUT3 DD DSN=newh1q.BSDS01,DISP=OLD
//SYSUT4 DD DSN=newh1q.BSDS02,DISP=OLD

```

Figure 47. Sample JCL to invoke the CSQJUCNV utility

Sample JCL to run the utility is also provided in thlqual.SCSQPROC (CSQ4BCNV).

**V 9.3.0** **V 9.3.0** From IBM MQ for z/OS 9.2.5 CSQJUCNV takes no parameters.

## Data definition (DD) statements

CSQJUCNV recognizes DD statements with the following DD names:

### **SYSUT1**

Specifies the old BSDS that is to be converted. This statement is required.

### **SYSUT2**

Specifies the second copy of the old BSDS that is to be converted. If you are using dual BSDS, you should specify this.

### **SYSUT3**

Specifies the new, converted BSDS. This statement is required.

### **SYSUT4**

Specifies the second copy of the converted BSDS. This statement is required if the installation uses dual BSDSs; otherwise, it is optional.

### **SYSPRINT**

Contains the output messages from the conversion utility. This statement is required.

**z/OS**

## The message security policy utility (CSQOUTIL)

The Advanced Message Security policy utility is provided to manage security policies that specify the cryptographic encryption and signature algorithms for encrypting and authenticating messages that flow through queues.

Using this utility program, you can display, define, alter, delete and export security policies.

The CSQOUTIL utility program runs as a z/OS batch utility that accepts **SYSIN** command input. Sample JCL to run the utility is provided in member CSQ40CFG of thlqual.SCSQPROC.

```

-----
//CSQ40CFG JOB 1,CSQ0,CLASS=A,MSGCLASS=X
//CSQ40CFG EXEC PGM=CSQOUTIL,
//          PARM='ENVAR("_CEE_ENVFILE_S=DD:ENVAR") /'
//STEPLIB DD DSN=thlqual.SCSQANLE,DISP=SHR
//          DD DSN=thlqual.SCSQAUTH,DISP=SHR
//ENVAR DD DSN=thlqual.SCSQPROC(CSQ40ENV),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
dspmqspl -m qmgr
/*
-----

```

The utility accepts the following commands:

### **dspmqspl**

Display or export information about one or more security policies.

## setmqspl

Define, alter or remove a security policy

For information on how to use these commands to manage security policies see [Managing security policies](#).

## General usage notes

When specifying distinguished names (DNs) that have embedded blanks, you must enclose the entire DN in double quotes ("). For example:

```
-a "CN=John Smith,O=IBM,C=US"  
-r "CN=JSmith,O=IBM Australia,C=AU"
```

Arguments that would exceed column 80 of a SYSIN input record can be continued on subsequent SYSIN records provided those arguments are enclosed in double quotes ("), and relevant continuations resume in column 1 of subsequent SYSIN records.

When exporting policy information using **dspmqspl** with the `-export` parameter the output is written to an additional DD named EXPORT. The EXPORT DD can be `SYSOUT=*`, a sequential data set, or the member of a partitioned data set. The record format is fixed block and the logical record length is 80. The output is in the form of one or more **setmqspl** commands that can subsequently be used as input to CSQOUTIL.

## Specific security information

To use this utility you need authority to connect to the queue manager as a batch application. This authority is granted by giving READ access to the hlq.BATCH profile in the MQCONN class.

You also need authority to put messages to the queue SYSTEM.PROTECTION.POLICY.QUEUE. This authority is granted by giving UPDATE access to the hlq.SYSTEM.PROTECTION.POLICY.QUEUE profile in the MQQUEUE class.


If command events have been enabled for the queue manager you also need put authority to the queue SYSTEM.ADMIN.COMMAND.EVENT. If configuration events have been enabled for the queue manager you need put authority to the queue SYSTEM.ADMIN.CONFIG.EVENT.

### Related concepts


[Security policies](#)

### Related reference

[“dspmqspl \(display security policy\)” on page 99](#)

Use the **dspmqspl** command to display a list of all policies and details of a named policy.  On z/OS you use the command with the CSQOUTIL utility.

[“setmqspl \(set security policy\)” on page 216](#)

Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy.  On z/OS you use the command with the CSQOUTIL utility.

 z/OS

## Display queue manager information utility (CSQUDSPM)

CSQUDSPM displays information about queue managers and provides the equivalent function to **dspmq** on Multiplatforms.

### Purpose

You use the CSQUDSPM utility to list all IBM MQ subsystems on the LPAR, regardless of what version of IBM MQ they are associated with.

Sample JCL, CSQ4DSPM, is provided for this purpose. The JCL is in the SCSQPROC data set.

## Packaging

The CSQUDSPM load module is provided in the SCSQAUTH data set with an alias called DSPMQ.

If you need to run CSQUDSPM from z/OS UNIX System Services (z/OS UNIX), you can follow this procedure:

1. Create an empty file in z/OS UNIX with the name `csqudspm` or `dspmq`. For example, issue the following command:

```
touch dspmq
```

2. Set the file permissions so that it is executable:

```
chmod 755 dspmq
```

3. Enable the sticky bit:

```
chmod +t dspmq
```

4. Set the APF authorized attribute:

```
extattr +a dspmq
```

To be authorized to issue the **extattr** command with the **+a** option, you must have at least read access to the BPX.FILEATTR.APF resource in the FACILITY class profile.

5. Ensure that the SCSQAUTH library is in the STEPLIB environment variable, and that all libraries in the STEPLIB concatenation are APF authorized. For example, to set the STEPLIB concatenation to contain the SCSQANLE and SCSQAUTH libraries, issue the following command:

```
export STEPLIB=thqual.SCSQANLE:thqual.SCSQAUTH
```

You can now execute the file you created to run CSQUDSPM from z/OS UNIX.

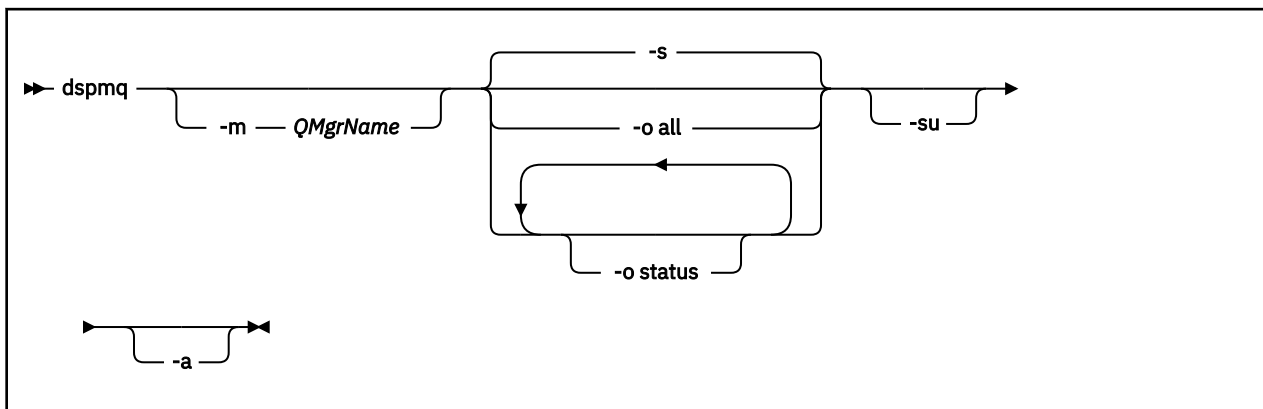
## Syntax

**V 9.3.0** From IBM MQ for z/OS 9.2.4, all the parameters for this utility are case insensitive. For example:

```
dspmq -o status
```

is the same as

```
dspmq -O STATUS
```



## Required parameters

None

## Optional parameters

- a**  
Displays information about running queue managers only.
- m *QMgrName***  
The queue manager for which to display details. If you do not specify a name, all queue managers on the LPAR are displayed.
- s**  
The operational status of the queue managers is displayed. This parameter is the default status setting.  
The parameter **-o status** is equivalent to **-s**.
- o all**  
All details about the queue manager, or queue managers, are displayed.
- o status**  
The operational status of the queue managers is displayed.
- su**  
Suppress information about queue managers whose version is unknown.  
An unknown version displays an INSTVER V . R . M of 0 . 0 . 0.

## Command output

Output name	Details
QMNAME	The name of the queue manager consisting of up to four characters. If the queue manager name is less than four characters the string is not padded. This parameter is always output. Examples: QMNAME(MQ21), QMNAME(MQ1)
STATUS	The status of the queue manager. Either Running or Stopped. This parameter is always output. Examples: STATUS(Running), STATUS(Stopped)
INSTVER	The version that the queue manager was last started up with, in the format V . R . M. <b>Note:</b> In the case of a queue manager that has not been started since the last IPL of the LPAR, the version of that queue manager cannot be obtained. In that situation, the INSTVER attribute displays a V . R . M of 0 . 0 . 0. Examples: INSTVER(8.0.0), INSTVER(9.0.1)
ERLYVER	The version of early code associated with the queue manager, in the format V . R . M. This is usually the same for all queue managers in the LPAR, as a single set of early code modules is loaded into the Link Pack Area (LPA) and should be used by all queue managers. Examples: ERLYVER(9.0.1)

Output name	Details
CMDPFX	The command prefix for the queue manager subsystem. This can be from one to eight characters long, and is not padded.  Examples: CMDPFX(!MQ21), CMDPFX(MQ90ATST)
QSGNAME	The name of the queue sharing group, that the queue manager is a member of, consisting of up to four characters. If the queue manager name is less than four characters the string is not padded. This parameter is always output.  If the queue manager is not a member of a queue sharing group then QSGNAME() is displayed.  QSGNAME information can only be obtained when the queue manager is running, that is, STATUS(Running). If the queue manager is stopped QSGNAME(Unknown) is displayed.  Example: QSGNAME(QSG1)
RELTYPE	The release type. The queue manager is running against a CD release if the value is <i>CDR</i> or against an LTS release if the value is <i>LTSR</i> .  In the case of a queue manager that has not been started since the last IPL of the LPAR, the release type of that queue manager cannot be obtained. In that situation, the RELTYPE attribute displays <i>Unknown</i> .

## Examples

### 1. Input:

```
dspmqr
```

#### Output:

```
QMNAME(QM01) STATUS(Stopped)
QMNAME(QM02) STATUS(Running)
QMNAME(QM03) STATUS(Stopped)
QMNAME(QM04) STATUS(Running)
```

### 2. Input:

```
dspmqr -o all
```

#### Output:

```
QMNAME(QM01) STATUS(Stopped) INSTVER(0.0.0) ERLYVER(9.2.0) CMDPFX(!QM01) QSGNAME(Unknown)
RELTYPE(Unknown)
QMNAME(QM02) STATUS(Running) INSTVER(9.2.0) ERLYVER(9.2.0) CMDPFX(!QM02) QSGNAME(QSG1) RELTYPE(LTSR)
QMNAME(QM03) STATUS(Stopped) INSTVER(9.2.0) ERLYVER(9.2.0) CMDPFX(!QM03) QSGNAME(Unknown) RELTYPE(CDR)
QMNAME(QM04) STATUS(Running) INSTVER(9.1.0) ERLYVER(9.2.0) CMDPFX(!QM04) QSGNAME() RELTYPE(LTSR)
```

### 3. Input:

```
dspmqr -o all -su
```

#### Output:

```
QMNAME(QM02) STATUS(Running) INSTVER(9.2.0) ERLYVER(9.2.0) CMDPFX(!QM02) QSGNAME(QSG1) RELTYPE(LTSR)
```

```
QMNAME(QM03) STATUS(Stopped) INSTVER(9.2.0) ERLYVER(9.2.0) CMDPFX(!QM03) QSGNAME(Unknown) RELTYPE(CDR)
QMNAME(QM04) STATUS(Running) INSTVER(9.1.0) ERLYVER(9.2.0) CMDPFX(!QM04) QSGNAME() RELTYPE(LTSR)
```

**Related reference**

[“dspmq \(display queue managers\)” on page 73](#)

Display information about queue managers on Multiplatforms.

## Notices

---

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
19-21, Nihonbashi-Hakozakicho, Chuo-ku  
Tokyo 103-8510, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Software Interoperability Coordinator, Department 49XA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Programming interface information

---

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

**Important:** Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

## Trademarks

---

IBM, the IBM logo, [ibm.com](http://ibm.com)<sup>®</sup>, are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml). Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.



UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

This product includes software developed by the Eclipse Project (<https://www.eclipse.org/>).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.







Part Number:

(1P) P/N: