8.0

IBM MQ Administration Reference

Note

Before using this information and the product it supports, read the information in “Notices” on page 2031.

This edition applies to version 8 release 0 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, 2025.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Administration reference..... 7
Syntax diagrams.....7
How to read railroad diagrams.....7
IBM MQ Control commands.....8
Using control commands.....9
The control commands..... 10
Comparing command sets..... 162
Managing keys and certificates.....168
IBM MQ for IBM i CL commands..... 187
Add Queue Manager Information (ADDMQMINF).....191
Add Queue Manager Journal (ADDMQMJRN)..... 192
Connect MQ (CCTMQM)..... 194
Change Message Queue Manager (CHGMQM)..... 194
Change MQ AuthInfo object (CHGMQMAUTI)..... 216
Change MQ Channel (CHGMQMCHL)..... 223
Change Queue Manager Journal (CHGMQMJRN)..... 248
Change MQ Listener (CHGMQMLSR)..... 250
Change MQ Namelist (CHGMQMNL)..... 252
Change MQ Process (CHGMQMPRC)..... 253
Change MQ Queue (CHGMQMQ).....256
Change MQ Subscription (CHGMQMSUB)..... 274
Change MQ Service (CHGMQMSVC)..... 280
Change MQ Topic (CHGMQMTOP)..... 283
Clear MQ Pub/Sub Broker (CLRMQMBRK).....289
Clear MQ Queue (CLRMQMQ)..... 289
Clear MQ Topic String (CLRMQMTOP).....290
Copy MQ AuthInfo object (CPYMQMAUTI).....291
Copy MQ Channel (CPYMQMCHL).....299
Copy MQ Listener (CPYMQMLSR)..... 323
Copy MQ Namelist (CPYMQMNL).....326
Copy MQ Process (CPYMQMPRC)..... 328
Copy MQ Queue (CPYMQMQ)..... 331
Copy MQ Subscription (CPYMQMSUB)..... 349
Copy MQ Service (CPYMQMSVC)..... 356
Copy MQ Topic (CPYMQMTOP)..... 359
Create Message Queue Manager (CRTMQM).....365
Create MQ AuthInfo object (CRTMQMAUTI)..... 370
Create MQ Channel (CRTMQMCHL)..... 377
Create MQ Listener (CRTMQMLSR).....403
Create MQ Namelist (CRTMQMNL)..... 405
Create MQ Process (CRTMQMPRC).....406
Create MQ Queue (CRTMQMQ).....410
Create MQ Subscription (CRTMQMSUB).....427
Create MQ Service (CRTMQMSVC).....433
Create MQ Topic (CRTMQMTOP).....437
Convert MQ Data Type (CVTMQMDTA)..... 443
Delete Message Queue Manager (DLTMQM)..... 445
Delete MQ AuthInfo object (DLTMQMAUTI).....445
Delete MQ Pub/Sub Broker (DLTMQMBRK).....446
Delete MQ Channel (DLTMQMCHL).....447
Delete MQ Listener (DLTMQMLSR)..... 448
Delete MQ Namelist (DLTMQMNL).....449

Delete MQ Process (DLTMQMPRC)... 449
Delete MQ Queue (DLTMQMQ)...450
Delete MQ Subscription (DLTMQMSUB)... 451
Delete MQ Service (DLTMQMSVC)... 452
Delete MQ Topic (DLTMQMTOP)... 453
Dump MQ Configuration (DMPMQMCFG)... 454
Disconnect MQ (DSCMQM)... 459
Display Message Queue Manager (DSPMQM)... 459
Display MQ Object Authority (DSPMQMAUT)... 460
Display MQ AuthInfo object (DSPMQMAUTI)...462
Display MQ Pub/Sub Broker (DSPMQMBRK)...463
Display MQ Channel (DSPMQMCHL)...464
Display MQ Command Server (DSPMQMCSVR)... 465
Display MQ Listener (DSPMQMLSR)... 466
Display MQ Namelist (DSPMQMNL)...467
Display MQ Object Names (DSPMQMOBJN)... 468
Display MQ Process (DSPMQMPRC)... 471
Display MQ Queue (DSPMQMQ)...472
Display MQ Route Information (DSPMQMRTE)... 473
Display Queue Manager Status (DSPMQMSTS)...479
Display MQ Service (DSPMQMSVC)... 480
Display MQM Security Policy (DSPMQMSPL)...481
Display MQ Subscription (DSPMQMSUB)... 482
Display MQ Topic (DSPMQMTOP)... 483
Display MQ Version (DSPMQMVER)...484
End Message Queue Manager (ENDMQM)... 485
End MQ Pub/Sub Broker (ENDMQMBRK)... 488
End MQ Channel (ENDMQMCHL)... 489
End Queue Manager Connection (ENDMQMCONN)... 490
End MQ Command Server (ENDMQMCSVR)...491
End MQ Listeners (ENDMQMLSR)...492
End MQ Service (ENDMQMSVC)...493
Grant MQ Object Authority (GRTMQMAUT)...494
Ping MQ Channel (PNGMQMCHL)... 499
Record MQ Object Image (RCDMQMIMG)... 501
Re-create MQ Object (RCRMQMOBJ)... 503
Refresh IBM MQ Authority (RFRMQMAUT)... 505
Refresh MQ Cluster (RFRMQMCL)... 506
Refresh Message Queue Manager (RFRMQM)...507
Remove Queue Manager Info. (RMVMQMINF)...509
Remove Queue Manager Journal (RMVMQMJRN)... 509
Resume Cluster Queue Manager (RSMMQMCLQM)... 510
Reset MQ Channel (RSTMQMCHL)...511
Reset Cluster (RSTMQMCL)... 512
Resolve MQ Channel (RSVMQMCHL)... 514
RUNMQSC (RUNMQSC)... 515
Revoke MQ Object Authority (RVKMQMAUT)... 516
Set MQM Security Policy (SETMQMSPL)...521
Suspend Cluster Queue Manager (SPDMQMCLQM)... 524
Start Message Queue Manager (STRMQM)... 525
Start MQ Pub/Sub Broker (STRMQMBRK)... 527
Start MQ Channel (STRMQMCHL)... 528
Start MQ Channel Initiator (STRMQMCHLI)... 529
Start MQ Command Server (STRMQMCSVR)... 529
Start IBM MQ DLQ Handler (STRMQMDLQ)... 530
Start MQ Listener (STRMQMLSR)... 532
Start IBM MQ Commands (STRMQMMQSC)... 533
Start MQ Service (STRMQMSVC)... 535

Start MQ Trigger Monitor (STRMQMTRM)....536
Trace MQ (TRCMQM)....537
Work with MQ Queue Manager (WRKMQM)....543
Work with MQ Authority (WRKMQMAUT)....544
Work with MQ Authority Data (WRKMQMAUTD)....546
Work with AuthInfo objects (WRKMQMAUTI)....548
Work with MQ Channels (WRKMQMCHL)....551
Work with MQ Channel Status (WRKMQMCHST)....561
Work with MQ Clusters (WRKMQMCL)....567
Work with MQ Cluster Queues (WRKMQMCLQ)....577
Work with MQ Connections (WRKMQMCONN)....581
Work Queue Manager Journals (WRKMQMJRN)....585
Work with MQ Listeners (WRKMQMLSR)....586
Work with MQ Messages (WRKMQMMSG)....589
Work with MQ Namelist (WRKMQMNL)....591
Work with MQ Processes (WRKMQMPRC)....593
Work with MQ Queues (WRKMQMQ)....597
Work with Queue Status (WRKMQMQSTS)....609
Work with MQM Security Policies (WRKMQMSPL)....612
Work with MQ Subscriptions (WRKMQMSUB)....613
Work with MQ Service object (WRKMQMSVC)....618
Work with MQ Topics (WRKMQMTOP)....621
Work with MQ Transactions (WRKMQMTRN)....626
MQSC reference....627
Generic values and characters with special meanings....627
Building command scripts....628
Using commands on z/OS....629
The MQSC commands....630
Programmable command formats reference....1327
Definitions of the Programmable Command Formats....1327
Structures for commands and responses....1836
PCF example....1862
Using the IBM MQ utilities for z/OS....1872
An overview of the IBM MQ utilities....1873
Syntax diagrams....1876
IBM MQ utility program (CSQUTIL)....1876
The change log inventory utility (CSQJU003)....1910
The print log map utility (CSQJU004)....1918
The log print utility (CSQ1LOGP)....1919
The queue-sharing group utility (CSQ5PQSG)....1929
The active log preformat utility (CSQJUFMT)....1933
The dead-letter queue handler utility (CSQUDLQH)....1934
The BSDS conversion utility (CSQJUCNV)....1943
The message security policy utility (CSQ0UTIL)....1945
IBM MQ Administration Interface....1949
MQAI calls....1949
MQAI Selectors....2028
Example code....2029

Notices....2031
Programming interface information....2032
Trademarks....2032

Administration reference

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

- “Syntax diagrams” on page 7
- “IBM MQ Control commands” on page 8
- IBM i “IBM MQ for IBM i CL commands” on page 187
- “MQSC reference” on page 627
- “Programmable command formats reference” on page 1327
- z/OS “Using the IBM MQ utilities for z/OS” on page 1872
- “IBM MQ Administration Interface” on page 1949

Related reference

Queue names

IBM i IBM MQ for IBM i system and default objects

Syntax diagrams

The syntax for a command and its options is presented in the form of a syntax diagram called a railroad diagram.

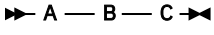
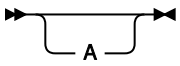
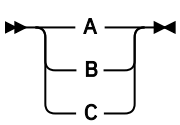
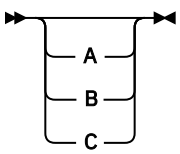
Railroad diagrams are a visual format suitable for sighted users; see, “How to read railroad diagrams” on page 7. It tells you what options you can supply with the command, how to enter them, indicates relationships between different options, and sometimes different values of an option.

How to read railroad diagrams

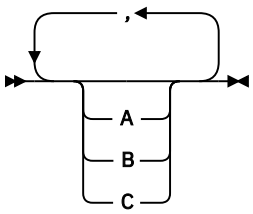
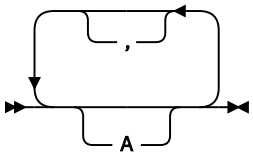
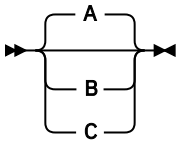
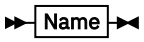
Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

Other conventions used in railroad diagrams are:

Table 1. How to read railroad diagrams

Convention	Meaning
	You must specify values A, B, and C. Required values are shown on the main line of a railroad diagram.
	You may specify value A. Optional values are shown below the main line of a railroad 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.

© Copyright IBM Corp. 2007, 2025

Table 1. How to read railroad diagrams (continued)	
Convention	**Meaning**
	You might specify one or more of the values A, B, and C. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow.
	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.
 **Name** 	The railroad fragment Name is shown separately from the main railroad diagram.
Punctuation and uppercase values	Specify exactly as shown.

IBM MQ Control commands

Find out how to use the IBM MQ control commands.

If you want to issue control commands, your user ID must be a member of the mqm group. For more information, see Authority to administer IBM MQ on UNIX, Linux®, and Windows systems.

When using control commands that operate on a queue manager configured to use connection authentication with CHCKLOCL(REQUIRED), and a failure to connect is observed, either

- Supply a user id and password if the control command allows this.
- Use MQSC equivalents of the control commands where those exist.
- Start the queue manager using the -ns option, while control commands that cannot connect need to be run.

When using control commands that operate on a queue manager, you must use the command from the installation associated with the queue manager you are working with.

In addition, note the following environment-specific information:

- On Windows, all control commands can be issued from a command line. Command names and their flags 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) are case-sensitive.

In the syntax descriptions, the hyphen (-) is used as a flag indicator. You can use the forward slash (/) instead of the hyphen.

- On UNIX and Linux systems, all IBM MQ control commands can be issued from a shell. All commands are case-sensitive.
- A subset of the control commands can be issued using the IBM MQ Explorer.

For a list of the control commands see, "The control commands" on page 10.

For a comparison of the different administration command sets, see "Comparing command sets" on page 162.

For information about commands for managing keys and certificates, see "Managing keys and certificates" on page 168.

Related concepts

"MQSC reference" on page 627
Use MQSC commands to manage queue manager objects, including the queue manager itself, queues, process definitions, channels, client connection channels, listeners, services, namelists, clusters, and authentication information objects.

"Programmable command formats reference" on page 1327
Programmable Command Formats (PCFs) define command and reply messages that can be exchanged between a program and any queue manager (that supports PCFs) in a network. PCFs simplify queue manager administration and other network administration.

Related reference

"IBM MQ for IBM i CL commands" on page 187
A list of IBM MQ for IBM i CL commands grouped according to command type:

Using control commands

The table in this topic shows the three categories of control commands: queue manager commands, channel commands, and utility commands.

Control commands can be divided into three categories, as shown in Table 2 on page 9.

Table 2. Categories of control commands

Category	Description
Queue manager commands	Queue manager control commands include commands for creating, starting, stopping, and deleting queue managers and command servers
Channel commands	Channel commands include commands for starting and ending channels and channel initiators
Utility commands	Utility commands include commands associated with: • Running MQSC commands • Conversion exits • Authority management • Recording and recovering media images of queue manager resources • Displaying and resolving transactions • Trigger monitors • Displaying the file names of IBM MQ objects

For more information, see "IBM MQ Control commands" on page 8

Using control commands on Windows systems

In IBM MQ for Windows, you enter control commands at a command prompt.

In Windows environments, control commands and their flags are not case-sensitive, but arguments to those commands (such as queue names and queue-manager names) are case-sensitive.

For example, in the command:

```
crtmqm /u SYSTEM.DEAD.LETTER.QUEUE jupiter.queue.manager
```

- The command name can be entered in uppercase or lowercase, or a mixture of the two. These are all valid: `crtmqm`, `CRTMQM`, and `CRTmqm`.
- The flag can be entered as `-u`, `-U`, `/u`, or `/U`.
- `SYSTEM.DEAD.LETTER.QUEUE` and `jupiter.queue.manager` must be entered exactly as shown.

For more information, see IBM MQ control commands.

Using control commands on UNIX and Linux systems

In IBM MQ for UNIX and Linux systems, you enter control commands in a shell window.

In UNIX environments, control commands, including the command name itself, the flags, and any arguments, are case-sensitive. For example, in the command:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE jupiter.queue.manager
```

- The command name must be `crtmqm`, not `CRTMQM`.
- The flag must be `-u`, not `-U`.
- The dead-letter queue is called `SYSTEM.DEAD.LETTER.QUEUE`.
- The argument is specified as `jupiter.queue.manager`, which is different from `JUPITER.queue.manager`.

Take care to type the commands exactly as you see them in the examples.

For more information about the `crtmqm` command, see “crtmqm” on page 25.

For more information on control commands, see “IBM MQ Control commands” on page 8

The control commands

This collection of topics provides reference information for each of the IBM MQ control commands.

For information about running these commands refer to “IBM MQ Control commands” on page 8.

addmqinf

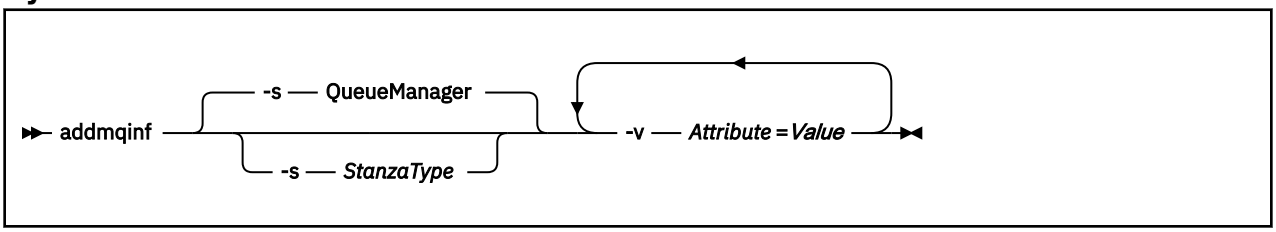
Add IBM MQ configuration information (Windows and UNIX platforms only).

Purpose

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

For example, use **dspmqinf** 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 3 on page 11 lists the QueueManager stanza attribute values. The queue manager stanza is the only stanza that is currently supported.

Table 3. QueueManager stanza attributes

Attribute	***Value***	**Required or optional**
Name	The name of the queue manager. You must provide a different name from any other queue manager stanza on the system.	Required
Prefix	The directory path *under* which this queue manager data directory is stored by default. You can use **Prefix** to modify the location of the queue manager data directories. The value of **Directory** *is* automatically appended to this path.	Required
Directory	The name of the queue manager data directory. Sometimes the name must be provided (as in "Example" on page 12), because it is different from the queue manager name. Copy the directory name from the value returned by **dspmqinf**. The rules for transforming queue manager names into directory names are described in Understanding IBM MQ file names.	Required
DataPath	The directory path where the queue manager data files are placed. The value of **Directory** is *not* automatically appended to this path - you must provide the transformed queue manager name as part of **DataPath**. If the **DataPath** attribute is omitted on UNIX, the queue manager data directory path is defined as ***Prefix*** / ***Directory*** .	UNIX: Optional Windows: Required

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

Return code	Description
0	Successful operation
1	Queue manager location is invalid (either **Prefix** or **DataPath**)
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

Command	Description
“dspmqinf” on page 62	Display IBM MQ configuration information
“rmvmqinf” on page 100	Remove IBM MQ configuration information

amqmdain

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

Purpose

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

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

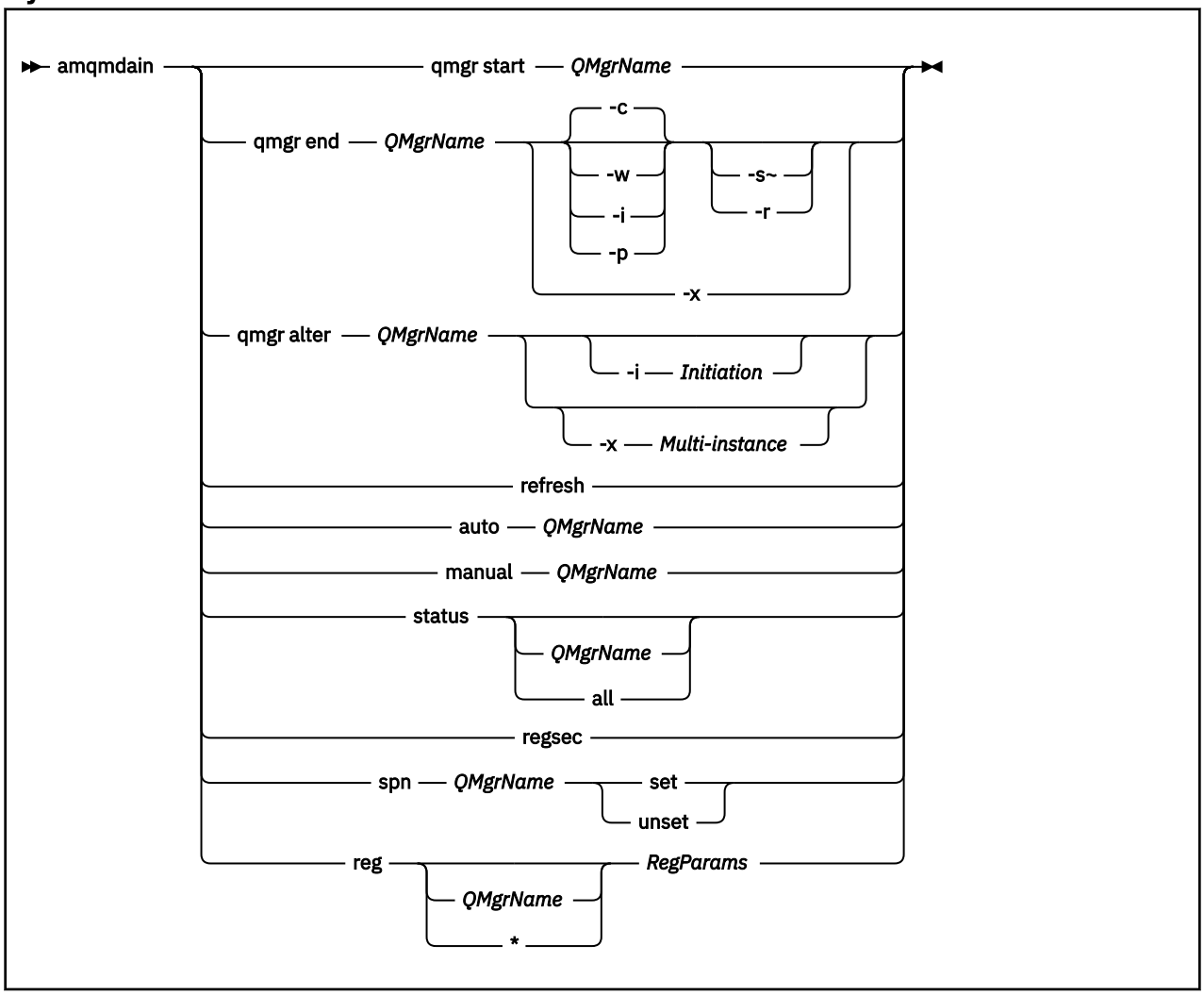
account. However, to ensure that all queue manager startup feedback is returned to the command line, use the `strmqm -ss` command rather than **amqmdain**.

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 -o installation` command.

To administer and define IBM MQ service and listener objects, use MQSC commands, PCF commands, or the 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" on page 84.

-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:

Table 4. Initiation command parameters.

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

-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:

Table 5. Multi-instance command parameters.	
Value	**Description**
set	Sets automatic queue manager startup to permit multiple instances. Issues `strmqm -x` . The `set` option is ignored for queue managers that are initiated interactively or as a manual service startup. The syntax of the command is: `amqmdain qmgr alter QmgrName -x set`
unset	Sets automatic queue manager startup to single instance. Issues `strmqm`. The `unset` option is ignored for queue managers that are initiated interactively or as a manual service startup. The syntax of the command is: `amqmdain qmgr alter QmgrName -x unset`

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 6. 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 *all* 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 7. Reg command parameters.	
Value	**Description**
If *RegParams* is specified alone:	Modifies **queue manager** configuration information related to the default queue manager.

Table 7. Reg command parameters. (continued)	
Value	**Description**
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
```

The following are 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=closeit
```

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, 2025. 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

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
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” on page 150.
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” on page 84.

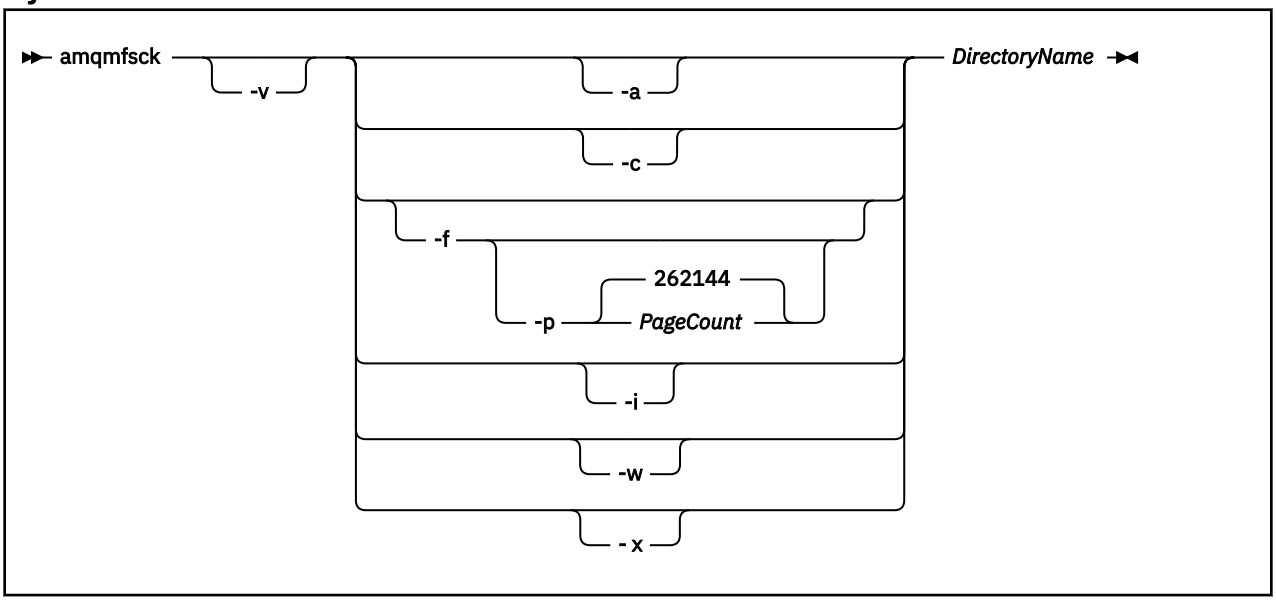
amqmfsck (file system check)

amqmfsck checks whether a shared file system on UNIX and IBM i systems meets the requirements for storing the queue manager data of a multi-instance queue manager.

Purpose

The **amqmfsck** command applies only to UNIX and IBM i systems. You do not need to check the network drive on Windows. **amqmfsck** 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 **amqmfsck** 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 **amqmfsck** 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 and support statement for IBM MQ multi-instance queue managers, 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

crtmqcvx

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

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

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`.

crtmqenv

Create a list of environment variables for an installation of IBM MQ, on UNIX, 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_JAVA_INSTALL_PATH
- MQ_JAVA_DATA_PATH
- MQ_JAVA_LIB_PATH
- MQ_JAVA_JVM_FLAG
- MQ_JRE_PATH

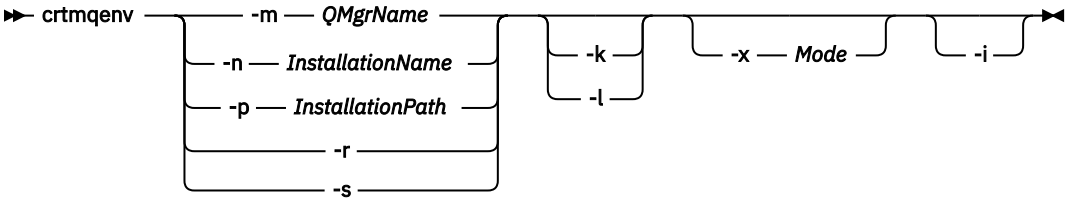
- PATH

On UNIX and Linux systems, if the **-l** or **-k** flag is specified, the *LIBPATH* environment variable is set on AIX®, and the *LD_LIBRARY_PATH* environment variable is set on HP-UX, Linux, and Solaris.

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

-k
UNIX and Linux only.

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.

-l
UNIX and Linux only.

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.

-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

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 UNIX or Linux 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\WebSphere MQ` on a Windows system.

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

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

Related concepts
Multiple installations
Related tasks
Choosing a primary installation
Related reference
“setmqenv” on page 136
Use the **setmqenv** command to set up the IBM MQ environment on UNIX, Linux, and Windows.

crtmqinst

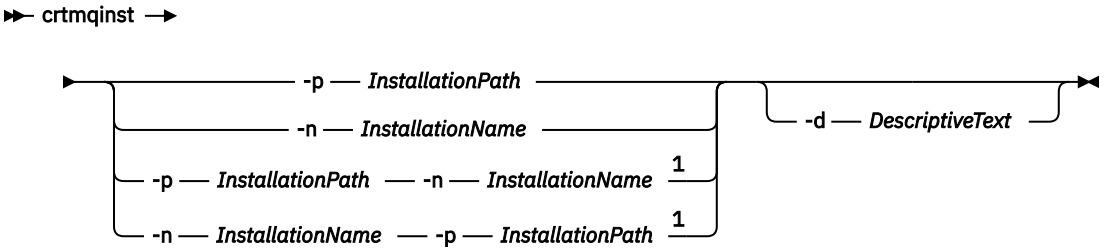
Create installation entries in `mqinst.ini` on UNIX and Linux systems.

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.

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



Notes:

[1] 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 UNIX and Linux systems, and `/usr/mqm` is used on AIX.

Return codes

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 UNIX systems, 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

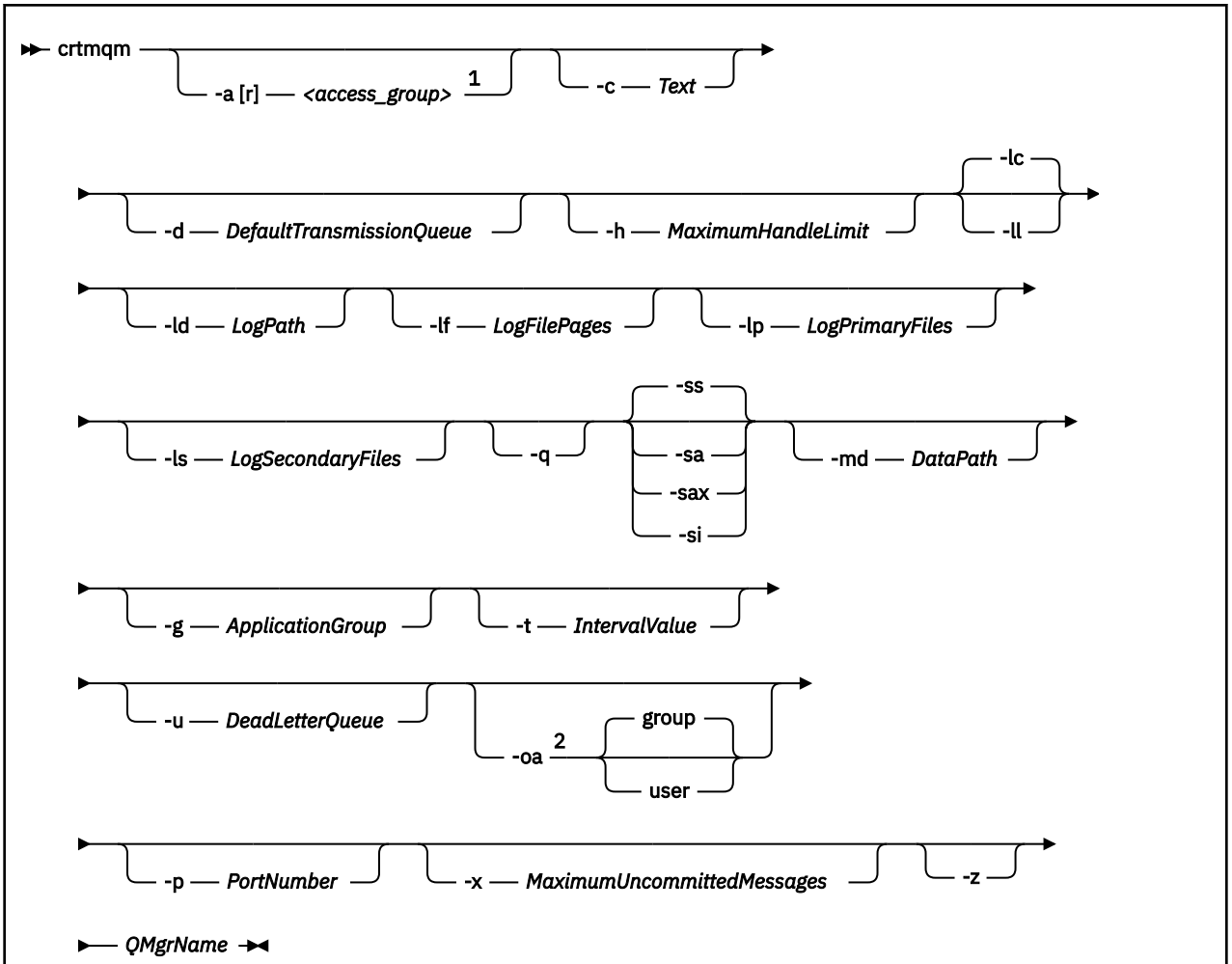
Create a queue manager.

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. 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 UNIX, Linux and Windows systems.

Syntax



Notes:

[1] Windows only
[2] UNIX and Linux only

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[r]** ***access_group***

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 [r]` 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.

-g ***ApplicationGroup***

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

This option applies to IBM MQ for AIX, Solaris, HP-UX, and Linux.

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

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

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.

`-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 UNIX and Linux are different from those for the Windows system:

UNIX 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
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.

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.

In IBM MQ for UNIX and 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.

-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.

In IBM MQ for UNIX and 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.

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.

-ll ***LinearLogging***
Use linear logging.

-lp ***LogPrimaryFiles***
The log files allocated when the queue manager is created.

On a Windows system, the minimum number of primary log files you can have is 2 and the maximum is 254. On UNIX and Linux systems, the minimum number of primary log files you can have is 2 and the maximum is 510. The default is 3.

On a Windows system, the total number of primary and secondary log files must not exceed 255 and must not be less than 3. On UNIX and Linux systems 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.

On a Windows system, the minimum number of secondary log files you can have is 1 and the maximum is 253. On UNIX and Linux systems, the minimum number of secondary log files you can have is 2 and the maximum is 509. The default is 2.

On a Windows system, the total number of primary and secondary log files must not exceed 255 and must not be less than 3. On UNIX and Linux systems 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.

-md ***DataPath***
The directory used to hold the data files for a queue manager.

In IBM MQ for Windows systems, the default is `C:\ProgramData\IBM\MQ\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.

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

The DataPath parameter is provided to assist in the configuration of multi-instance queue managers. For example, on UNIX 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 IBM MQ earlier than Version 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-Version 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 MQ to the pre-v7.0.1 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

On UNIX 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 format).

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.

-sa

Automatic queue manager startup. For Windows systems only.

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 MQ Explorer.

Queue managers created in IBM MQ releases earlier than Version 7 retain their existing startup type.

-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.

Queue managers created in IBM MQ versions earlier than Version 7.0.1 retain their existing startup type.

-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.

-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

Return code	Description
0	Queue manager created
8	Queue manager exists
39	Invalid parameter specified
49	Queue manager stopping
58	Inconsistent use of installations detected
69	Storage unavailable
70	Queue space unavailable
71	Unexpected error
72	Queue manager name error
74	The IBM MQ service is not started.
100	Log location invalid
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)

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 UNIX 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 `rwxrwxr-x`.

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

Related commands

Command	Description
“strmqm” on page 150	Start queue manager
“endmqm” on page 84	End queue manager
“dltmqm” on page 34	Delete queue manager
“setmqm” on page 140	Set associated installation

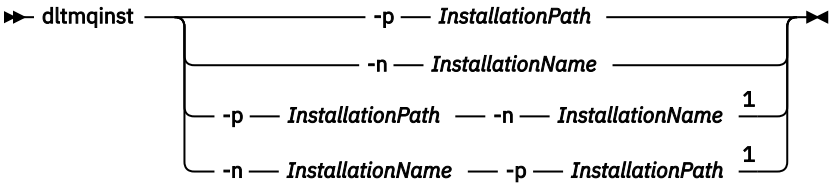
dltmqinst

Delete installation entries from `mqinst.ini` on UNIX 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.

Syntax



Notes:

1 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

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.

Note: On a Solaris 10 MQ Client installation, only the root user has permissions to edit the mqinst.ini file.

dltmqm

Delete a queue manager.

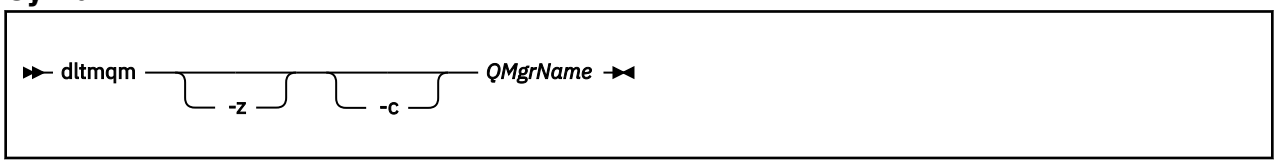
Purpose

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 `dspmq -o installation` command.

In IBM MQ for 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



Required parameters

QMgrName
The name of the queue manager to delete.

Optional parameters

-z
Suppresses error messages.

HPE NonStop V8.1 **-c**
Applies only to IBM MQ for HPE NonStop. If you recreate a queue manager with a name already known to **runnscnf**, the previous settings remain in effect. If you want to delete the settings completely, you must use **dltmqm** with the `-c` option. All settings made with **runnscnf** are persisted in the Guardian key sequenced file AMQPARMS. The file is global, not queue manager-specific, and contains all **runnscnf** settings of all queue managers. Settings remain in AMQPARMS after the queue manager is deleted.

Return codes

Return code	Description
0	Queue manager deleted
3	Queue manager being created

Return code	Description
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.
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
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

In IBM MQ for 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 commands

Command	Description
crtmqm	Create queue manager
strmqm	Start queue manager
endmqm	End queue manager

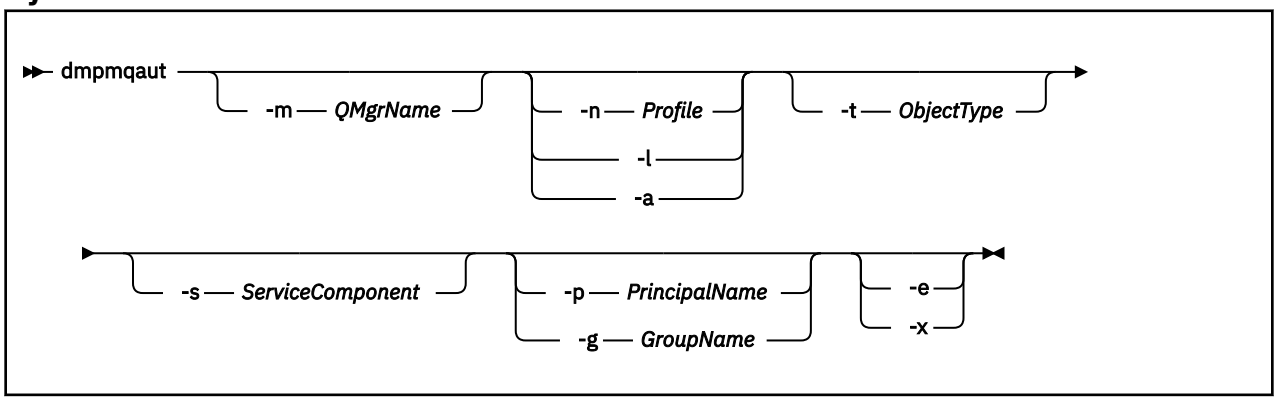
dmpmqaut

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 UNIX, 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
authinfo	An authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	A channel
clntconn or **clcn**	A client connection channel
listener or **lstr**	A listener
namelist or **nl**	A namelist
process or **prcs**	A process
queue or **q**	A queue or queues matching the object name parameter
qmgr	A queue manager
rqmname or **rqmn**	A remote queue manager name
service or **srvc**	A service
topic or **top**	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.

-p *PrincipalName*
This parameter applies to IBM MQ for Windows only; UNIX 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.

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: UNIX users cannot use the `-p` option; they 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 IBM MQ 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 IBM MQ 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.

dmpmqcfg

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

Purpose

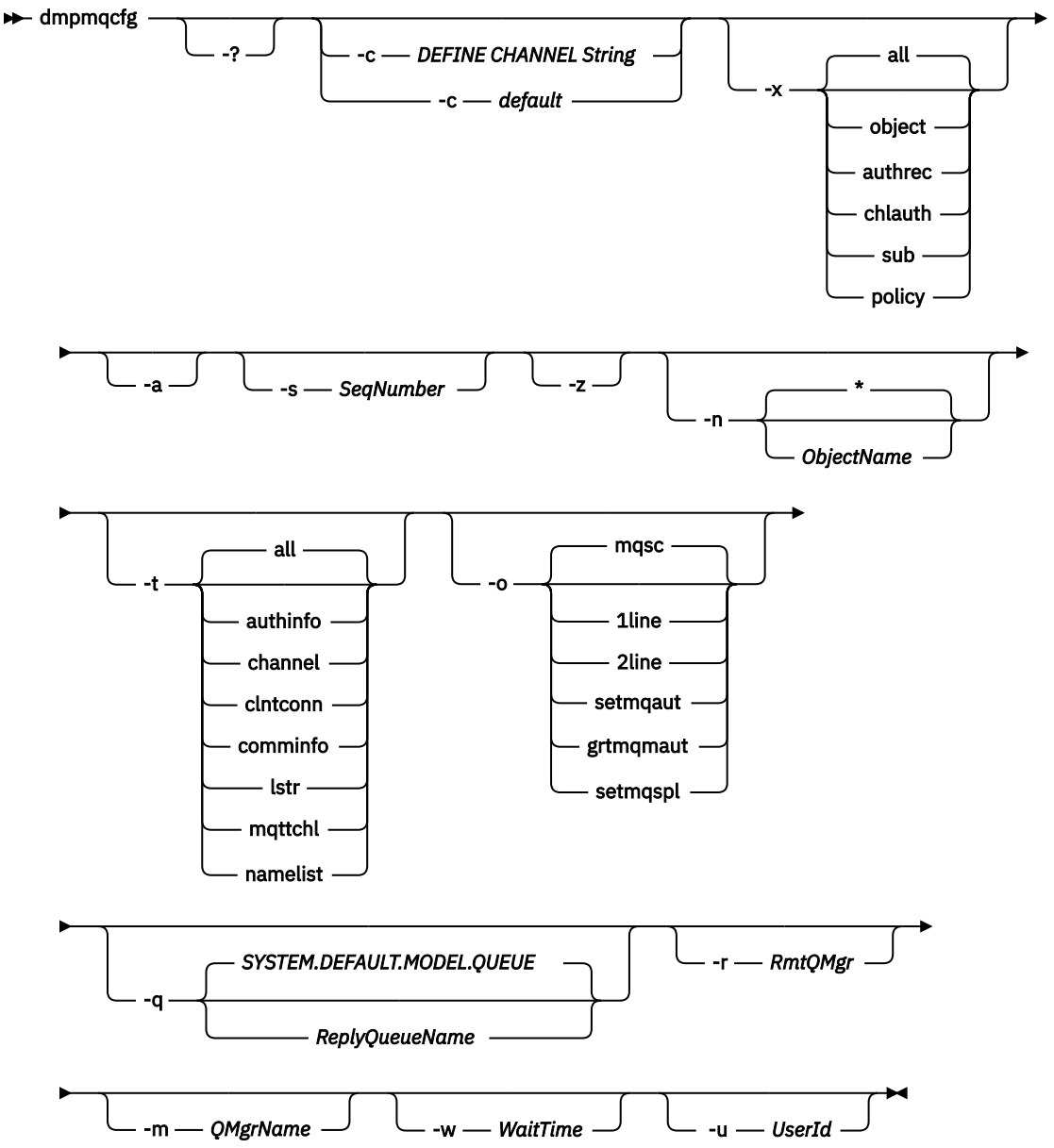
Use the dmpmqcfg command to dump the configuration of IBM MQ queue managers. If any default object has been edited, the -a option must be used if the dumped configuration will be used to restore the configuration.

The **dmpmqcfg** 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 **dmpmqcfg** 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 **dmpmqcfg**. 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 IBM MQ Telemetry.

Note: The **dmpmqcfg** command does not make a backup of the IBM MQ Advanced Message Security policies. If you want to export the IBM MQ Advanced Message Security policies, ensure that you run **dspmqspl** with the `-export` flag. This command exports the policies for IBM MQ Advanced Message Security into a text file, which can be used for restoration purposes. For more information see "dspmqspl" on page 73.

Attention: z/OS V 8.0.0.14 From Version 8.0.0, Fix Pack 14, The inquiries used by **dmpmqcfg**, the inquiries used by **dmpmqcfg** inquire only QSGDISP(QMGR) definitions by default. You can inquire additional definitions by using the environment variable **AMQ_DMPMQCFG_QSGDISP_DEFAULT**. For more information about the values that you can set with this environment variable, see **AMQ_DMPMQCFG_QSGDISP_DEFAULT**.

z/OS For example, you could use **AMQ_DMPMQCFG_QSGDISP_DEFAULT** to query a z/OS® queue manager in a queue sharing group from an IBM MQ for Distributed platforms 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 **dmpmqcfg** 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. Possible values are:

Value	Description
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
srvc or service	A service
topic or top	A topic

-o *[mqsc | 1line | 2line | setmqaut | grtmqmaut | setmqspl]*
Possible values are:

Value	Description
mqsc	Multi-line MQSC that can be used as direct input to **runmqsc**
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 UNIX and Windows queue managers; valid only when -x authrec is specified
grtmqmaut	Linux only; generates iSeries syntax for granting access to the objects.

Value	Description
`setmqspl`	The security policies for the queue manager are reported in the format of **setmqspl** command lines. This format can be used to generate scripts to restore policy configuration to a queue manager. Note that the **setmqspl** 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. If you need to backup 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.

-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.

V 8.0.0.8 **-w** ***WaitTime***
The time, in seconds, that **dmpmqcfg** 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

The user 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.

The user 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 **dmpmqcfg** command to display details about any OBJTYPE(RQMNAME).

Return code

If a failure occurs **dmpmqcfg** 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 Preparing queue managers for remote administration and Preparing channels and transmission queues for remote administration.

```
dmpmqcfg -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.

```
dmpmqcfg -m LOCALQM -r 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

Restoring queue manager configuration

dmpmqlog

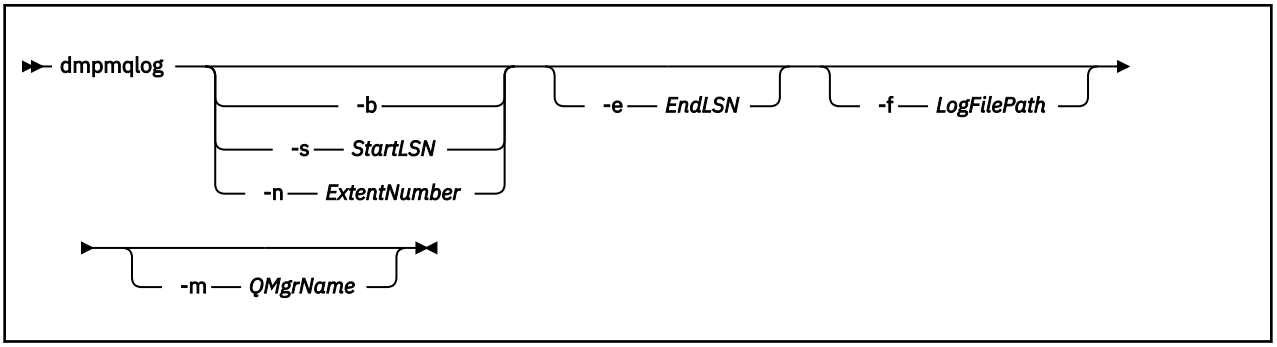
Display and format a portion of the IBM MQ system log.

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.

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 (`amqhlctl.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.

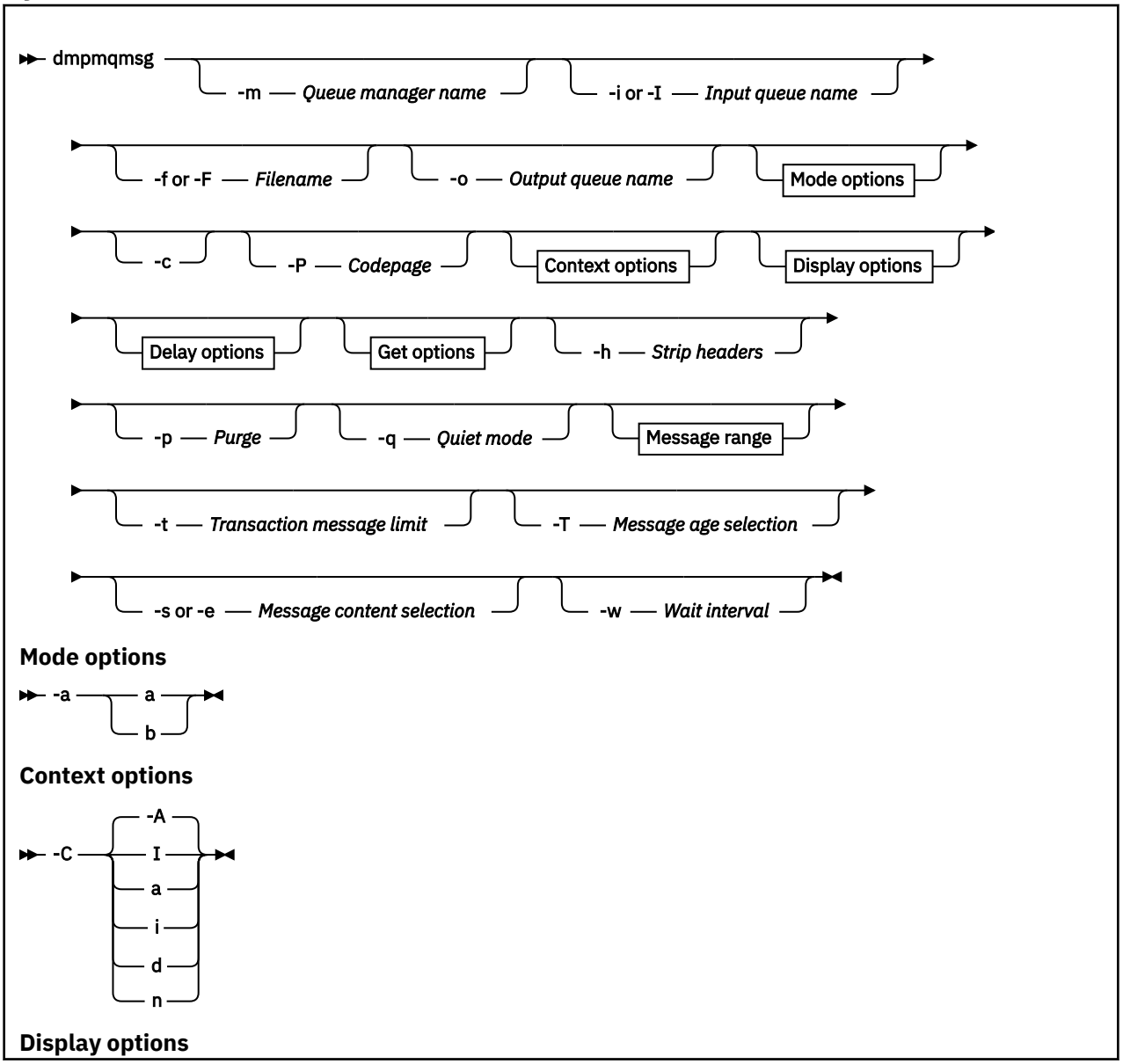
dmpmqmsg

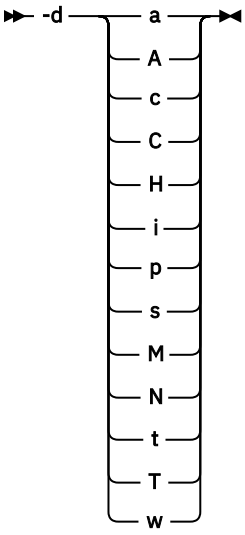
Formerly the IBM MQ queue load and unload utility.

Purpose

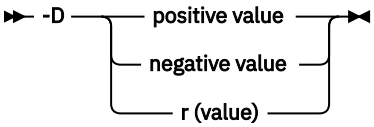
Use the **dmpmqmsg** utility to copy or move the contents of a queue, or its messages, to a file.

Syntax

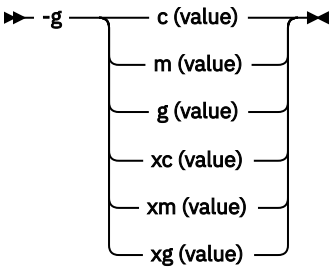




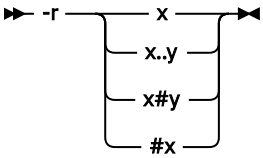
Delay options



Get options



Message range options



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.

-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 codepage safe. Loading a file written in this format, while running in a new codepage 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 codepage safe. Loading a file written in this format, while running in a new codepage 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.

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

<negative 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.

r <value>
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.

c<value>
Get by character Correlation identifier.

m<value>
Get by character Message identifier.

g<value>
Get by character Group identifier.

xc<value>
Get by hexadecimal Correlation identifier.

xm<value>
Get by hexadecimal Message identifier.

xg<value>
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 `-t 1000` deals with 1000 messages in a single transaction.

-T
Allows message selection based on message age.

See "Using message age" on page 51 for information on selection using message age.

-s or -e
Allows message selection based on message content.

On ASCII platforms (Windows and UNIX and Linux) 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 51 for information on selection using message content.

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

See Examples of using the **dmpmqmsg** utility for examples on using the utility.

Message selection

The following information describes how message selection occurs using message age and 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.

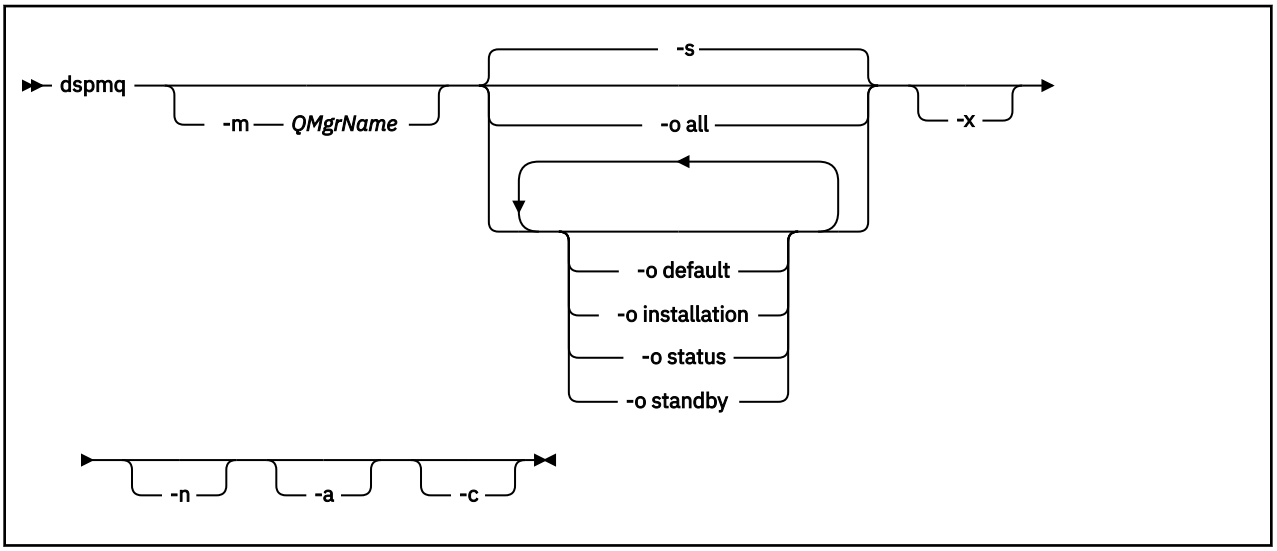
dspmq

Display information about queue managers.

Purpose

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

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 **dspmq** 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.

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

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

-o installation
Windows, UNIX and Linux 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 8 on page 53.

Table 8. Standby values

Value	Description
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.

-x
Information about queue manager instances are displayed. The possible values are shown in Table 9 on page 53.

Table 9. Instance values

Value	Description
Active	The instance is the active instance.
Standby	The instance is a standby instance.

-c
Shows the list of processes currently connected to the IPCC, QMGR, and PERSISTENT sub pools 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 following is a list of the different states a queue manager can be in:

Table 10. Queue manager states.
Queue Manager States
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 11. Return code identifiers and descriptions

Return code	Description
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:

```
dspmq -o standby
```

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

```
dspmq -o standby -x
```

dspmqaut

dspmqaut displays the authorizations of a specific IBM MQ object.

Purpose

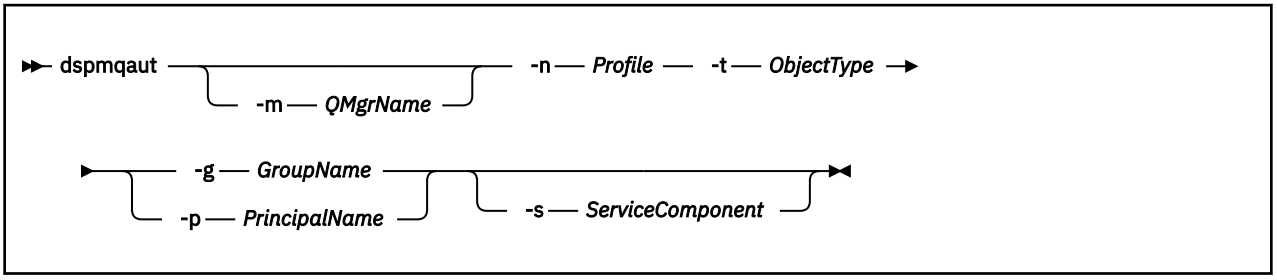
Use the `dspmqaut` command to display the current authorizations to a specified object.

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:

Table 12. The object type on which to make the inquiry.

Object Type	Description
authinfo	An authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	A channel
clntconn or **clcn**	A client connection channel
listener or **lstr**	A Listener
namelist or **nl**	A namelist
process or **prcs**	A process
queue or **q**	A queue or queues matching the object name parameter
qmgr	A queue manager
rqmname or **rqmn**	A remote queue manager name
service or **srvc**	A service
topic or **top**	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.

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.

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 13 on page 56 shows the authorities that can be given to the different object types.

Table 13. Specifying authorities for different object types

Authority	Queue	Process	Queue manager	Remote queue manager name	Namelist	Topic	Auth info	Clntconn	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

Table 13. Specifying authorities for different object types (continued)

Authority	Queue	Process	Queue manager	Remote queue manager name	Namelist	Topic	Auth info	Clntconn	Channel	Listener	Service
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 14. 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
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

Table 14. Authorizations associated with each value. (continued)	
Value	**Description**
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

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

Return code	Description
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`:

```
dspmqaut -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:

```
dspmqaut -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
```

dspmqcsv

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

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

Command	Description
strmqcsv	Start a command server
endmqcsv	End a command server

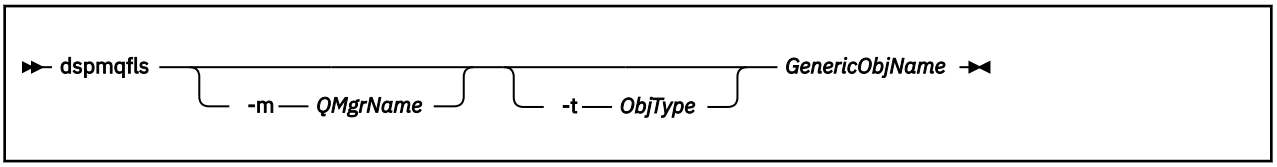
dspmqfls

Display the file names corresponding to IBM MQ objects.

Purpose

Use the `dspmqfls` 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 15. Valid object types.

Object Type	Description
***** or **all**	All object types; this parameter is the default
authinfo	Authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	A channel
clntconn or **clcn**	A client connection channel
catalog or **ctlg**	An object catalog
namelist or **nl**	A namelist
listener or **lstr**	A listener
process or **prcs**	A process
queue or **q**	A queue or queues matching the object name parameter

Table 15. Valid object types. (continued)	
Object Type	**Description**
qalias or **qa**	An alias queue
qlocal or **ql**	A local queue
qmodel or **qm**	A model queue
qremote or **qr**	A remote queue
qmgr	A queue manager object
service or **srvc**	A service

Note:

1. The `dspmqfls` command displays the name of the directory containing the queue, ***not*** the name of the queue itself.
2. In IBM MQ for UNIX systems, 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

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.

```
dspmqfls SYSTEM.ADMIN*
```

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

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

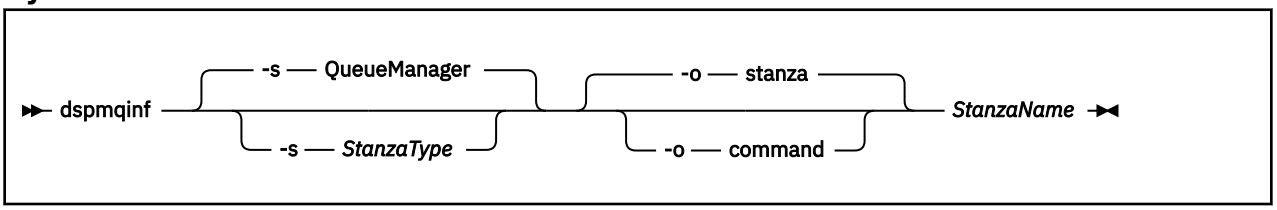
dspmqinf

Display IBM MQ configuration information (Windows and UNIX platforms only).

Purpose

Use the `dspmqinf` 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

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

Command	Description
“addmqinf” on page 10	Add queue manager configuration information
“rmvmqinf” on page 100	Remove queue manager configuration information

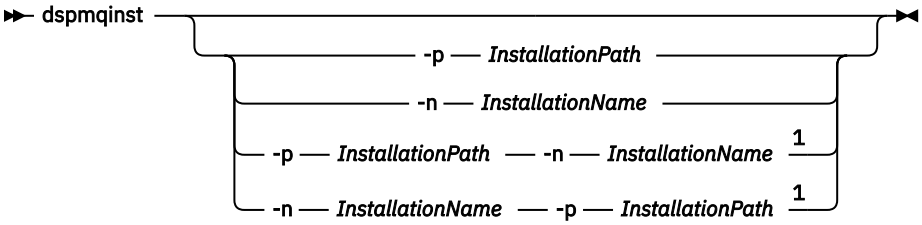
dspmqinst

Display installation entries from `mqinst.ini` on UNIX, Linux, and Windows.

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.

Syntax



Notes:

[1] When specified together, the installation name and installation path must refer to the same installation.

Required parameters

None

Optional parameters

-n *InstallationName*
The name of the installation.

-p *InstallationPath*
The installation path.

?
Display usage information.

Return codes

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

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
```

dspmqrte

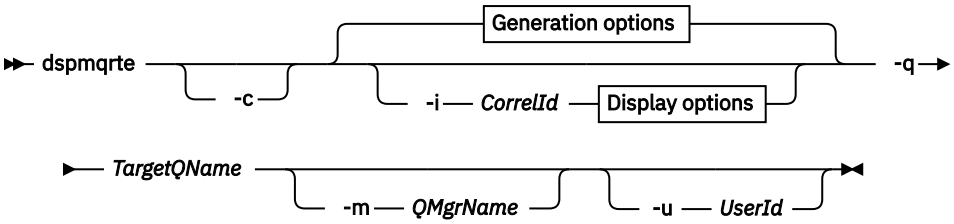
Determine the route that a message has taken through a queue manager network.

Purpose

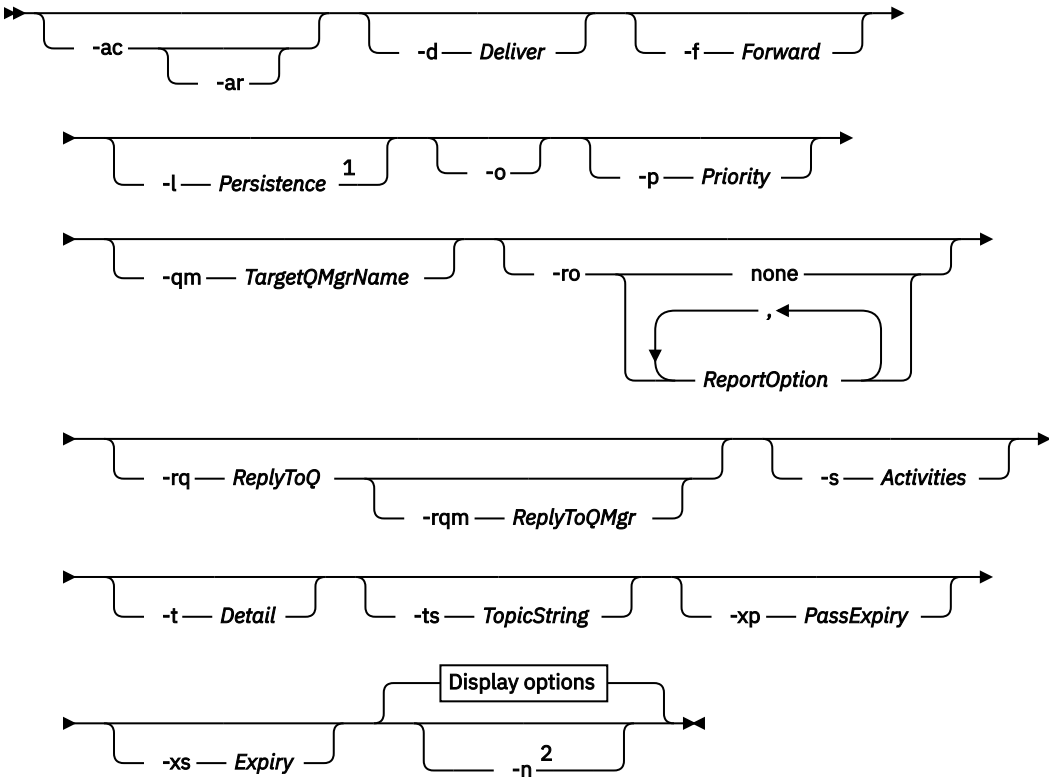
The IBM MQ display route application (dspmqrte) can be run on all platforms except z/OS. You can run the IBM MQ display route application as a client to an IBM MQ for z/OS queue manager by specifying the -c parameter when issuing the dspmqrte command.

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.

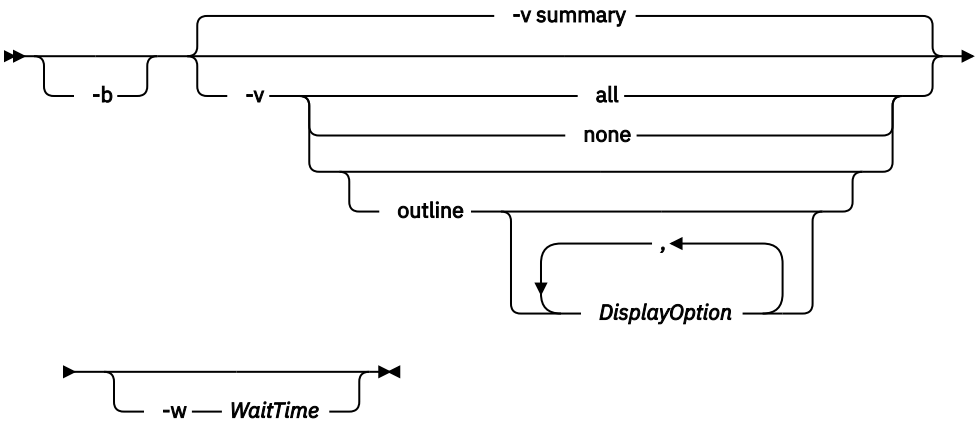
Syntax



Generation options



Display options



Notes:

[1] 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.
[2] 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 Installing an IBM MQ client.

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 in the following circumstances:

- The trace-route message is discarded by a IBM WebSphere® MQ 7.0 queue manager.
- The trace-route message is put to a local queue (target queue or dead-letter queue) by a IBM WebSphere MQ 7.0 queue manager.
- 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:

Table 16. Delivery parameter values.

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:

Table 17. Forward parameter values.

Value	Description
all	The trace-route message is forwarded to any queue manager. **Warning:** If forwarded to an IBM MQ queue manager before Version 6.0, the trace-route message is not recognized and can be delivered to a local queue despite the value of the *-d Deliver* parameter.
supported	The trace-route message is only forwarded to a queue manager that honors the *Deliver* parameter from the *TraceRoute* 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:

Table 18. Persistence parameter values.

Value	Description
yes	The generated trace-route message is persistent. (MQPER_PERSISTENT).
no	The generated trace-route message is not persistent. (MQPER_NOT_PERSISTENT).
q	The generated trace-route message inherits its persistence value from the queue specified by *-q TargetQName*. (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 *-rq 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 *TargetQMgrName*

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 19. ReportOption parameter values.

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

If *-ro ReportOption* or *-ro 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:

Table 20. Detail parameter values.

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 21. PassExpiry parameter values.	
Value	**Description**
yes	The report option MQRO_PASS_DISCARD_AND_EXPIRY is specified in the message descriptor of the trace-route message. 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. This parameter is the default value.
no	The report option MQRO_PASS_DISCARD_AND_EXPIRY is **not** specified. 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.

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 (*-ar*), or any of the report generating options from (*-ro ReportOption*), then a specific (non-model) reply-to queue must be specified using *-rq 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 22. 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.
outline *DisplayOption*	Specifies display options for the trace-route message. Multiple display options can be specified using a comma as a separator. If no values are supplied the subsequent information is displayed: • The application name • The type of each operation • Any operation-specific parameters Possible values for *DisplayOption* are: **activity** All non-PCF group parameters in *Activity* PCF groups are displayed. **identifiers** Values with parameter identifiers MQBACF_MSG_ID or MQBACF_CORREL_ID are displayed. This overrides *msgdelta*. **message** All non-PCF group parameters in *Message* PCF groups are displayed. When this value is specified, you cannot specify *msgdelta*. **msgdelta** All non-PCF group parameters in *Message* PCF groups, that have changed since the last operation, are displayed. When this value is specified, you cannot specify *message*. **operation** All non-PCF group parameters in *Operation* PCF groups are displayed. **traceroute** All non-PCF group parameters in *TraceRoute* PCF groups are displayed.

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

Return code	Description
0	Command completed normally
10	Invalid arguments supplied
20	An error occurred during processing

Examples

1. The following command puts a trace-route message into a queue manager network with the target queue specified as TARGET.Q. Providing queue managers on route are enabled for activity recording, activity reports are generated. Depending on the queue manager attribute, ACTIVREC, activity reports are either delivered to the reply-to queue ACT.REPORT.REPLY.Q, or are delivered to a system queue. The trace-route message is discarded on arrival at the target queue.

   ```
   dspmqrte -q TARGET.Q -rq ACT.REPORT.REPLY.Q
   ```

   Providing one or more activity reports are delivered to the reply-to queue, ACT.REPORT.REPLY.Q, the IBM MQ display route application orders and displays the activity information.
2. The following command puts a trace-route message into a queue manager network with the target queue specified as TARGET.Q. Activity information is accumulated within the trace-route message, but activity reports are not generated. On arrival at the target queue, the trace-route message is discarded. Depending on the value of the target queue manager attribute, ROUTEREC, a trace-route reply message can be generated and delivered to either the reply-to queue, TRR.REPLY.TO.Q, or to a system queue.

   ```
   dspmqrte -ac -ar -ro discard -rq TRR.REPLY.TO.Q -q TARGET.Q
   ```

   Providing a trace-route reply message is generated, and delivered to the reply-to queue TRR.REPLY.TO.Q, the IBM MQ display route application orders and displays the activity information that was accumulated in the trace-route message.

For more examples of using the IBM MQ display route application and its output, see IBM MQ display route application examples.

dspmqspl

Use the **dspmqspl** command to display a list of all policies and details of a named policy.

Syntax

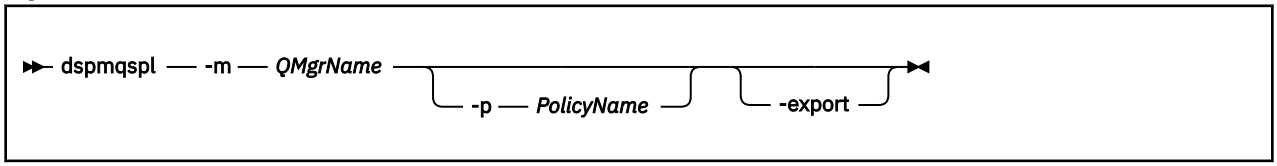


Table 23. dspmqspl command flags.

Command flag	Explanation
-m	Queue manager name (**mandatory**).
-p	Policy name.

Table 23. dspmqspl command flags. (continued)	
Command flag	**Explanation**
`-export`	Adding this flag generates output which can easily be applied to a different queue manager.

Related reference

"SET POLICY" on page 1284
Use the MQSC command SET POLICY to set a security policy.

"DISPLAY POLICY" on page 1113
Use the MQSC command DISPLAY POLICY to display a security policy.

"setmqspl" on page 144
Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy.

dspmqtrc

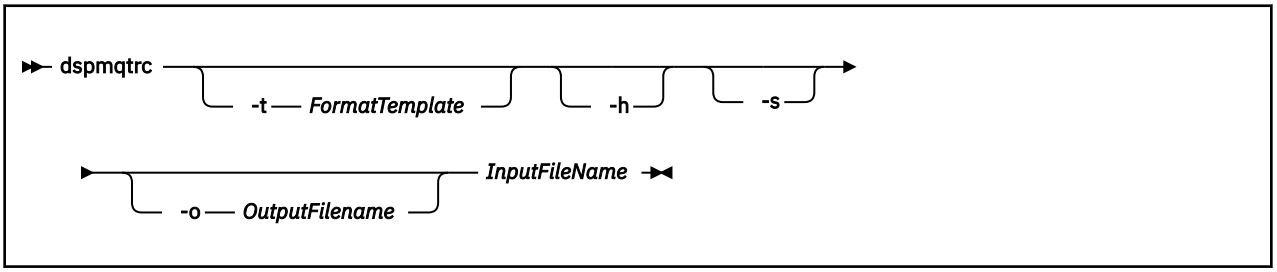
Format and display IBM MQ trace.

Purpose

The `dspmqtrc` command is supported on UNIX and HP Integrity NonStop Server systems only. Use the `dspmqtrc` command to display IBM MQ formatted trace output.

The runtime SSL trace files have the names AMQ.SSL.TRC and AMQ.SSL.TRC.1. You cannot format any of the SSL trace files. The SSL 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:

For AIX systems, the default value is as follows:

```
MQ_INSTALLATION_PATH/lib/amqtrc2.fmt
```

For all HP Integrity NonStop Server, and UNIX systems other than AIX systems, 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

Command	Description
endmqtrc	End trace
“strmqtrc” on page 154	Start trace

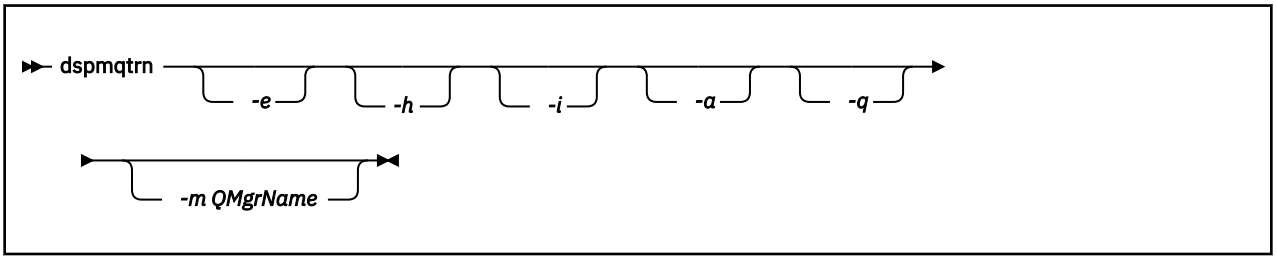
dspmqtrn

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 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.

-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 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.

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" on page 1082 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

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

Return code	Description
102	No transactions found

Example

Here is a typical use of the command, showing the form that is usually requested by IBM MQ Support:

```
dspmqtrn -m <qmgrname> -q -a
```

The output generated would be long. As an illustration, here is what one XA transaction might look like in the output:

```
TranNum(0,12294)
   TRANSTATE(ACTIVE)
   UOWLOGDA(2024-11-04)
   UOWLOGTI(14.58.32)
   UOWSTDA(2024-11-04)
   UOWSTTI(14.58.32)
   UOWLOG( )
   EXTURID(XA_FORMATID[40]
   XA_GTRID[737871746732313400000000000000000000000000
   00000000000000000000000004D8CF5C2E29D80014751A]
   XA_BQUAL[00000028000000100000003])
   OBJECT(TEST.Q)
```

Related tasks

Displaying outstanding units of work with the dspmqtrn command

Related reference

“rsvmqtrn” on page 101

Resolve in-doubt and heuristically completed transactions

dspmqver (display version information)

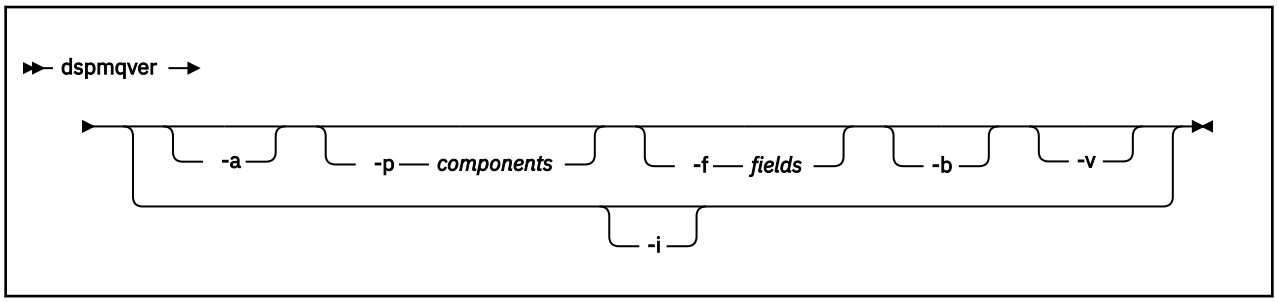
Display IBM MQ version and build information.

Purpose

Use the `dspmqver` command to display IBM MQ version and build information.

By default, the **dspmqver** command displays details of the installation from which it was invoked. 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.

When 32-bit support is missing from a 64-bit system, the **dspmqver -a** command might issue a message suggesting that the 32 bit version of IBM Global Security Kit (GSKit) is not installed. For more information, see the "Command failure" section of this topic.

-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 follow:

Value	Description
1	IBM MQ server, or client.
2	IBM MQ classes for Java.
4	IBM MQ classes for Java Message Service.
8	WebScale Distribution Hub
16 "1" on page 78	IBM MQ custom channel for Windows Communication Foundation
32	IBM Message Service Client for .NET (XMS .NET) - this component is only available on Windows
64	GSKit , or for HP Integrity NonStop Server, SSL When 32-bit support is missing from a 64-bit system, the **dspmqver -p 64** 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.
128	IBM MQ Advanced Message Security
V 8.0.0.4 256	IBM MQ AMQP Service
512	IBM MQ Telemetry Service

Notes:

1. 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 follow:

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

Value	Description
8	Build type
16	Platform
32	Addressing mode
64	Operating system
128	Installation path
256 [1]	Installation description
512 [1]	Installation name
1024 [1]	Maximum command level
2048 [1]	Primary installation
4096	Data Path
8192	License type

Note:

1. Not applicable to HP Integrity NonStop Server.

Information for each selected field is displayed on a separate line when the dspmqver command is run.

The default value is 8191. 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 dspmqver 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. Not applicable to HP Integrity NonStop Server.

Return codes

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**:

```
dspmqver
```

The following command displays information about all fields and components and is the equivalent of specifying `dspmqver -p 63 -f 4095`:

```
dspmqver -a
```

The following command displays version and build information for the IBM MQ classes for Java:

```
dspmqver -p 2
```

The following command displays the Common Services for Java Platform Standard Edition, IBM MQ, Java Message Service Client, and IBM MQ classes for Java Message Service:

```
dspmqver -p 4
```

The following command displays the build level of the WebScale Distribution Hub:

```
dspmqver -p 8 -f 4
```

The following command displays the name and build type for IBM MQ custom channel for Windows Communication Foundation:

```
dspmqver -p 16 -f 9
```

The following command displays information about installations of IBM MQ.

```
dspmqver -i
```

Command Failure

Failure when 32-bit support is missing from a 64-bit system

In IBM MQ Versions 8.0, 9.0 and 9.1, the 32-bit and 64-bit versions of IBM Global Security Kit (GSKit) are bundled together. When you run `dspmqver -a` or `dspmqver -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. 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 UNIX and Linux systems.

Failure when viewing the IBM MQ classes for Java

The **dspmqver** 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. For example, you might see the following message:

```
[root@blade883 ~]# dspmqver -p2
AMQ8351: IBM MQ Java environment has not been configured correctly.
```

To resolve this problem, ensure that the path is configured to include the JRE, and that the correct environment variables are set; for example, by using `setjmsenv` or `setjmsenv64`. For example:

```
export PATH=$PATH:/opt/mqm/java/jre/bin
cd /opt/mqm/java/bin/
. ./setjmsenv64

[root@blade883 bin]# dspmqver -p2
Name:       WebSphere MQ classes for Java
Version:    8.0.0.0
Level:      k000-L110908
Build Type: Production
```

UNIX Note that the **setjmsenv** and **setjmsenv64** commands apply to UNIX platforms only.

Windows On Windows you need to issue the **setmqenv** command to resolve error AMQ8351.

endmqcsv

Stop the command server for a queue manager.

Purpose

Use the **endmqscv** command to stop the command server on the specified queue manager.

You must use the **endmqscv** 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.

If the queue manager attribute, SCMDSERV, is specified as QMGR then changing the state of the command server using **endmqscv** 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

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

Command	Description
strmqcsv	Start a command server
dspmqcsv	Display the status of a command server

endmqlsr

End all listener process for a queue manager.

Purpose

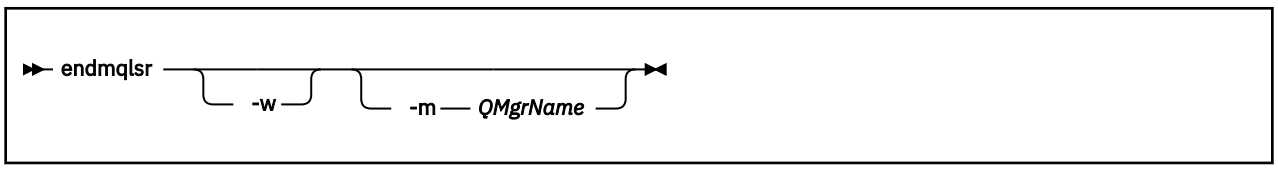
The **endmqlsr** command ends all listener processes for the specified queue manager.

You must use the **endmqlsr** 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.

You do not have to stop the queue manager before issuing the **endmqlsr** command. If any of the listeners are configured to have inbound channels running within the **runmqlsr** 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 **endmqlsr** 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

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

endmqdnm

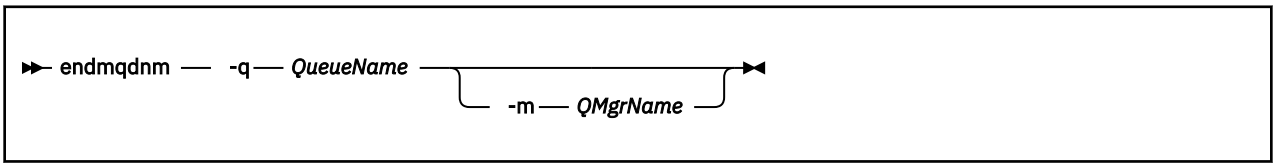
Stop the .NET monitor for a queue (Windows only).

Purpose

Note: The endmqdnm command applies to IBM MQ for Windows only.

Use the **endmqdnm** control command to stop a .NET monitor.

Syntax



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

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

endmqm

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

Purpose

Use the endmqm command to end (stop) a specified queue manager. This command stops a queue manager in one of three modes:

- Controlled or quiesced shutdown
- Immediate shutdown
- Pre-emptive shut down

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.

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 `-r` 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.

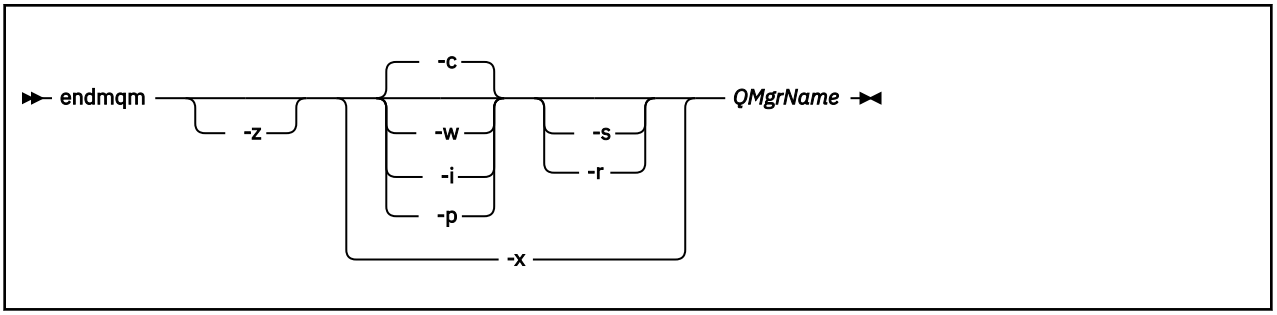
Note: The client might not reconnect to this queue manager. Depending on the MQCONNX reconnect option the client has used, and the definition of the queue manager group in the client connection table, the client might reconnect to a different queue manager. You can configure the client to force it to 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 `dspmq -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 **dltmqm** (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" on page 52 command is issued in the small timeframe between the applications disconnecting and the queue manager actually stopping, the "dspmq" on page 52 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.

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
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.

-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" on page 52 command is issued in the small timeframe between the applications disconnecting and the queue manager actually stopping, the "dspmq" on page 52 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
End a standby instance of the queue manager, without ending the active instance of the queue manager.

-z
Suppresses error messages on the command.

Return codes

Return code	Description
0	Queue manager ended
3	Queue manager being created
16	Queue manager does not exist
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
72	Queue manager name error
77	IBM MQ queue manager cannot switch over
79	Active instance of IBM MQ queue manager *QmgrName* not ended
90	Standby instance of IBM MQ queue manager *QmgrName* 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 Table 24 on page 87. The results of the command depend on whether the -s or -x switch is used, and the running status of local and remote instances of the queue manager.

Table 24. endmqm actions.

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 *QmgrName* not ended.
-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 *QmgrName* not ended.
-x	Active	None	79	AMQ8367	Active instance of IBM MQ queue manager *QmgrName* not ended.
		Standby			
	Standby	Active	0	-	Standby instance of queue manager QMNAME ended.

Related commands

Command	Description
“crtmqm” on page 25	Create queue manager
“endmqm” on page 84	End queue manager
“dltmqm” on page 34	Delete 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.

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" on page 149
Start the IBM MQ service on Windows.

endmqtrc

End trace for some or all of the entities that are being traced.

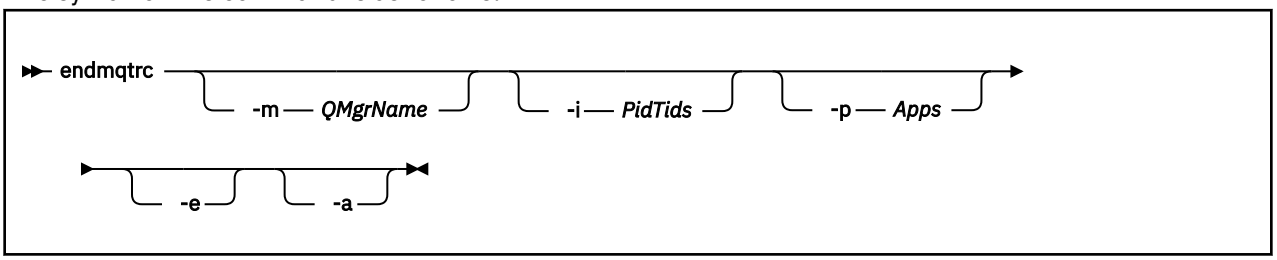
Purpose

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.

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. This parameter applies to server products only.

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.

This flag ***must*** be specified alone.

Return codes

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
   ```

HPE NonStop V8.1 These commands stop tracing of all queue manager processes on all CPUs:

```
run -cpu=0 endmqtrc -a
...
run -cpu=15 endmqtrc -a
```

Related commands

Command	Description
dspmqtrc	Display formatted trace output
“strmqtrc” on page 154	Start trace

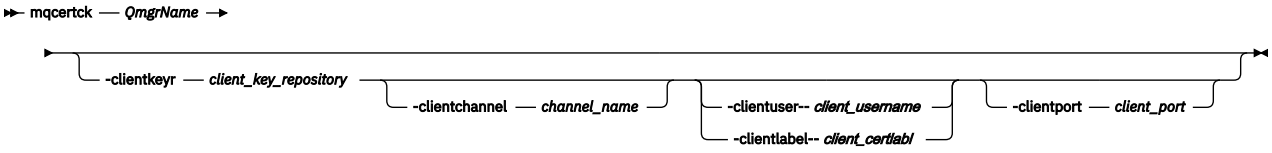
mqcertck (certify TLS setup)

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_certlabl*
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, 2025.
+-----------------------------------------------------------
| 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 ibmwebspheremqqmgr
```

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.

mqconfig

Checks that the system configuration meets the requirements to run IBM MQ (UNIX 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 Operating System configuration and tuning for IBM MQ on UNIX and Linux systems, and IBM i .

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
semmns   (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.

MQExplorer (launch MQ Explorer)

Start MQ Explorer (Windows and Linux x86-64 platforms only).

Purpose

To launch MQ Explorer by using the system menu on Linux, or the start menu on Windows, you must left-click on the installation that you want to launch.

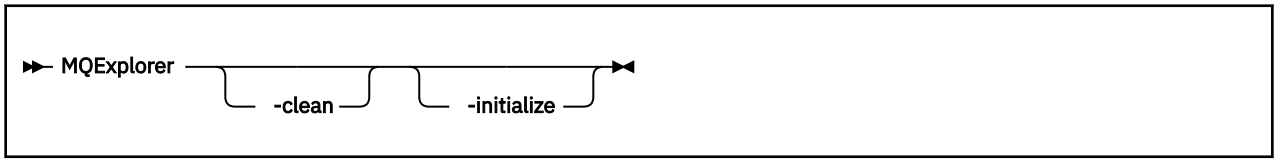
On Windows, open the start menu, and select the MQ Explorer installation entry under the **IBM MQ** folder that corresponds to the installation that you want to launch. Each instance of MQ Explorer listed is identified by the name that you chose for its installation.

On Linux, the system menu entry for MQ Explorer is added to the **Development** category. Where it appears within the system menu is dependent on your Linux distribution (SUSE or Red Hat), and your desktop environment (GNOME or KDE).

- On SUSE
  - Left-click **Computer > More Applications...,** and find the installation of MQ Explorer that you want to launch under the **Development** category.
- On Red Hat
  - The installation of MQ Explorer that you want to launch can be found under **Applications> Programming**.

Syntax

The **MQExplorer** command is stored in `MQ_INSTALLATION_PATH/bin`. **MQExplorer.exe** (the MQExplorer command) supports standard Eclipse runtime options. The syntax of this command is as follows:



Optional parameters

`-clean`
Is passed to Eclipse. This parameter causes Eclipse to delete any cached data used by the Eclipse runtime.

`-initialize`
Is passed to Eclipse. This parameter causes Eclipse to discard configuration information used by the Eclipse runtime.

The graphical user interface (GUI) does not start.

mqrc (MQ return code)

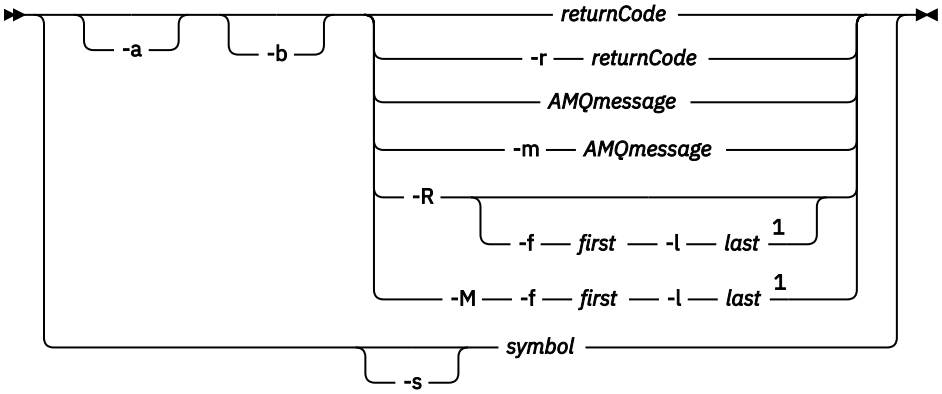
Display information about return codes.

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:

[1] 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

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
```

rcdmqimg

Write the image of an object or group of objects to the log for media recovery.

Purpose

Use the **rcdmqimg** 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 **rcrmqobj** to recreate the object from the image.

rcdmqimg must be run manually or from an automated task you have created. 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 .

Running **rcdmqimg** moves the log sequence number (LSN) forwards and frees up old log files for archival or deletion.

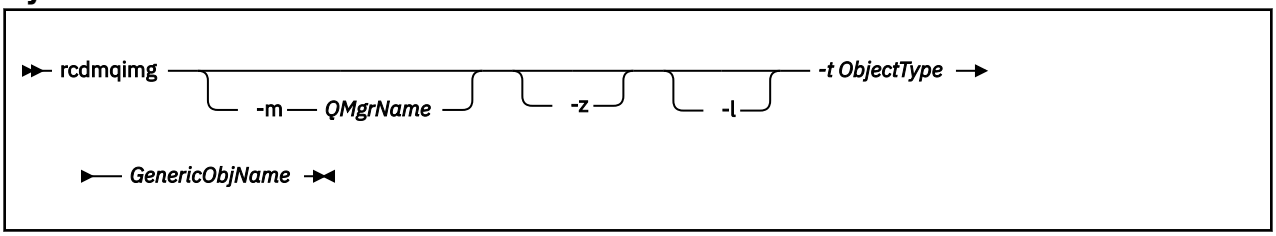
When determining when and how often to run **rcdmqimg**, consider these factors:

Disk space
If disk space is limited, regular running of **rcdmqimg** releases log files for archive or deletion.

Impact on normal system performance
rcdmqimg 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 **rcdmqimg** 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:

Table 25. Valid object types

Object Type	Description
all and *	All the object types; **ALL** for objtype and * for GenericObjName

Table 25. Valid object types (continued)	
Object Type	**Description**
authinfo	Authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	Channels
clntconn or **clcn**	Client connection channels
catalog or **ctlg**	An object catalog
listener or **lstr**	Listeners
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
qmgr	Queue manager object
service or **srvc**	Service
syncfile	Channel synchronization file.
topic or **top**	Topics

Note: When using IBM MQ for UNIX 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

Return code	Description
0	Successful operation
26	Queue manager running as a standby instance.

Return code	Description
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

Command	Description
rcrmqobj	Recreate a queue manager object

rcrmqobj

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.

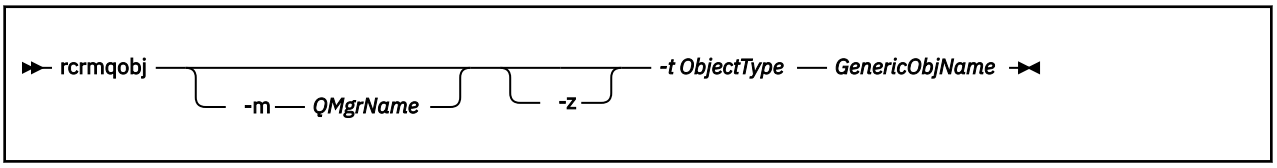
Note: Use this command on a running queue manager.

- 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:

Table 26. Valid object types.

Object Type	Description
***** or **all**	All object types
authinfo	Authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	Channels
clntconn or **clcn**	Client connection channels
clchltab	Client channel table
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 26. 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 AMQ7353 (krcE_SYNCFILE_UPDATE_FAILED).
topic or **top**	Topics

Note: When using IBM MQ for UNIX 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

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
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

Command	Description
rcdmqimg	Record an object in the log

rmvmqinf

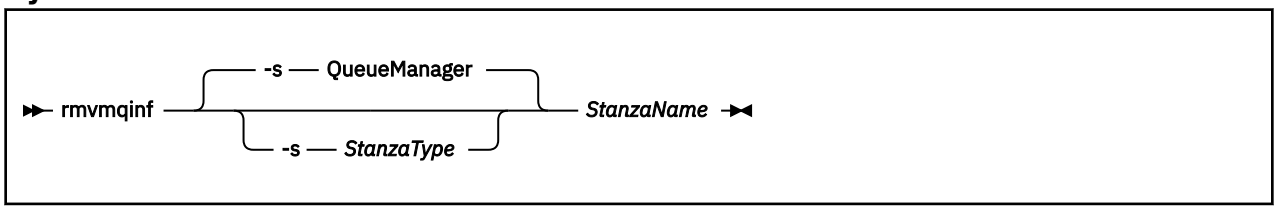
Remove IBM MQ configuration information (Windows and UNIX platforms only).

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

Return code	Description
0	Successful operation
5	Queue manager is running
26	Queue manager is running as a standby instance

Return code	Description
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

Command	Description
“addmqinf” on page 10	Add queue manager configuration information
“dspmqinf” on page 62	Display queue manager configuration information

rsvmqtrn

Resolve in-doubt and heuristically completed transactions

Purpose

The `rsvmqtrn` command is used to resolve two different transaction states.

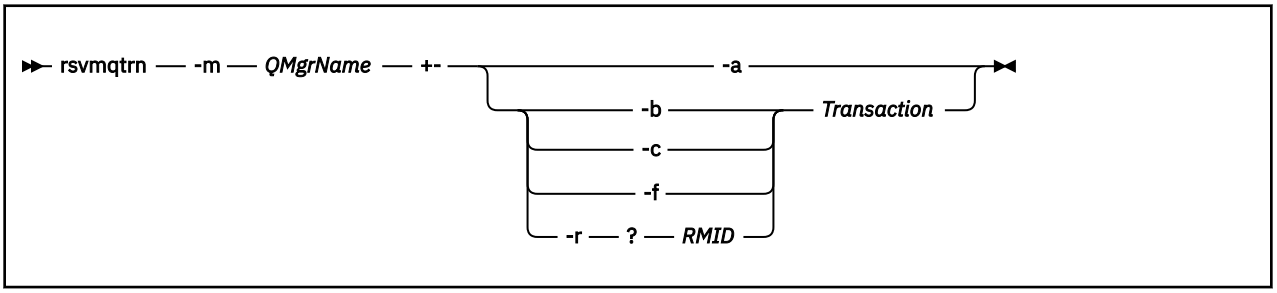
in-doubt transactions
Use the `rsvmqtrn` command to commit or back out internally or externally coordinated in-doubt transactions.

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 option 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 **dspmqtrn** 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

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

Command	Description
dspmqtrn	Display list of prepared transactions

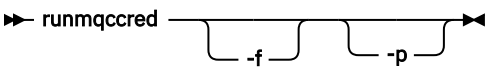
runmqccred (obfuscate passwords for mqccred exit)

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



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 filesystem 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 doesn't support the Windows NTFS 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, You must run the program on a Version 8.0 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:

Windows platforms
The `<installation directory>\Tools\c\Samples\mqccred\`

UNIX platforms
The `<installation directory>/usr/mqm/samp/mqccred/`

If the file permissions are not secure enough **runmqccred** produces this message:

```
Configuration file 'C:\Users\User1\.mqs\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\.mqs\mqccred.in' processed successfully.
Plaintext passwords found: 3
```

runmqchi

Run a channel initiator process to automate starting channels.

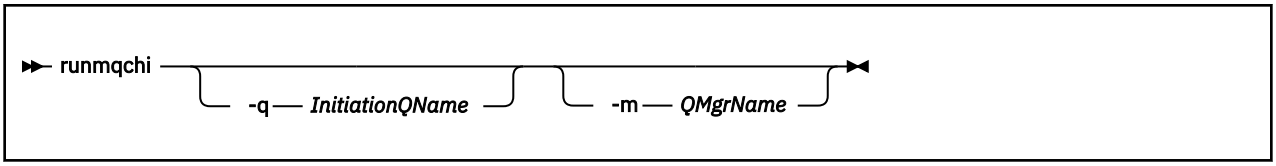
Purpose

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 `dspmq -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

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.

runmqchl

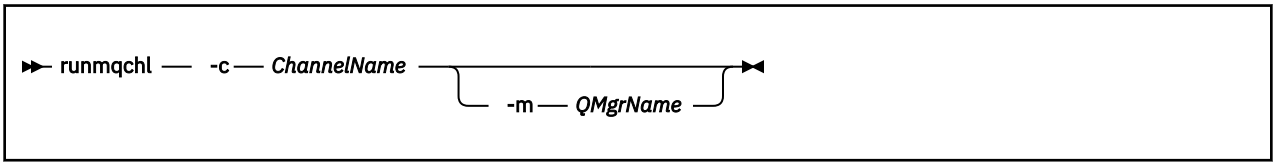
Start a sender or requester channel

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

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.

runmqdlq

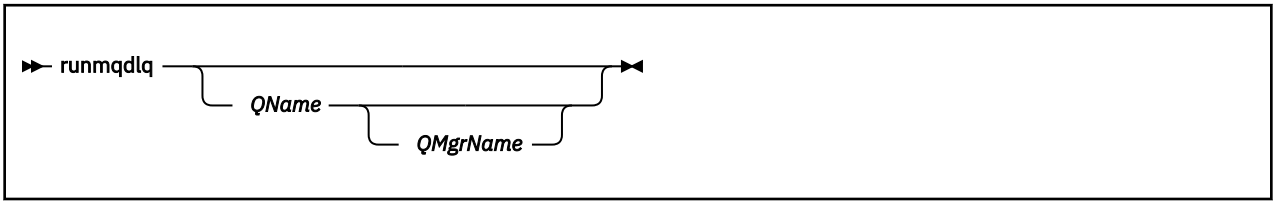
Start the dead-letter queue handler to monitor and process messages on the dead-letter queue.

Purpose

Use the `runmqdlq` command to start the dead-letter queue (DLQ) handler, which monitors and handles messages on a dead-letter queue.

This command is used on servers. If you want client mode you should compile **amqsdlq** in client mode. See The sample DLQ handler **amqsdlq** for more information.

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. In IBM MQ for AIX, Solaris, HP-UX, and Linux, the DLQ handler does not start to process the named queue until it receives an end_of_file (Ctrl+D) character. In IBM MQ for Windows, it

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.

runmqdnm

Start processing messages on a queue using the .NET monitor (Windows only).

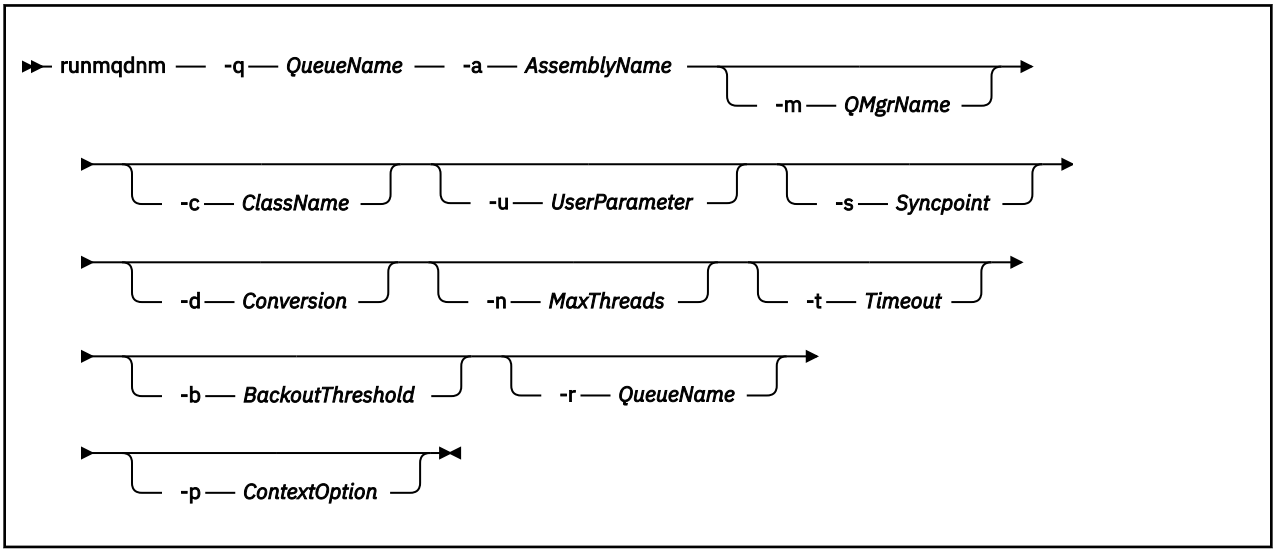
Purpose

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 27. 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 28. 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:

Table 29. BackoutThreshold parameter values.

Value	Description
-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:

Table 30. ContextOption parameter values.

Value	Description
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

Return code	Description
0	Successful operation
36	Invalid arguments supplied
40	Queue manager not available

Return code	Description
49	Queue manager stopping
58	Inconsistent use of installations detected
71	Unexpected error
72	Queue manager name error
133	Unknown object name error

runmqlsr

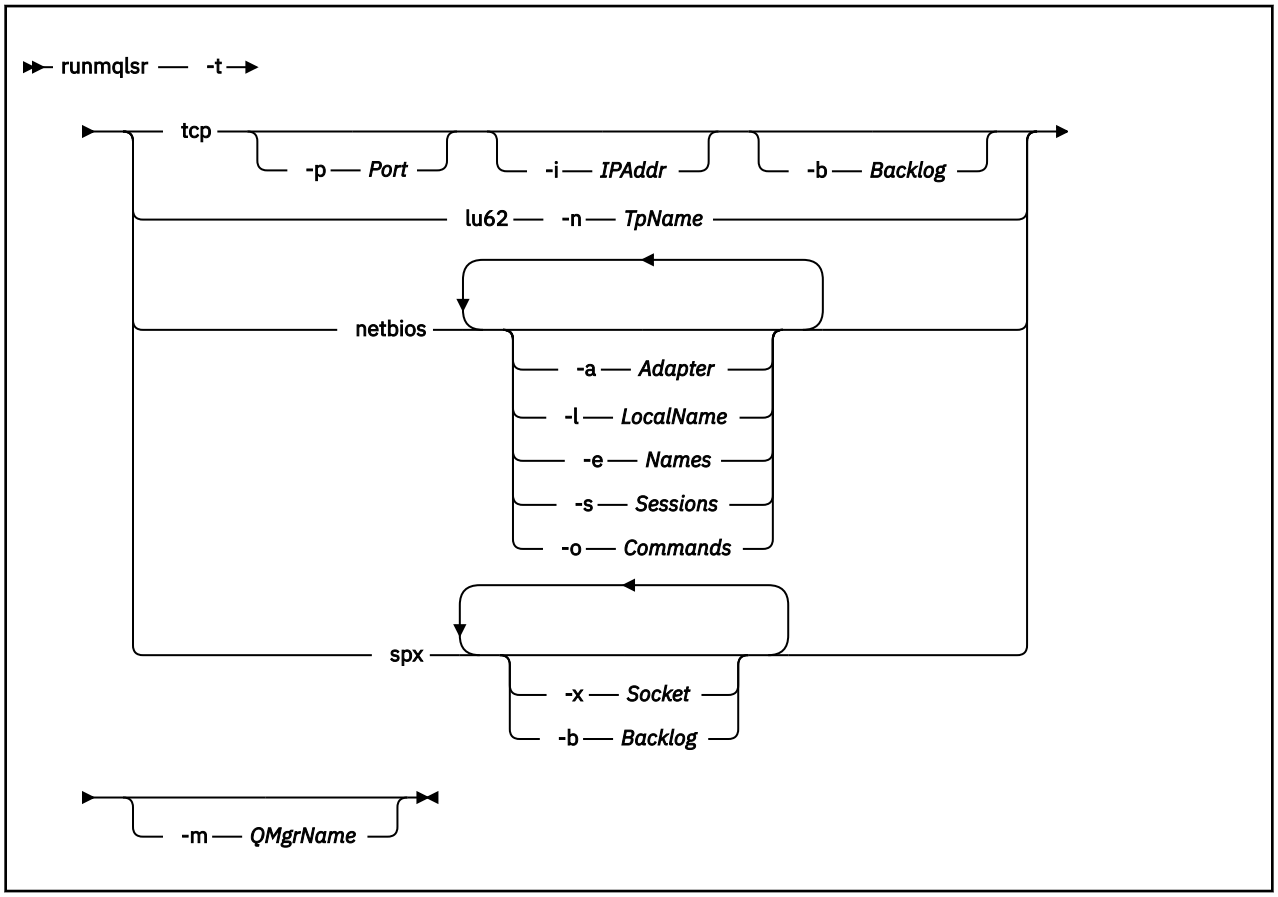
Run a listener process to listen for remote requests on various communication protocols.

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:

Table 31. Transmission protocol values.

Header	Header
`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

Return code	Description
0	Command completed normally
4	Command completed after being ended by the **endmqlsr** 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
```

distributed runmqras

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.

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.

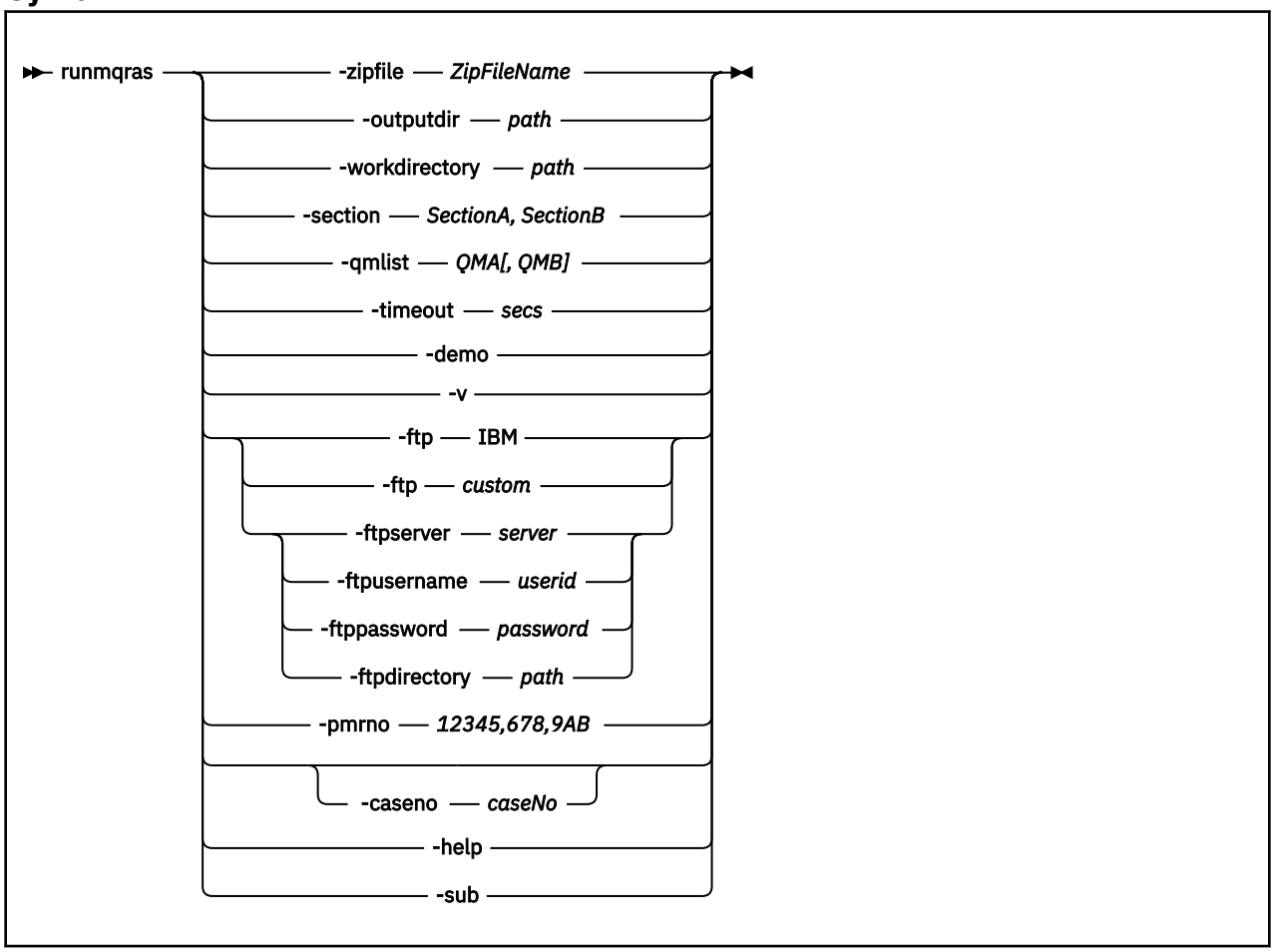
These additional *sections* gather more detailed information, depending on the type of problem being diagnosed. If non-default sections are needed by IBM support personnel, they will tell you.

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 to allow the command to gather queue manager files and command outputs.

Solaris Linux AIX V 8.0.0.9 From IBM MQ 8.0.0, Fix Pack 9, the **runmqras** command, by default, retrieves the environment variable information. This applies to Linux, Solaris, and AIX.

distributed V 8.0.0.9 From IBM MQ 8.0.0, Fix Pack 9, the **runmqras** command retrieves a listing of the queue manager's data directory by default. This applies to distributed platforms.

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.

-zipfile *ZipFileName*
Supply the file name of the resulting archive.

By default, the name of the output archive is `runmqras.zip`.

-outputdir *path*
The directory in which the resulting output file is placed.

By default, the output directory is the same as the work directory.

-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 under `/tmp` is used on UNIX systems, and under `%temp%` is used on Windows, whose name starts with **runmqras** and is suffixed by the date and time.

-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.

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, UNIX and IBM i, and in `set.stdout` on Windows) and current user limits (saved in `mqconfig.stdout` on Linux and UNIX) 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.

nodefault
Prevents the default collections from occurring, but other explicitly requested sections are still collected.

trace
Gathers all the trace file information plus the default information.

Note: Does not enable tracing.

defs
Gathers the queue manager definitions and status information.

cluster
Gathers cluster configuration and queue information.

dap
Gathers transaction and persistence information.

kernel
Gathers queue manager kernel information.

logger
Gathers recovery logging information.

topic
Gathers topic tree information.

QMGR
Gathers all queue manager files: queues, logs, and configuration files.

V 8.0.0.9 **leak**
From IBM MQ 8.0.0, Fix Pack 9, gathers IBM MQ process resource usage information.

Linux UNIX This section applies to Linux, HP-UX, Solaris, and AIX.

V 8.0.0.8 **mft**
From IBM MQ 8.0.0, Fix Pack 8, captures the data obtained by the **fteRas** command.

Note: `-section mft` only collects information for the default coordination queue manager topology.

For more information, see Section names and descriptions, in the IBM technote on using the IBM MQ **runmqras** command to collect data.

-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.

-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.

-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.

-v
Extends the amount of information that is logged in the `console.log` file, contained in the output `.zip` file.

-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, either directly into IBM, or to a site of your choosing. If you select the *ibm* option, anonymous FTP is used to deliver the archive into the IBM ECuRep server. This process is identical to submitting the file manually using FTP.

Note if you select the *ibm* option, you must also provide the *pmrno* option, and all other FTP* options are ignored.

-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.

-ftppassword ***password***
The password to log in to the FTP server with, when an FTP custom option is used.

-ftpdirectory ***path***
The directory on the FTP server to place the resulting `.zip` file into, used when an FTP custom option is used.

-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.

-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: -caseno is equivalent to **-pmrno** and both are optional parameters, but it is not permitted to supply both together.

-help
Provide simple help.

-sub
Shows the keywords that will be substituted in the xml.

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, and sends it directly into IBM to be associated with PMR number 11111,222,333 using the basic FTP capability:

```
runmqras -ftp ibm -pmrno 11111,222,333
```

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
```

Return codes

A non zero return code indicates failure.

runmqsc

Run IBM MQ commands on a queue manager.

Purpose

Use the **runmqsc** command to issue MQSC commands to a queue manager. MQSC commands enable you to perform administration tasks. For example, you can define, alter, or delete a local queue object. MQSC commands and their syntax are described in MQSC commands.

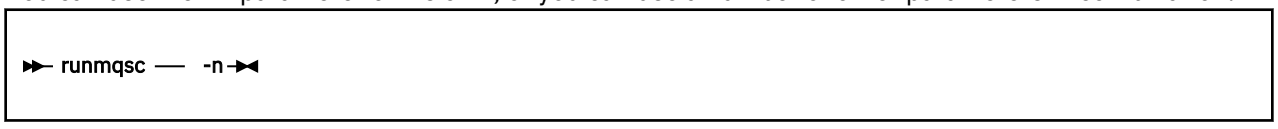
You must use the **runmqsc** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with by using the `dspmq -o installation` command.

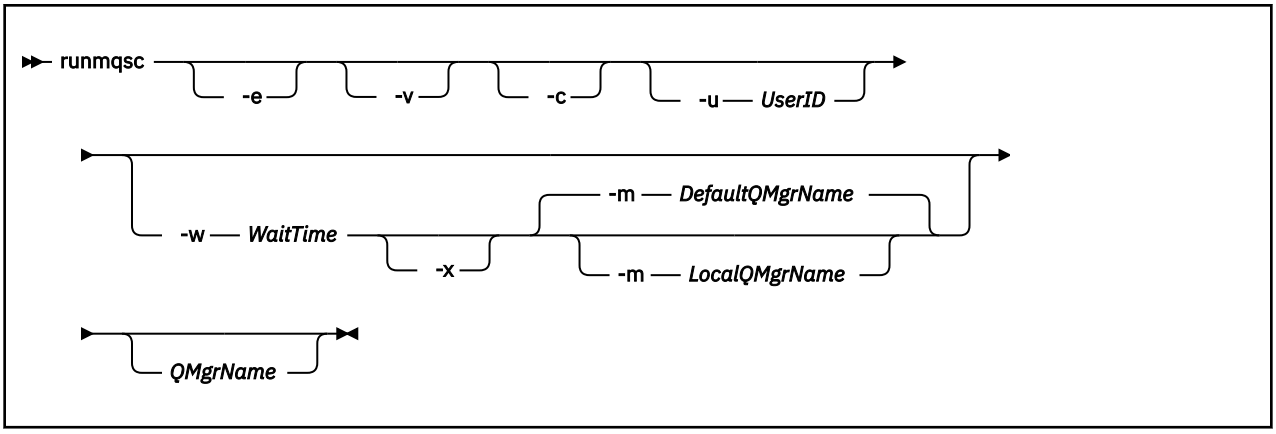
To stop the **runmqsc** command, use the **end** command. You can also use the **exit** or the **quit** command.

From IBM MQ 8.0, you can make it easier to see that you are in an MQSC environment and see some details of the current environment by setting a prompt of your choice by using the MQPROMPT environment variable. For more information, see Performing local administration tasks using MQSC commands.

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.

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.

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 recognised:

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.

issuing 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**.

-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 ql(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. See Preparing channels and transmission queues for remote administration for more information.

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. 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.

Return codes

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

Examples

1. Enter this command at a command prompt:

```
runmqsc
```

Now you can enter MQSC commands directly at the command prompt. No queue manager name is specified, so the MQSC commands are processed on the default queue manager.

2. Use one of these commands, as appropriate in your environment, to specify that MQSC commands are to be verified only:

```
runmqsc -v BANK < "/u/users/commfile.in"

runmqsc -v BANK < "c:\users\commfile.in"
```

The queue manager name is BANK. The command verifies the MQSC commands in file `commfile.in` and displays the output in the current window.

3. These commands run the MQSC command file `mqscfile.in` against the default queue manager.

```
runmqsc < "/var/mqm/mqsc/mqscfile.in" > "/var/mqm/mqsc/mqscfile.out"

runmqsc < "C:\Program Files\IBM\WebSphere MQ\mqsc\mqscfile.in" >
        "C:\Program Files\IBM\WebSphere MQ\mqsc\mqscfile.out"
```

In this example, the output is directed to file `mqscfile.out`.

4. This command submits commands to the QMREMOTE queue manager, using QMLOCAL to submit the commands.

```
runmqsc -w 30 -m QMLOCAL QMREMOTE
```

Related tasks

Performing local administration tasks using MQSC commands

runmqtmc

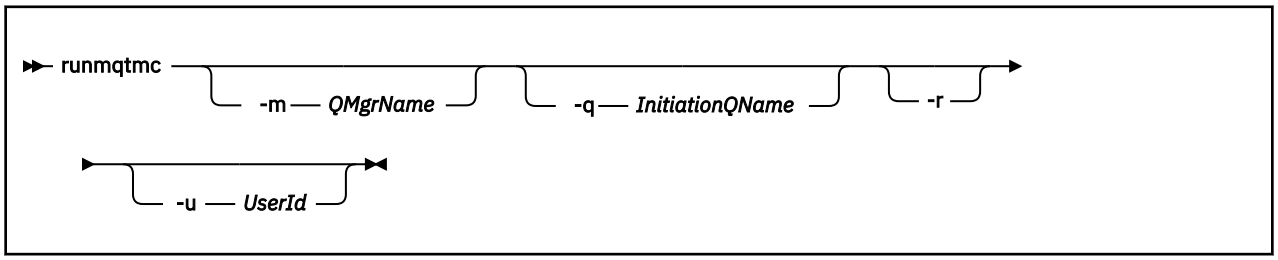
Start the trigger monitor on a client.

Purpose

Use the `runmqtmc` 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" on page 84. 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 **runmqtmc** 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

Return code	Description
0	Not used. The client trigger monitor is designed to run continuously and therefore not to end. The value is reserved.
10	Client 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.

runmqtrm

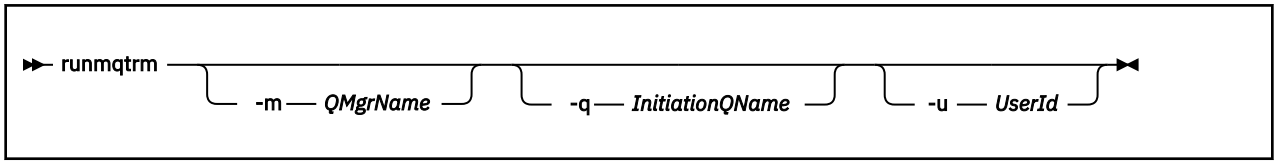
Start the trigger monitor on a server.

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” on page 84. 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

Return code	Description
0	Not used. The trigger monitor is designed to run continuously and therefore not to end. Hence a value of 0 would not be seen. The value is reserved.
10	Trigger monitor interrupted by an error.
20	Error; trigger monitor not run.

runswchl

runswchl (switch cluster channel) on UNIX, Linux, and Windows.

Purpose

The command switches or queries the cluster transmission queues associated with cluster-sender channels.

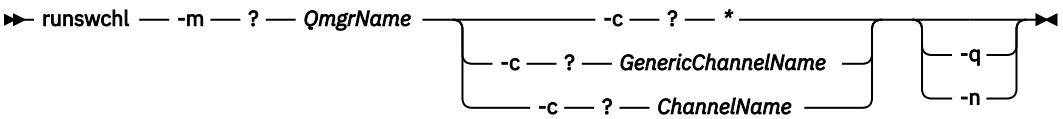
Usage notes

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



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 commands 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

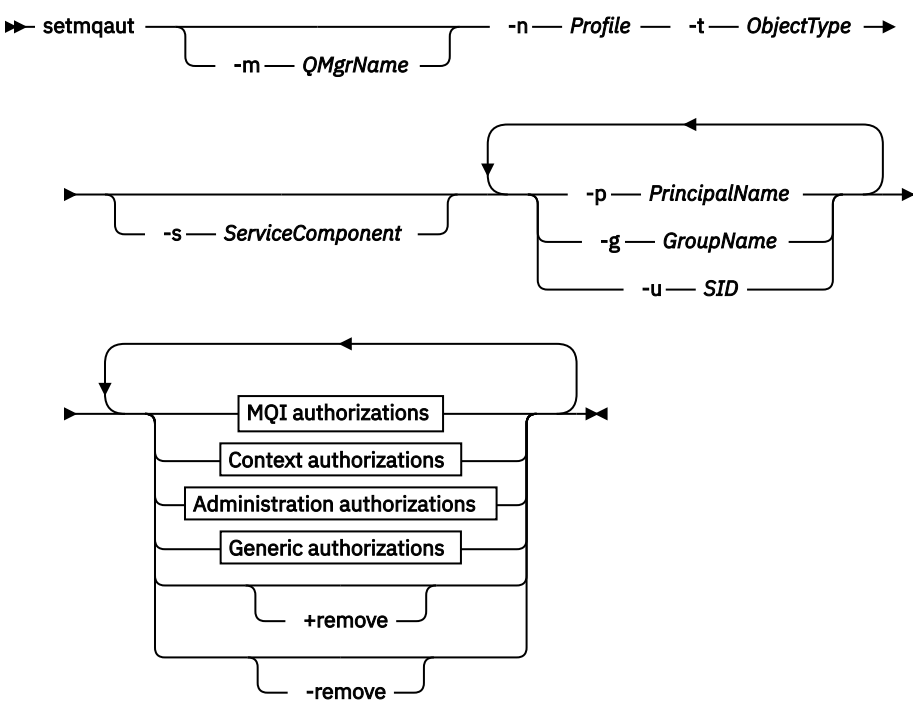
Change the authorizations to a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

For more information about authorization service components, see Installable services, Service components, and Authorization service interface.

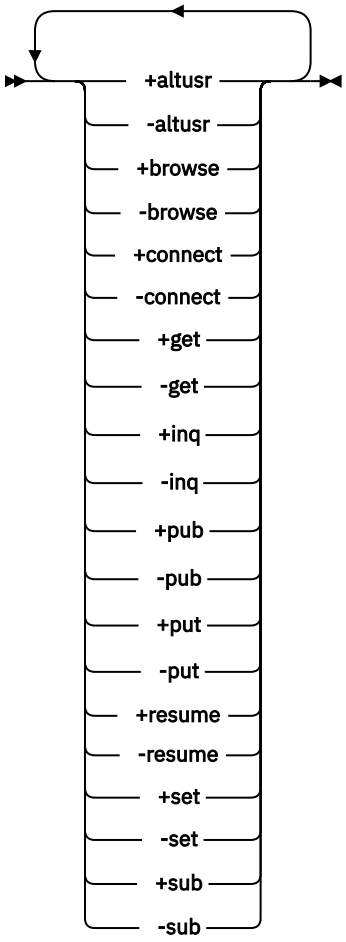
For more information about how authorizations work, see How authorizations work.

Linux UNIX 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 User-based permissions on Unix and Linux.

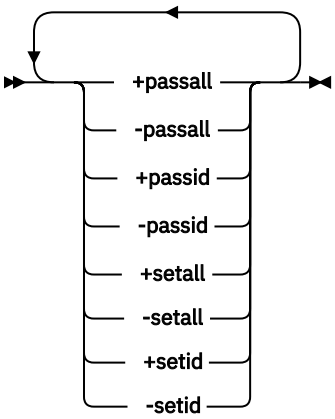
Syntax



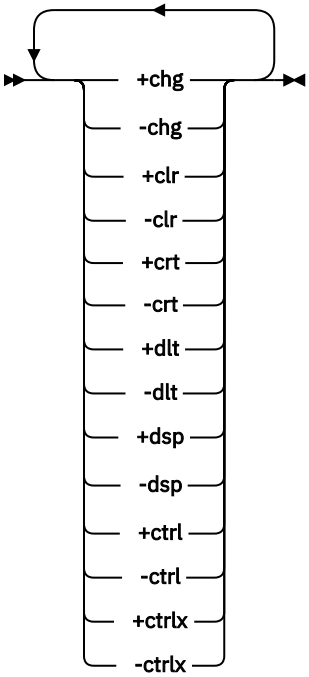
MQI authorizations



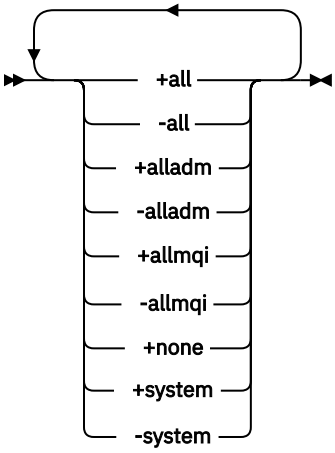
Context authorizations



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 UNIX, 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 UNIX On UNIX and Linux systems, for Version 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 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 UNIX On UNIX 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 Version 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 32. ObjectType values.

Value	Description
authinfo	An authentication information object
channel or **chl**	A channel
clntconn or **clcn**	A client connection channel
comminfo	A communication information object
listener or **lstr**	A listener
namelist or **nl**	A namelist
process or **prcs**	A process
queue or **q**	A queue
qmgr	A queue manager
rqmname or **rqmn**	A remote queue manager name
service or **srvc**	A service
topic or **top**	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 UNIX, 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.

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.

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 **GroupModel** attribute to *GlobalGroups* in the Security 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 33 on page 129 shows the authorities that can be given to the different object types.

Table 33. Specifying authorities for different object types

Authority	Queue	Process	Queuemanager	Remote queuemanager name	Namelist	Topic	Authinfo	Clntconn	Channel	Listener	Service	Cominfo
all [1]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
alladm [2]	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
allmqi [3]	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 [4]	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
setall [5]	Yes	No	Yes	No	No	No	No	No	No	No	No	No
setid [5]	Yes	No	Yes	No	No	Yes	No	No	No	No	No	No
sub	No	No	No	No	No	Yes	No	No	No	No	No	No

Table 33. Specifying authorities for different object types (continued)

Authority	Queue	Process	Queuemanager	Remote queuemanager name	Namelist	Topic	Authinfo	Clntconn	Channel	Listener	Service	Cominfo
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. `allmqi` authority 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

Return code	Explanation
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
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

"SET AUTHREC" on page 1268
Use the MQSC command SET AUTHREC to set authority records associated with a profile name.

Authorizations for MQI calls

Table 34. Authorizations for MQI calls.

Value	Description
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.) If you are using IBM MQ prior to Version 7.0.1.4, you must set +altusr for the group containing the user ID specified in MCAUSER on a receiver channel. This action prevents error message AMQ2035 appearing if you reset the sequence number of the corresponding sender channel.
browse	Retrieve a message from a queue using an MQGET call with the BROWSE option.

Table 34. Authorizations for MQI calls. (continued)	
Value	**Description**
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.

Authorizations for context

Table 35. Authorizations for context.	
Value	**Description**
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. 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.

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 36. Authorizations for commands.	
Value	**Description**
chg	Change the attributes of the specified object.
clr	Clear the specified queue or a topic.

Table 36. Authorizations for commands. (continued)	
Value	**Description**
crt	Create objects of the specified type.
dlt	Delete the specified object. Note, that the dlt authority has no effect on a queue manager object.
dsp	Display the attributes of the specified object.
ctrl	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.
ctrlx	Reset or resolve the specified channel.

Authorizations for generic operations

Table 37. 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.
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.

setmqcrl

Administer CRL (certificate revocation list) LDAP definitions in an Active Directory (Windows only).

Purpose

Note: The `setmqcrl` command applies to IBM MQ for Windows only.

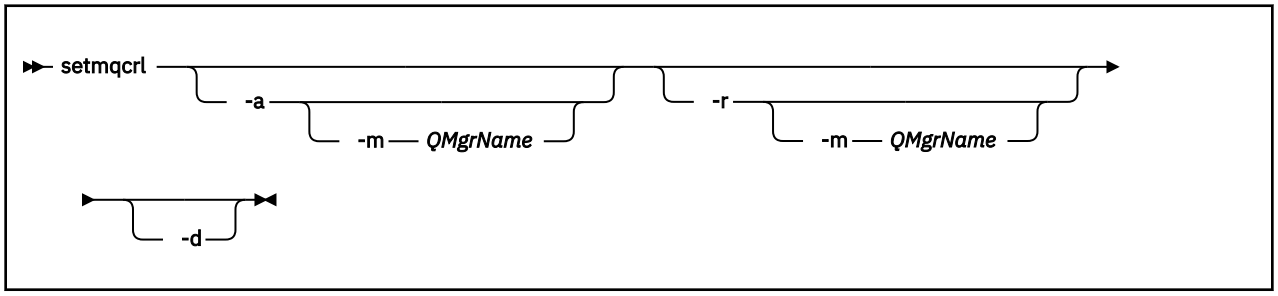
Use the `setmqcrl` command to configure and administer support for publishing CRL (certificate revocation list) LDAP definitions in an Active Directory.

A domain administrator must use this command, or `setmqscp` , 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, **deleting any other CRL definitions from the Active Directory first**:

```
setmqcrl -a -m Paint.queue.manager
```

setmqenv

Use the **setmqenv** command to set up the IBM MQ environment on UNIX, 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" on page 21 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_JAVA_INSTALL_PATH
- MQ_JAVA_DATA_PATH
- MQ_JAVA_LIB_PATH
- MQ_JAVA_JVM_FLAG
- MQ_JRE_PATH
- PATH

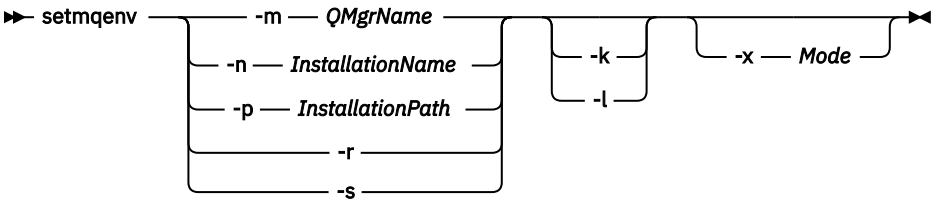
On UNIX and Linux systems, if the **-l** or **-k** flag is specified, the *LIBPATH* environment variable is set on AIX, and the *LD_LIBRARY_PATH* environment variable is set on HP-UX, Linux, and Solaris.

Usage notes

- If you have installed IBM MQ V 7.0.1, do not use the **setmqenv** command. Some of the components of IBM MQ V 7.0.1, such as Explorer, reference the environment variables for their library paths and therefore will not work if the **setmqenv** command has been used to alter the environment variables to point to an IBM MQ V 7.0.1 installation path.
- 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.

-k
UNIX and Linux only.

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.

-l
UNIX and Linux only.

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.

Return codes

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 the `/opt/mqm` directory on a UNIX or Linux system.

Note: The period character (`.`) 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.

- 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
```

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

This command sets up the environment for an installation called `Installation1`:

```
"C:\Program Files\IBM\WebSphere MQ\bin\setmqenv.cmd" -n Installation1
```

Related concepts
Multiple installations
Related tasks
Choosing a primary installation
Related reference
“crtmqenv” on page 21
Create a list of environment variables for an installation of IBM MQ, on UNIX, Linux, and Windows.

setmqinst

Set IBM MQ installations, on UNIX, Linux, and Windows.

Purpose

You can use the **setmqinst** command to change the installation description of an installation, or to set or unset an installation as the primary 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.

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.

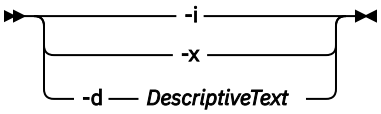
On UNIX platforms you should not assume that the current directory is in the path. If you are in `/opt/mqm/bin` and want to run, for example, `/opt/mqm/bin/dspmqver` you need to enter "**/opt/mqm/bin/dspmqver**" or "**./dspmqver**".

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.

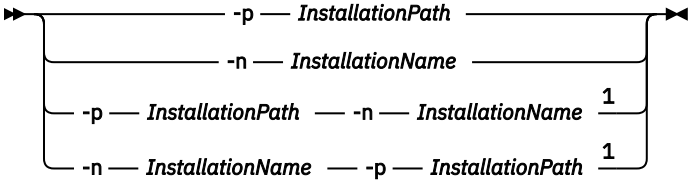
On UNIX or Linux systems, you must run this command as root. On Windows systems, you must run this command as a member of the Administrators group. The command does not have to be run from the installation you are modifying.

Syntax

Action



Installation



Notes:

[1] When specified together, the installation name and installation path must refer to the same installation.

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. You must use double quotation marks around the path if it contains spaces

Return codes

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

Return code	Description
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.

Related tasks

Choosing a primary installation

Changing the primary installation

setmqm

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.

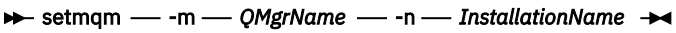
This command is only applicable to UNIX, Linux and Windows.

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 **dspmq** command. See "dspmq" on page 52 for more information.

Syntax



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

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
```

setmqprd

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. UNIX requires root access, and Windows with UAC (User Account Control) requires Administrator access to run this command.

Syntax

Required parameters

LicenseFile
Specifies the fully-qualified name of the production license certificate file.

The full license file is `amqpcert.lic`:

- On UNIX and Linux, it is in the */MediaRoot*`/licenses` directory on the installation media.
- 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.
- On IBM i, issue the command

```
CALL PGM(QMQM/SETMQPRD) PARM('/QOPT/OPT01/amqpcert.lic')
```

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
Converting a trial license on UNIX, Linux, and Windows

setmqscp

Publish client connection channel definitions in an Active Directory (Windows only).

Purpose

Note: The `setmqscp` command applies to IBM MQ for Windows only.

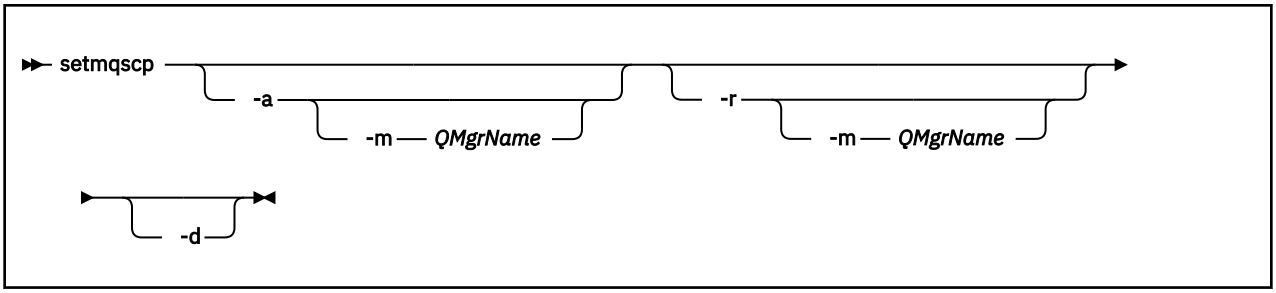
Use the `setmqscp` command to configure and administer support for publishing client connection channel definitions in an Active Directory.

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
```

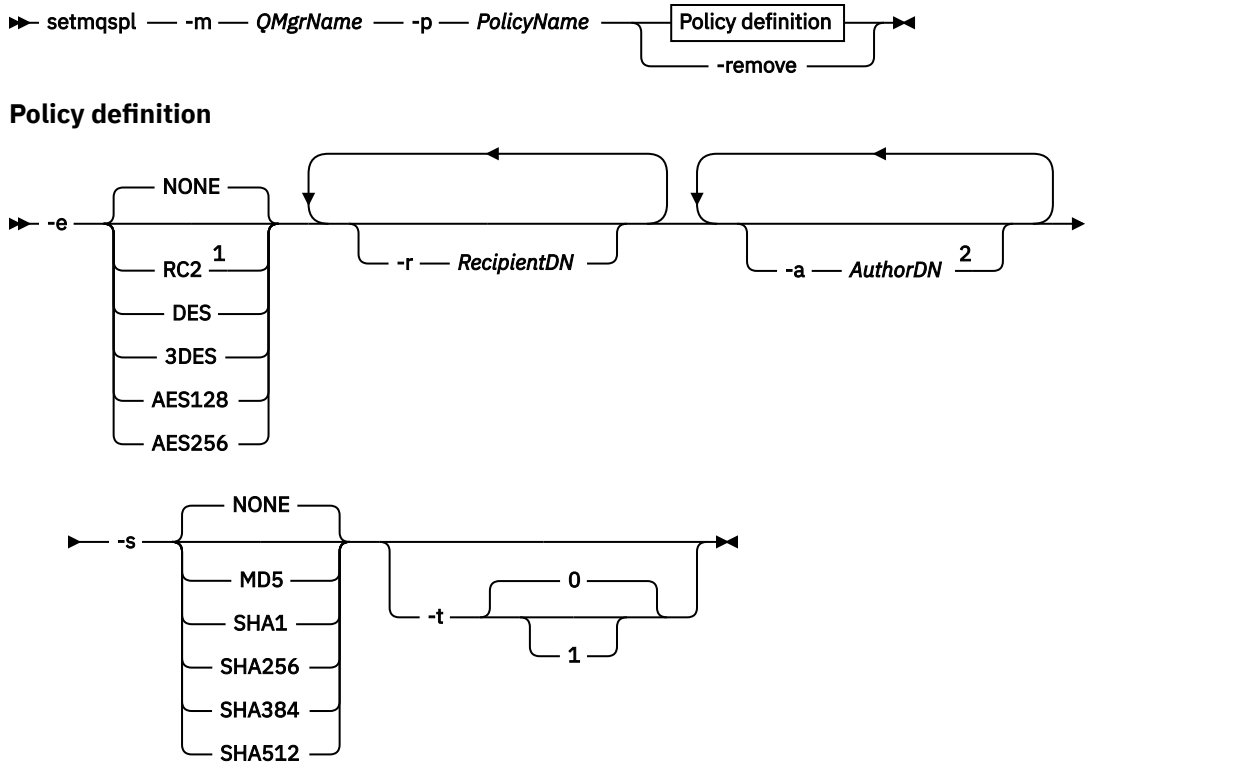
The following command migrates all client connection definitions on the local server to the Active Directory:

```
setmqscp -a -m *
```

setmqspl

Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy.

Syntax



Notes:

[1] If an encryption algorithm is selected, a recipient DN must also be provided.
[2] If an author DN is provided, a signing algorithm must also be selected.

Table 38. `setmqspl` command flags.

Command flag	Explanation
`-m`	Queue manager name. This flag is mandatory for all actions on security policies.
`-p`	Policy name. Set the policy name to the name of the queue you want the policy to apply to.

Table 38. `setmqspl` command flags. (continued)

Command flag	Explanation
`-s`	Digital signature algorithm. IBM MQ Advanced Message Security supports the following values: MD5, SHA1, SHA256, SHA384, and SHA512. All must be in uppercase. The default value is NONE **Important:** • For the SHA384 and SHA512 cryptographic hash functions, keys used for signing must be longer than 768 bits. • The name, or names, of the encryption algorithms must be in uppercase. • The signature algorithm cannot be NONE if the encryption algorithm is different from NONE.
`-e`	Digital encryption algorithm. IBM MQ Advanced Message Security supports the following encryption algorithms: RC2, DES, 3DES, AES128, AES256. The default value is NONE. **Important:** • The name of the encryption algorithm must be provided in uppercase • z/OS On z/OS encryption algorithm RC2 is not supported for confidentiality policies.
`-r`	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 **`-r`** flag. **Important:** • DN attribute names must be in uppercase. • Commas must be used as a name separators. • To avoid command interpreter errors, place quotation marks around the DNs. For example: `-r "CN=alice, O=ibm, C=US"`

Table 38. `setmqspl` command flags. (continued)	
Command flag	**Explanation**
`-a`	Signature DN that is validated during message retrieval. Only messages signed by a user with a DN provided are accepted during the retrieval. Signature DNs can be specified only if the signature algorithm is different from NONE. Multiple authors can be included. Each author needs to have a separate `-a` flag. **Important:** DN attribute name must be in uppercase.
`-t`	Toleration flag that indicates whether a policy that is associated with a queue can be ignored when an attempt to retrieve a message from the queue involves a message with no security policy set. Valid values include: • **0 (default)** Toleration flag off. • **1** Toleration flag on. 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.
`-remove`	Delete policy. If specified, only the `-p` flag remains valid.

Related reference

"SET POLICY" on page 1284
Use the MQSC command SET POLICY to set a security policy.

"DISPLAY POLICY" on page 1113
Use the MQSC command DISPLAY POLICY to display a security policy.

"dspmqspl" on page 73
Use the **dspmqspl** command to display a list of all policies and details of a named policy.

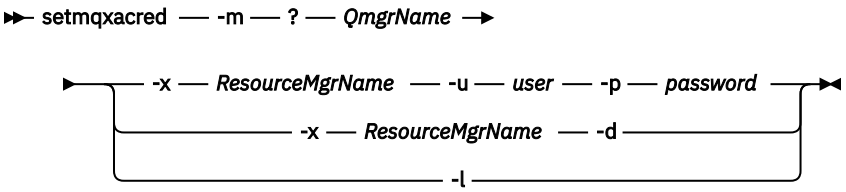
V 8.0.0.4 setmqxacred (add XA credentials)

Use the **setmqxacred** command to add or modify credentials in the IBM MQ XA credentials store.

Purpose

The command adds new credentials to the IBM MQ XA credentials store, or modifies or deletes existing credentials.

Syntax



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`:

```
# setmqxacred -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`:

```
# setmqxacred -m QM1 -x mydb2 -d
Successfully removed credentials for XA Resource Manager mydb2
```

To list details about the credentials stored in the credentials store.

```
# setmqxacred -m QM1 -l
ResourceName(mydb2) UserName(user1)
ResourceName(myora) UserName(user2)
```

strmqcfg

Start MQ Explorer (Windows and Linux x86-64 platforms only).

Purpose

For IBM MQ for Windows only, note that if you use `runas` to execute this command, you must define the Environment Variable *APPDATA* to set a path to a directory that the user you are running as has access. For example:

```
set APPDATA=C:\Users\user_name\AppData\Roaming
```

You can use the following command to identify the path that *APPDATA* is set to:

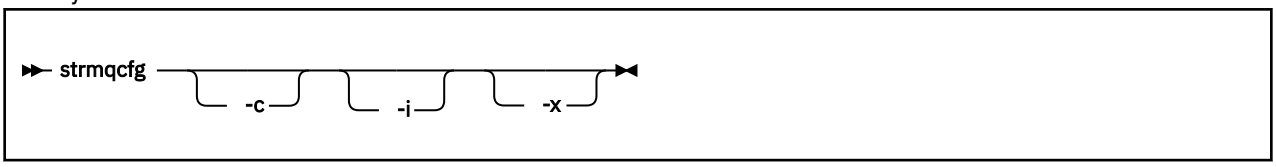
```
set APPDATA
```

On Linux, to start MQ Explorer successfully, you must be able to write a file to your home directory, and the home directory must exist.

Note: The preferred way to start MQ Explorer on Windows and Linux systems is by using the system menu, or the `MQExplorer` executable file.

Syntax

The syntax of this command follows:



Optional parameters

-c
`-clean` is passed to Eclipse. This parameter causes Eclipse to delete any cached data used by the Eclipse runtime.

-i
`-clean -initialize` is passed to Eclipse. This parameter causes Eclipse to delete any cached data as well as discard configuration information used by the Eclipse runtime.

MQ Explorer starts briefly and then ends without displaying the user interface.

-x
Output debug messages to the console.

strmqcsv

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 `dspmq -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

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

Command	Description
endmqcsv	End a command server
dspmqcsv	Display the status of a command server

strmqsvc

Start the IBM MQ service on Windows.

Purpose

The command starts the IBM MQ service on Windows. Run the command on Windows only.

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 87
End the IBM MQ service on Windows.

strmqm

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 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.

If a queue manager has no associated installation and there is no installation of IBM WebSphere MQ 7.0.1 on the system, the **strmqm** command will associate the queue manager with the installation that issued the **strmqm** command.

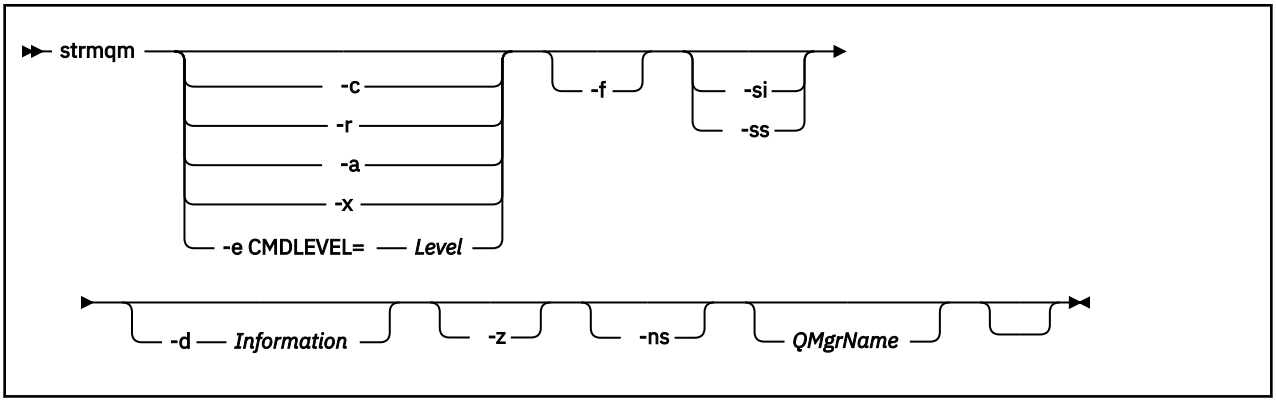
If the queue manager startup takes more than a few seconds IBM MQ shows intermittent messages detailing the startup progress.

Usage notes

V 8.0.0.8

From IBM MQ 8.0.0, Fix Pack 8, the **strmqm** command checks the syntax of the CHANNELS and SSL stanzas in the `qm.ini` file 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.

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 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 belonging 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 `crtmqm` 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 an IBM MQ 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 IBM MQ Managed File Transfer configuration directory.

-d *Information*

Specifies whether information messages are displayed. Possible values for *Information* follow:

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 `-z` .

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 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 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* 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.

In releases of IBM MQ earlier than 7.0.1, **strmqm**, with no -f option, automatically repaired missing data directories and then tried to start. This behavior has changed.

From IBM WebSphere MQ 7.0.1 onwards, 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**.

The reason for the change to the behavior of **strmqm** is that with the support for networked file storage in IBM WebSphere MQ 7.0.1, the most likely cause of missing or corrupted queue manager data directories is a configuration error that can be rectified, rather than the data directories are corrupted or irretrievably unavailable.

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 hosting the queue manager data directory.

From IBM WebSphere MQ 7.0.1, 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.

-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 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 specified on the **crtmqm** command is used.

-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 configured with service startup continue to run even after the interactive user has 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 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

Return code	Description
0	Queue manager started
3	Queue manager being created
5	Queue manager running
16	Queue manager does not exist

Return code	Description
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
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.
100	Log location invalid
119	User not authorized to start the queue manager

Examples

The following command starts the queue manager `account`:

```
strmqm account
```

Related commands

Command	Description
“crtmqm” on page 25	Create a queue manager
“dltmqm” on page 34	Delete a queue manager
“dspmqver (display version information)” on page 77	Display MQ version information
“endmqm” on page 84	End a queue manager

strmqtrc

Enable trace at a specified level of detail, or report the level of tracing in effect.

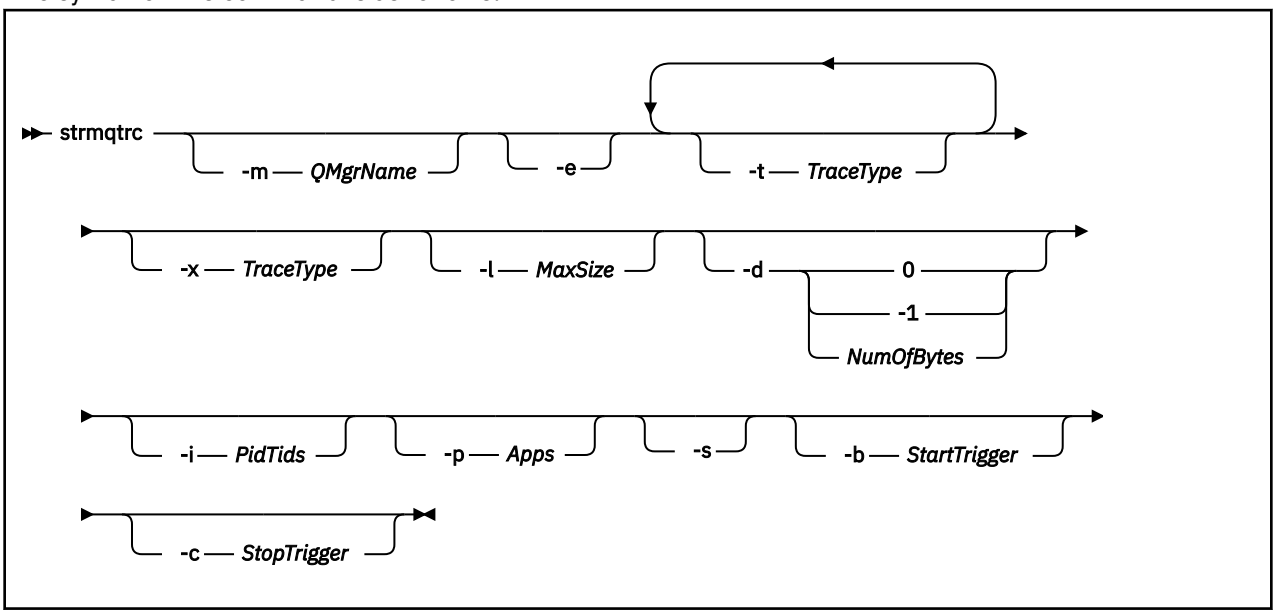
Purpose

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 using the `dspmq -o installation` command. This does not apply to a client product (for example, HP Integrity NonStop Server) because there are no queue managers from which to request direct output.

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 enqmqtrc 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.

For the IBM MQ client on HP Integrity NonStop Server, you must direct your trace commands to specific processors. For example, if your client is running on processor 2 and your shell is on processor 1, initiating

trace with **strmqtrc <options>** does not trace the client. In this case, `run -cpu=2 strmqtrc` is required.

Optional parameters

-m *QMgrName*
The name of the queue manager to trace. This parameter applies to server products only.

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 flag, 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 flag with the -m flag, -i flag, the -p flag, the -c flag, or the -b flag. If you try to use the -e flag with the -m flag, the -i flag, the -p flag, the -c flag, or the -b flag, 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 flag. You can include any number of -t flags, if each has a valid trace type associated with it.

It is not an error to specify the same trace type on multiple -t flags.

Table 39. TraceType parameter values.

Value	Description
all	Output data for every trace point in the system (the default). The all parameter activates tracing at default detail level.

Table 39. TraceType parameter values. (continued)

Value	Description
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.
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.

Table 39. TraceType parameter values. (continued)	
Value	**Description**
splflows	Output data for trace points associated with entry and exit data for functions that use a security policy (AMS) operation.
soap	Output data for trace points associated with IBM MQ Transport for SOAP.
ssl	Output data associated with using IBM Global Security Kit (GSKit) to enable Secure Sockets Layer (SSL) 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 flag.

You can use the -x flag 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 flag. You can include any number of -x flags, if each has a valid *tracetype* associated with it.

-l *MaxSize*
The maximum size of a trace file (AMQ*ppppp*.*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 AMQ*ppppp*.*qq*.TRS and a new AMQ*ppppp*.*qq*.TRC file is started. If a previous copy of an AMQ*ppppp*.*qq*.TRS file exists, it is deleted.

The highest value that *MaxSize* can be set to is 2048 MB.

-d 0
Trace no user data.

-d -1 or all
Trace all user data.

-d *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 flag with the -e flag. If you try to use the -p flag with the -e flag, 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 flag with the -e flag. If you try to use the -b flag with the -e flag, then an error message is issued. This parameter must only be used under the guidance of IBM Service personnel.

Start_Trigger	Effect
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.

Stop_Trigger	Effect
FDC=comma-separated list of FDC probe IDs.	Turns tracing off when any FDCs with the specified FDC probe IDs are generated.
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.

Return codes

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

This command enables tracing of processing flow from common services and the local queue manager for a queue manager called QM1 in IBM MQ for UNIX systems. Trace data is generated at the default level of detail.

```
strmqtrc -m QM1 -t csflows -t lqmflows -t parms
```

This command disables tracing of SSL activity on a queue manager called QM1. Other trace data is generated at the parms level of detail.

```
strmqtrc -m QM1 -x ssl -t parms
```

This command enables high-detail tracing of the processing flow for all components:

```
strmqtrc -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 flags for individual and multiple invocations of **strmqtrc**

The following examples use the -p and -m flags 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 enqmqtrc 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
     ```

HPE NonStop V8.1 This command enables tracing of all queue manager processes and applications running on and connected with queue manager QM1 on CPU 3:

```
run -cpu=3 strmqtrc -m QM1
```

If you want to trace all queue manager processes on all CPUs, you can run **strmqtrc** on all CPUs available on your system.

Example of enabling dynamic tracing of LDAP client library code shipped with IBM MQ

V 8.0.0.14

From IBM MQ 8.0.0, Fix Pack 14, 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

Command	Description
dspmqtrc	Display formatted trace output

Command	Description
endmqtrc	End trace

Comparing command sets

The tables in this section compare the facilities available from the different administration command sets, and state whether you can perform each function from within the IBM MQ Explorer.

Note: The following tables do not apply to IBM MQ for z/OS or IBM MQ for IBM i. z/OS For information on how to use PCF commands on z/OS and on IBM i, see Introduction to Programmable Command Formats. IBM i For information on how to use MQSC commands on z/OS and on IBM i, see Script (MQSC) Commands.

Queue manager commands

A table of queue manager commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 40. Queue manager commands

Description	PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Change Queue Manager	Change Queue Manager	ALTER QMGR	No equivalent	Yes
Create Queue Manager	No equivalent	No equivalent	`crtmqm`	Yes
Delete Queue Manager	No equivalent	No equivalent	`dltmqm`	Yes
Inquire Queue Manager	Inquire Queue Manager	DISPLAY QMGR	No equivalent	Yes
Inquire Queue Manager Status	Inquire Queue Manager Status	DISPLAY QMSTATUS	`dspmq`	Yes
Ping Queue Manager	Ping Queue Manager	PING QMGR	No equivalent	No
Refresh Queue Manager	Refresh Queue Manager	REFRESH QMGR	No equivalent	Yes
Reset Queue Manager	Reset Queue Manager	RESET QMGR	No equivalent	No
Start Queue Manager	No equivalent	No equivalent	`strmqm`	Yes
Stop Queue Manager	No equivalent	No equivalent	`endmqm`	Yes

Command server commands

A table of command server commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 41. Commands for command server administration

Description	PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Display command server	Inquire Queue Manager Status	DISPLAY QMSTATUS	`dspmqcsv`	Yes
Start command server	Change Queue Manager	ALTER QMGR	`strmqcsv`	Yes
Stop command server	No equivalent	No equivalent	`endmqcsv`	Yes

Authority commands

A table of authority commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 42. Commands for authority administration

PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Delete authority record	DELETE AUTHREC	`setmqaut`	Yes
Inquire authority records	DISPLAY AUTHREC	`dmpmqaut`	Yes
Inquire entity authority	DISPLAY ENTAUTH	`dspmqaut`	Yes
Refresh Security	REFRESH SECURITY	No equivalent	Yes
Set authority record	SET AUTHREC	`setmqaut`	Yes

Cluster commands

A table of cluster commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 43. Cluster commands

PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Inquire Cluster Queue Manager	DISPLAY CLUSQMGR	No equivalent	Yes
Refresh Cluster	REFRESH CLUSTER	No equivalent	Yes
Reset Cluster	RESET CLUSTER	No equivalent	No
Resume Queue Manager Cluster	RESUME QMGR	No equivalent	Yes
Suspend Queue Manager Cluster	SUSPEND QMGR	No equivalent	Yes

Authentication information commands

A table of authentication information commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 44. Authentication information commands

PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Change Authentication Information Object	ALTER AUTHINFO	No equivalent	Yes
Copy Authentication Information Object	DEFINE AUTHINFO(x) LIKE(y)	No equivalent	Yes
Create Authentication Information Object	DEFINE AUTHINFO	No equivalent	Yes
Delete Authentication Information Object	DELETE AUTHINFO	No equivalent	Yes
Inquire Authentication Information Object	DISPLAY AUTHINFO	No equivalent	Yes

Channel commands

A table of channel commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and MQ Explorer equivalents, if available.

Table 45. Channel commands

PCF command	MQSC command	Control command	MQ Explorer equivalent?
Change Channel	ALTER CHANNEL	No equivalent	Yes
Copy Channel	DEFINE CHANNEL(x) LIKE(y)	No equivalent	Yes
Create Channel	DEFINE CHANNEL	No equivalent	Yes
Delete Channel	DELETE CHANNEL	No equivalent	Yes
Inquire Channel	DISPLAY CHANNEL	No equivalent	Yes
Inquire Channel Names	DISPLAY CHANNEL	No equivalent	Yes
Inquire Channel Status	DISPLAY CHSTATUS	No equivalent	Yes
Ping Channel	PING CHANNEL	No equivalent	Yes
Purge Channel	PURGE CHANNEL	No equivalent	Yes
Reset Channel	RESET CHANNEL	No equivalent	Yes
Resolve Channel	RESOLVE CHANNEL	No equivalent	Yes
Start Channel	START CHANNEL	`runmqchl`	Yes
Start Channel Initiator	START CHINIT	`runmqchi`	No
Stop Channel	STOP CHANNEL	No equivalent	Yes

Listener commands

A table of listener commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 46. Listener commands

PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Change Listener	ALTER LISTENER	No equivalent	Yes
Copy Listener	DEFINE LISTENER(x) LIKE(y)	No equivalent	Yes
Create Listener	DEFINE LISTENER	No equivalent	Yes
Delete Listener	DELETE LISTENER	No equivalent	Yes
Inquire Listener	DISPLAY LISTENER	No equivalent	Yes
Inquire Listener Status	DISPLAY LSSTATUS	No equivalent	Yes
Start Channel Listener	START LISTENER "1" on page 165	`runmqlsr`	Yes
Stop Listener	STOP LISTENER	`endmqlsr` "2" on page 165	Yes

Notes:

1. Used with listener objects only
2. Stops all active listeners

Namelist commands

A table of namelist commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 47. Namelist commands

PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Change Namelist	ALTER NAMELIST	No equivalent	Yes
Copy Namelist	DEFINE NAMELIST(x) LIKE(y)	No equivalent	Yes
Create Namelist	DEFINE NAMELIST	No equivalent	Yes
Delete Namelist	DELETE NAMELIST	No equivalent	Yes
Inquire Namelist	DISPLAY NAMELIST	No equivalent	Yes
Inquire Namelist Names	DISPLAY NAMELIST	No equivalent	Yes

Process commands

A table of process commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 48. Process commands

PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Change Process	ALTER PROCESS	No equivalent	Yes
Copy Process	DEFINE PROCESS(x) LIKE(y)	No equivalent	Yes
Create Process	DEFINE PROCESS	No equivalent	Yes
Delete Process	DELETE PROCESS	No equivalent	Yes
Inquire Process	DISPLAY PROCESS	No equivalent	Yes
Inquire Process Names	DISPLAY PROCESS	No equivalent	Yes

Queue commands

A table of queue commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 49. Queue commands

PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
Change Queue	ALTER QLOCAL ALTER QALIAS ALTER QMODEL ALTER QREMOTE	No equivalent	Yes
Clear Queue	CLEAR QLOCAL	No equivalent	Yes
Copy Queue	DEFINE QLOCAL(x) LIKE(y) DEFINE QALIAS(x) LIKE(y) DEFINE QMODEL(x) LIKE(y) DEFINE QREMOTE(x) LIKE(y)	No equivalent	Yes
Create Queue	DEFINE QLOCAL DEFINE QALIAS DEFINE QMODEL DEFINE QREMOTE	No equivalent	Yes
Delete Queue	DELETE QLOCAL DELETE QALIAS DELETE QMODEL DELETE QREMOTE	No equivalent	Yes
Inquire Queue	DISPLAY QUEUE	No equivalent	Yes
Inquire Queue Names	DISPLAY QUEUE	No equivalent	Yes

Table 49. Queue commands (continued)			
PCF command	**MQSC command**	**Control command**	**IBM MQ Explorer equivalent?**
Inquire Queue Status	DISPLAY QSTATUS	No equivalent	Yes
Reset Queue Statistics	No equivalent	No equivalent	No

Service commands

A table of service commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 50. Service commands			
PCF command	**MQSC command**	**Control command**	**IBM MQ Explorer equivalent?**
Change Service	ALTER SERVICE	No equivalent	Yes
Copy Service	DEFINE SERVICE(x) LIKE(y)	No equivalent	Yes
Create Service	DEFINE SERVICE	No equivalent	Yes
Delete Service	DELETE SERVICE	No equivalent	Yes
Inquire Service	DISPLAY SERVICE	No equivalent	Yes
Inquire Service Status	DISPLAY SVSTATUS	No equivalent	Yes
Start Service	START SERVICE	No equivalent	Yes
Stop Service	STOP SERVICE	No equivalent	Yes

Other commands

A table of other commands, showing the command description, and its PCF command, MQSC command, control command equivalents, and IBM MQ Explorer equivalents, if available.

Table 51. Other commands				
Description	**PCF command**	**MQSC command**	**Control command**	**IBM MQ Explorer equivalent?**
Create conversion exit	No equivalent	No equivalent	`crtmqcvx`	No
Display files used by objects	No equivalent	No equivalent	`dspmqfls`	No
Display formatted trace	No equivalent	No equivalent	`dspmqtrc` "1" on page 168	No
Display version information	No equivalent	No equivalent	`dspmqver`	No
Display transactions	No equivalent	No equivalent	`dspmqtrn`	No
Dump log	No equivalent	No equivalent	`dmpmqlog`	No
Dump MQ Configuration	No equivalent	No equivalent	`dmpmqcfg`	No

Table 51. Other commands (continued)

Description	PCF command	MQSC command	Control command	IBM MQ Explorer equivalent?
End trace	No equivalent	No equivalent	`endmqtrc`	Yes
Escape	Escape	No equivalent	No equivalent	No
Record media image	No equivalent	No equivalent	`rcdmqimg`	No
Re-create media object	No equivalent	No equivalent	`rcrmqobj`	No
Resolve transactions	No equivalent	No equivalent	`rsvmqtrn`	No
Run client trigger monitor	No equivalent	No equivalent	`runmqtmc`	No
Run dead-letter queue handler	No equivalent	No equivalent	`runmqdlq`	No
Run MQSC commands	No equivalent	No equivalent	`runmqsc`	No
Run trigger monitor	No equivalent	No equivalent	`runmqtrm`	No
Set service connection points	No equivalent	No equivalent	`setmqscp` "2" on page 168	No
Start IBM MQ trace	No equivalent	No equivalent	`strmqtrc`	Yes
IBM MQ Services control	No equivalent	No equivalent	`amqmdain` "2" on page 168	No

Notes:

1. Not supported on IBM MQ for Windows.
2. Supported by IBM MQ for Windows only.

Managing keys and certificates

Use the `runmqckm` command (Windows and UNIX systems) to manage keys, certificates, and certificate requests.

The runmqckm command

The `runmqckm` command is available on Windows and UNIX systems.

The `runmqckm` command provides functions that are similar to those of iKeyman, described in Security.

Use the `runmqckm` command to do the following:

- Create the type of CMS key database files that IBM MQ requires
- Create certificate requests
- Import personal certificates
- Import CA certificates
- Manage self-signed certificates

Preparing to use the runmqckm and runmqakm commands

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a

64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

To run the **runmqckm** command line interfaces, ensure that the systems environment variables are correctly configured by running the **setmqenv** command.

runmqckm, and runmqakm commands

This section describes the runmqckm, and runmqakm commands according to the object of the command.

An overview of the main differences between the two commands:

- **runmqakm**
  - Supports the creation of certificates and certificate requests with Elliptic Curve public keys whereas the **runmqckm** command does not.
  - Supports stronger encryption of the key repository file than the **runmqckm** command through the **-strong** parameter.
  - Has been certified as FIPS 140-2 compliant, and can be configured to operate in a FIPS compliant manner, using the **-fips** parameter, unlike the **runmqckm** command.
- **runmqckm** supports the JKS and JCEKS key repository file formats whereas the **runmqakm** command does not.

Each command specifies at least one *object*. Commands for PKCS #11 device operations might specify additional objects. Commands for key database, certificate, and certificate request objects also specify an *action*. The object can be one of the following:

-keydb
Actions apply to a key database

-cert
Actions apply to a certificate

-certreq
Actions apply to a certificate request

-help
Displays help

-version
Displays version information

The following subtopics describe the actions that you can take on key database, certificate, and certificate request objects; See “runmqckm and runmqakm options” on page 177 for a description of the options on these commands.

Commands for a CMS key database only

You can use the **runmqckm**, and **runmqakm** commands to manage keys and certificates for a CMS key database.

-keydb -changepw
Change the password for a CMS key database:

```
-keydb -changepw -db filename -pw password -new_pw new_password

-stash
```

-keydb -create
Create a CMS key database:

```
-keydb -create -db filename -pw password
-type cms -expire days -stash
```

-keydb -stashpw
Stash the password of a CMS key database into a file:

```
-keydb -stashpw -db filename -pw password
```

-cert -getdefault

Note: The default certificate is not supported by IBM MQ 8.0. You should use certificate label configuration as described in Digital certificate labels, understanding the requirements.

Get the default personal certificate:

```
-cert -getdefault -db filename -pw password
```

-cert -modify
Modify a certificate.

Note: Currently, the only field that can be modified is the Certificate Trust field.

```
-cert -modify -db filename -pw password -label label
-trust enable | disable
```

-cert -setdefault

Note: The default certificate is not supported by IBM MQ 8.0. You should use certificate label configuration as described in Digital certificate labels, understanding the requirements.

Set the default personal certificate:

```
-cert -setdefault -db filename
-pw password -label label
```

Command for CMS or PKCS #12 key databases

You can use the runmqckm, and runmqakm commands to manage keys and certificates for a CMS key database or PKCS #12 key database.

Note: IBM MQ does not support SHA-3 or SHA-5 algorithms. You can use the digital signature algorithm names SHA384WithRSA and SHA512WithRSA because both algorithms are members of the SHA-2 family.

The digital signature algorithm names SHA3WithRSA and SHA5WithRSA are deprecated because they are an abbreviated form of SHA384WithRSA and SHA512WithRSA respectively.

-keydb -changepw
Change the password for a key database:

```
-keydb -changepw -db filename -pw password
-new_pw new_password -expire days
```

-keydb -convert
convert the key database from one format to another:

```
-keydb -convert -db filename -pw password
-old_format cms | pkcs12 -new_format cms
```

-keydb -create
Create a key database:

```
-keydb -create -db filename -pw password
-type cms | pkcs12
```

-keydb -delete
Delete a key database:

```
-keydb -delete -db filename -pw password
```

-keydb -list

List currently-supported types of key database:

```
-keydb -list
```

-cert -add

Add a certificate from a file into a key database:

```
-cert -add -db filename -pw password -label label
-file filename -format ascii | binary
```

-cert -create

Create a self-signed certificate:

```
-cert -create -db filename -pw password -label label
 -dn distinguished_name
    -size 1024 | 512 -x509version 3 | 1  | 2
    -expire days
    -sig_alg MD2_WITH_RSA | MD2WithRSA |
             MD5_WITH_RSA | MD5WithRSA |
             SHA1WithDSA | SHA1WithRSA |
             SHA256_WITH_RSA | SHA256WithRSA |
             SHA2WithRSA | SHA384_WITH_RSA |
             SHA384WithRSA |  SHA512_WITH_RSA |
             SHA512WithRSA | SHA_WITH_DSA  |
             SHA_WITH_RSA | SHAWithDSA | SHAWithRSA
```

-cert -delete

Delete a certificate:

```
-cert -delete -db filename -pw password -label label
```

-cert -details

List the detailed information for a specific certificate:

```
-cert -details -db filename -pw password -label label
```

-cert -export

Export a personal certificate and its associated private key from a key database into a PKCS #12 file, or to another key database:

```
-cert -export -db filename -pw password -label label
 -type cms | pkcs12
    -target filename -target_pw password -target_type
cms | pkcs12
```

-cert -extract

Extract a certificate from a key database:

```
-cert -extract -db filename -pw password -label label
 -target filename
    -format ascii | binary
```

-cert -import

Import a personal certificate from a key database:

```
 -cert -import -file filename -pw password -type
pkcs12 -target filename
        -target_pw password -target_type cms -label
label
```

The `-label` option is required and specifies the label of the certificate that is to be imported from the source key database.

The `-new_label` option is optional and allows the imported certificate to be given a different label in the target key database from the label in the source database.

-cert -list
List all certificates in a key database:

```
-cert -list all | personal | CA
    -db filename -pw password
```

-cert -receive
Receive a certificate from a file:

```
-cert -receive -file filename -db filename -pw password

    -format ascii | binary -default_cert yes | no
```

-cert -sign
Sign a certificate:

```
-cert -sign -db filename -file filename -pw password
   -label label -target filename
   -format ascii | binary -expire days
   -sig_alg MD2_WITH_RSA | MD2WithRSA | MD5_WITH_RSA |
            MD5WithRSA | SHA1WithDSA | SHA1WithRSA |
            SHA256_WITH_RSA | SHA256WithRSA |
            SHA2WithRSA | SHA384_WITH_RSA |
            SHA384WithRSA |  SHA512_WITH_RSA |
            SHA512WithRSA | SHA_WITH_DSA  |
            SHA_WITH_RSA | SHAWithDSA |
            SHAWithRSA
```

-certreq -create
Create a certificate request:

```
-certreq -create -db filename -pw password
    -label label -dn distinguished_name
    -size 1024 | 512 -file filename
    -sig_alg MD2_WITH_RSA | MD2WithRSA |
             MD5_WITH_RSA | MD5WithRSA |
             SHA1WithDSA | SHA1WithRSA |
             SHA256_WITH_RSA | SHA256WithRSA |
             SHA2WithRSA | SHA384_WITH_RSA |
             SHA384WithRSA | SHA512_WITH_RSA |
             SHA512WithRSA | SHA_WITH_DSA |
             SHA_WITH_RSA | SHAWithDSA |
             SHAWithRSA
```

-certreq -delete
Delete a certificate request:

```
-certreq -delete -db filename -pw password -label label
```

-certreq -details
List the detailed information of a specific certificate request:

```
-certreq -details -db filename -pw password -label label
```

List the detailed information about a certificate request and show the full certificate request:

```
-certreq -details -showOID -db filename
    -pw password -label label
```

-certreq -extract
Extract a certificate request from a certificate request database into a file:

```
-certreq -extract -db filename -pw password
     -label label -target filename
```

-certreq -list
List all certificate requests in the certificate request database:

```
-certreq -list -db filename -pw password
```

-certreq -recreate
Re-create a certificate request:

```
-certreq -recreate -db filename -pw password
    -label label -target filename
```

Commands for cryptographic device operations

You can use the runmqckm, and runmqakm commands to manage keys and certificates for cryptographic device operations.

Note: IBM MQ does not support SHA-3 or SHA-5 algorithms. You can use the digital signature algorithm names SHA384WithRSA and SHA512WithRSA because both algorithms are members of the SHA-2 family.

The digital signature algorithm names SHA3WithRSA and SHA5WithRSA are deprecated because they are an abbreviated form of SHA384WithRSA and SHA512WithRSA respectively.

-keydb -changepw
Change the password for a cryptographic device:

```
-keydb -changepw -crypto module_name -tokenlabel token_label
    -pw password -new_pw new_password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-keydb -list
List currently-supported types of key database:

```
-keydb -list
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-cert -add
Add a certificate from a file to a cryptographic device:

```
-cert -add -crypto module_name -tokenlabel token_label
    -pw password -label label -file filename -format
ascii | binary
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-cert -create
Create a self-signed certificate on a cryptographic device:

```
-cert -create -crypto module_name -tokenlabel token_label

    -pw password -label label -dn distinguished_name
    -size 1024 | 512
    -x509version 3 | 1 | 2
    -default_cert no | yes -expire days
    -sig_alg MD2_WITH_RSA | MD2WithRSA |
             MD5_WITH_RSA | MD5WithRSA |
             SHA1WithDSA | SHA1WithRSA |
             SHA256_WITH_RSA | SHA256WithRSA |
             SHA2WithRSA | SHA384_WITH_RSA |
```

```
            SHA384WithRSA | SHA512_WITH_RSA |
            SHA512WithRSA | SHA_WITH_DSA |
            SHA_WITH_RSA | SHAWithDSA |
            SHAWithRSA
```

Note: You cannot import a certificate containing multiple OU (organizational unit) attributes in the distinguished name.

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-cert -delete

Delete a certificate on a cryptographic device:

```
-cert -delete -crypto module_name -tokenlabel token_label
    -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-cert -details

List the detailed information for a specific certificate on a cryptographic device:

```
-cert -details -crypto module_name -tokenlabel token_label

    -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

List the detailed information and show the full certificate for a specific certificate on a cryptographic device:

```
-cert -details -showOID -crypto module_name -tokenlabel token_label
    -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-cert -extract

Extract a certificate from a key database:

```
-cert -extract -crypto module_name -tokenlabel token_label
    -pw password -label label -target filename
 -format ascii | binary
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-cert -import

Import a certificate to a cryptographic device with secondary key database support:

```
-cert -import -db filename -pw password -label label
 -type cms
    -crypto module_name -tokenlabel token_label -pw password
    -secondaryDB filename -secondaryDBpw password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

```
-cert -import -db filename -pw password -label label
 -type cms
    -crypto module_name -tokenlabel token_label -pw password
    -secondaryDB filename -secondaryDBpw password -fips
```

Import a PKCS #12 certificate to a cryptographic device with secondary key database support:

```
-cert -import -file filename -pw password -type pkcs12
    -crypto module_name -tokenlabel token_label -pw password
    -secondaryDB filename -secondaryDBpw password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

```
-cert -import -file filename -pw password -type pkcs12
    -crypto module_name -tokenlabel token_label -pw password
    -secondaryDB filename -secondaryDBpw password -fips
```

Note: You cannot import a certificate containing multiple OU (organizational unit) attributes in the distinguished name.

-cert -list

List all certificates on a cryptographic device:

```
-cert -list all | personal | CA
    -crypto module_name -tokenlabel token_label -pw password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-cert -receive

Receive a certificate from a file to a cryptographic device with secondary key database support:

```
-cert -receive -file filename -crypto module_name
-tokenlabel token_label
    -pw password  -default_cert yes | no
    -secondaryDB filename -secondaryDBpw password
    -format ascii | binary
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

Using the **runmqakm** command:

-certreq -create
Create a certificate request on a cryptographic device:

```
-certreq -create -crypto module_name -tokenlabel token_label

    -pw password -label label -dn distinguished_name
    -size 1024 | 512 -file filename
    -sig_alg MD2_WITH_RSA | MD2WithRSA |MD5_WITH_RSA |
             MD5WithRSA |SHA1WithDSA | SHA1WithRSA |
             SHA256_WITH_RSA | SHA256WithRSA
             SHA2WithRSA | SHA384_WITH_RSA |
             SHA384WithRSA | SHA512_WITH_RSA |
             SHA512WithRSA | SHA_WITH_DSA  |
             SHA_WITH_RSA | SHAWithDSA |
             SHAWithRSA
```

Note: You cannot import a certificate containing multiple OU (organizational unit) attributes in the distinguished name.

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-certreq -delete
Delete a certificate request from a cryptographic device:

```
-certreq -delete -crypto module_name -tokenlabel token_label

    -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-certreq -details
List the detailed information of a specific certificate request on a cryptographic device:

```
-certreq -details -crypto module_name -tokenlabel token_label

    -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

List the detailed information about a certificate request and show the full certificate request on a cryptographic device:

```
-certreq -details -showOID -crypto module_name -tokenlabel token_label
    -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-certreq -extract
Extract a certificate request from a certificate request database on a cryptographic device into a file:

```
-certreq -extract -crypto module_name -tokenlabel token_label
    -pw password -label label -target filename
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

-certreq -list

List all certificate requests in the certificate request database on a cryptographic device:

```
-certreq -list -crypto module_name -tokenlabel token_label
-pw password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 64-bit programs. External modules required for PKCS #11 support will be loaded into a 64-bit process, therefore you must have a 64-bit PKCS #11 library installed for the administration of cryptographic hardware. The Windows and Linux x86 32-bit platforms are the only exceptions, as the iKeyman and iKeycmd programs are 32-bit on those platforms.

runmqckm and runmqakm options

A table of the **runmqckm** and **runmqakm** options that can be present on the command line.

Note: IBM MQ does not support SHA-3 or SHA-5 algorithms. You can use the digital signature algorithm names SHA384WithRSA and SHA512WithRSA because both algorithms are members of the SHA-2 family.

The digital signature algorithm names SHA3WithRSA and SHA5WithRSA are deprecated because they are an abbreviated form of SHA384WithRSA and SHA512WithRSA respectively.

The meaning of an option can depend on the object and action specified in the command.

*Table 52. Options that can be used with **runmqckm** and **runmqakm***

Option	Description
`-create`	Option to create a key database.
`-crypto`	Name of the module to manage a PKCS #11 cryptographic device. The value after `-crypto` is optional if you specify the module name in the properties file. If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64-bit on the HP Itanium platform.
`-db`	Fully qualified path name of a key database.
`-default_cert`	Sets a certificate as the default certificate. The value can be `yes` or `no`. The default is `no`.
`-dn`	X.500 distinguished name. The value is a string enclosed in double quotation marks, for example `"CN=John Smith,O=IBM,OU=Test,C=GB"`. Note that the CN, O, and C attributes are required.
`-encryption`	Strength of encryption used in certificate export command. The value can be `strong` or `weak`. The default is `strong`.

*Table 52. Options that can be used with **runmqckm** and **runmqakm** (continued)*	
Option	**Description**
`-expire`	Expiration time in days of either a certificate or a database password. The default is 365 days for a certificate password. There is no default time for a database password: use the `-expire` option to set a database password expiration time explicitly.
`-file`	File name of a certificate or certificate request.
`-fips`	specifies that the command is run in FIPS mode. When in FIPS mode, the IBM Crypto for C (ICC) component uses algorithms that have been FIPS 140-2 validated. If the ICC component does not initialize in FIPS mode, the **runmqakm** command fails.
`-format`	Format of a certificate. The value can be `ascii` for Base64_encoded ASCII or `binary` for Binary DER data. The default is `ascii`.
`-label`	Label attached to a certificate or certificate request. If the certificate is a personal certificate used to identify an IBM MQ client application or queue manager, the label must correspond to the IBM MQ certificate label (CERTLABL) setting, for more information, see Digital certificate labels, understanding the requirements.
`-new_format`	New format of key database.
`-new_label`	Used on a certificate import command, this option allows a certificate to be imported with a different label from the label it had in the source key database. If the certificate is a personal certificate used to identify an IBM MQ client application or queue manager, the label must correspond to the IBM MQ certificate label (CERTLABL) setting, for more information, see Digital certificate labels, understanding the requirements.
`-new_pw`	New database password.
`-old_format`	Old format of key database.
`-pw`	Password for the key database or PKCS #12 file.
`-secondaryDB`	Name of a secondary key database for PKCS #11 device operations.
`-secondaryDBpw`	Password for the secondary key database for PKCS #11 device operations.
`-showOID`	Displays the full certificate or certificate request.

*Table 52. Options that can be used with **runmqckm** and **runmqakm** (continued)*	
Option	**Description**
-sig_alg	The hashing algorithm used during the creation of a certificate request, a self-signed certificate, or the signing of a certificate. This hashing algorithm is used to create the signature associated with the newly-created certificate or certificate request. For **runmqckm**, the value can be MD2_WITH_RSA, MD2WithRSA, MD5_WITH_RSA, MD5WithRSA, SHA1WithDSA, SHA1WithRSA, SHA256_WITH_RSA, SHA256WithRSA, SHA2WithRSA, SHA384_WITH_RSA, SHA384WithRSA, SHA512_WITH_RSA, SHA512WithRSA, SHA_WITH_DSA, SHA_WITH_RSA, SHAWithDSA, or SHAWithRSA. The default value is SHA1WithRSA. For **runmqakm**, the value can be md5, MD5_WITH_RSA, MD5WithRSA, SHA_WITH_DSA, SHA_WITH_RSA, sha1, SHA1WithDSA, SHA1WithECDSA, SHA1WithRSA, sha224, SHA224_WITH_RSA, SHA224WithDSA, SHA224WithECDSA, SHA224WithRSA, sha256, SHA256_WITH_RSA, SHA256WithDSA, SHA256WithECDSA, SHA256WithRSA, SHA2WithRSA, sha384, SHA384_WITH_RSA, SHA384WithECDSA, SHA384WithRSA, sha512, SHA512_WITH_RSA, SHA512WithECDSA, SHA512WithRSA, SHAWithDSA, SHAWithRSA, EC_ecdsa_with_SHA1, EC_ecdsa_with_SHA224, EC_ecdsa_with_SHA256, EC_ecdsa_with_SHA384, or EC_ecdsa_with_SHA512. The default value is SHA1WithRSA.
-size	Key size. For **runmqckm**, the value can be 512, 1024, or 2048. The default value is 1024 bits. For **runmqakm**, the value depends upon the signature algorithm: • For RSA signature algorithms (the default algorithm used if no -sig_alg is specified), the value can be 512, 1024, 2048, or 4096. An RSA key size of 512 bits is not permitted if the **-fips** parameter is enabled. The default RSA key size is 1024 bits. • For Elliptic Curve algorithms, the value can be 256, 384, or 512. The default Elliptic Curve key size depends upon the signature algorithm. For SHA256, it is 256; for SHA384, it is 384; and for SHA512, it is 512.
-stash	Stash the key database password to a file. **Note: -stash** is valid on **-keydb** -create commands to tell **runmqckm**/ **runmqakm** to create a stash file containing the password. Issuing the command $ runmqakm -help lists the high-level help parameters only.
-stashed	Indicates password for the key database is in a stash file. **Note:** The **-stashed** option is valid on calls apart from the **-keydb** -create commands. If you do not specify this option, you have to supply the password using **-pw**. In addition, only when you instruct the command what kind of action you are performing does the detailed help showing **-stashed** appear.
-target	Destination file or database.
-target_pw	Password for the key database if -target specifies a key database.

*Table 52. Options that can be used with **runmqckm** and **runmqakm** (continued)*	
Option	**Description**
`-target_type`	Type of database specified by `-target` operand. See `-type` option for permitted values.
`-tokenLabel`	Label of a PKCS #11 cryptographic device.
`-trust`	Trust status of a CA certificate. The value can be `enable` or `disable`. The default is `enable`.
`-type`	Type of database. The value can be any of the following values: • `cms` for a CMS key database • `pkcs12` for a PKCS #12 file.
`-x509version`	Version of X.509 certificate to create. The value can be 1, 2, or 3. The default is 3.

V 8.0.0.4
V 8.0.0.4
`-rfc3339`
	Use this parameter to output the date in the RFC 3339 format for the `runmqakm -cert -details` command, which is of the following format:
	`Not Before : 2015-08-26T08:53:37Z`
	`Not After : 2016-08-26T08:53:37Z`
	Note that the **`-rfc3339`** parameter has to appear in the command after the additional parameters:
	`runmqakm -cert -details -db exampleDB -stashed -label certficateLabel -rfc3339`

Note: Properties provided with GSKit relating to symmetric-key encryption `-seckey` option in the 'runmqckm' utility are ignored and not supported by IBM MQ.

runmqakm error codes

A table of the numeric error codes issued by runmqakm, and what they mean.

Error code	Error Message
0	Success
1	Unknown error occurred
2	An ASN.1 encoding/decoding error occurred.
3	An error occurred while initializing ASN.1 encoder/decoder.
4	An ASN.1 encoding/decoding error occurred because of an out-of-range index or non-existent optional field.
5	A database error occurred.
6	An error occurred while opening the database file, check for file existence and permission.
7	An error occurred while re-opening the database file.
8	Database creation failed.
9	The database already exists.

Error code	Error Message
10	An error occurred while deleting the database file.
11	The database could not be opened.
12	An error occurred while reading the database file.
13	An error occurred while writing data to the database file.
14	A database validation error occurred.
15	An invalid database version was encountered.
16	An invalid database password was encountered.
17	An invalid database file type was encountered.
18	The specified database has been corrupted.
19	An invalid password was provided or the key database has been tampered with or corrupted.
20	A database key entry integrity error occurred.
21	A duplicate certificate already exists in the database.
22	A duplicate key already exists in the database (Record ID).
23	A certificate with the same label already existed in the key database.
24	A duplicate key already exists in the database (Signature).
25	A duplicate key already exists in the database (Unsigned Certificate).
26	A duplicate key already exists in the database (Issuer and Serial Number).
27	A duplicate key already exists in the database (Subject Public Key Info).
28	A duplicate key already exists in the database (Unsigned CRL).
29	The label has been used in the database.
30	A password encryption error occurred.
31	An LDAP related error occurred. (LDAP is not supported by this program)
32	A cryptographic error occurred.
33	An encryption/decryption error occurred.
34	An invalid cryptographic algorithm was found.
35	An error occurred while signing data.
36	An error occurred while verifying data.
37	An error occurred while computing digest of data.

Error code	Error Message
38	An invalid cryptographic parameter was found.
39	An unsupported cryptographic algorithm was encountered.
40	The specified input size is greater than the supported modulus size.
41	An unsupported modulus size was found.
42	A database validation error occurred.
43	Key entry validation failed.
44	A duplicate extension field exists.
45	The version of the key is wrong.
46	A required extension field does not exist.
47	The validity period does not include today or does not fall within its issuer's validity period
48	The validity period does not include today or does not fall within its issuer's validity period.
49	An error occurred while validating private key usage extension.
50	The issuer of the key was not found.
51	A required certificate extension is missing.
52	An invalid basic constraint extension was found.
53	The key signature validation failed.
54	The root key of the key is not trusted.
55	The key has been revoked.
56	An error occurred while validating authority key identifier extension.
57	An error occurred while validating private key usage extension.
58	An error occurred while validating subject alternative name extension.
59	An error occurred while validating issuer alternative name extension.
60	An error occurred while validating key usage extension.
61	An unknown critical extension was found.
62	An error occurred while validating key pair entries.
63	An error occurred while validating CRL.
64	A mutex error occurred.
65	An invalid parameter was found.

Error code	Error Message
66	A null parameter or memory allocation error was encountered.
67	Number or size is too large or too small.
68	The old password is invalid.
69	The new password is invalid.
70	The password has expired.
71	A thread related error occurred.
72	An error occurred while creating threads.
73	An error occurred while a thread was waiting to exit.
74	An I/O error occurred.
75	An error occurred while loading CMS.
76	A cryptography hardware related error occurred.
77	The library initialization routine was not successfully called.
78	The internal database handle table is corrupted.
79	A memory allocation error occurred.
80	An unrecognized option was found.
81	An error occurred while getting time information.
82	Mutex creation error occurred.
83	An error occurred while opening message catalog.
84	An error occurred while opening error message catalog
85	A null file name was found.
86	An error occurred while opening files, check for file existence and permissions.
87	An error occurred while opening files to read.
88	An error occurred while opening files to write.
89	There is no such file.
90	The file cannot be opened because of its permission setting.
91	An error occurred while writing data to files.
92	An error occurred while deleting files.
93	Invalid Base64-encoded data was found.
94	An invalid Base64 message type was found.
95	An error occurred while encoding data with Base64 encoding rule.

Error code	Error Message
96	An error occurred while decoding Base64-encoded data.
97	An error occurred while getting a distinguished name tag.
98	The required common name field is empty.
99	The required country or region name field is empty.
100	An invalid database handle was found.
101	The key database does not exist.
102	The request key pair database does not exist.
103	The password file does not exist.
104	The new password is identical to the old one.
105	No key was found in the key database.
106	No request key was found.
107	No trusted CA was found.
108	No request key was found for the certificate.
109	There is no private key in the key database.
110	There is no default key in the key database.
111	There is no private key in the key record.
112	There is no certificate in the key record.
113	There is no CRL entry.
114	An invalid key database file name was found.
115	An unrecognized private key type was found.
116	An invalid distinguished name input was found.
117	No key entry was found that has the specified key label.
118	The key label list has been corrupted.
119	The input data is not valid PKCS12 data.
120	The password is invalid or the PKCS12 data has been corrupted or been created with later version of PKCS12
121	An unrecognized key export type was found.
122	An unsupported password-based encryption algorithm was found.
123	An error occurred while converting the key ring file to a CMS key database.
124	An error occurred while converting the CMS key database to a key ring file.

Error code	Error Message
125	An error occurred while creating a certificate for the certificate request.
126	A complete issuer chain cannot be built.
127	Invalid WEBDB data was found.
128	There is no data to be written to the key ring file.
129	The number of days that you entered extends beyond the permitted validity period.
130	The password is too short; it must consist of at least {0} characters.
131	A password must contain at least one numeric digit.
132	All characters in the password are either alphabetic or numeric characters.
133	An unrecognized or unsupported signature algorithm was specified.
134	An invalid database type was encountered.
135	The specified secondary key database is in use by another PKCS#11 device.
136	No secondary key database was specified.
137	The label does not exist on the PKCS#11 device.
138	Password required to access the PKCS#11 device.
139	Password not required to access the PKCS#11 device.
140	Unable to load the cryptographic library.
141	PKCS#11 is not supported for this operation.
142	An operation on a PKCS#11 device has failed.
143	The LDAP user is not a valid user. (LDAP is not supported by this program)
144	The LDAP user is not a valid user. (LDAP is not supported by this program)
145	The LDAP query failed. (LDAP is not supported by this program)
146	An invalid certificate chain was found.
147	The root certificate is not trusted.
148	A revoked certificate was encountered.
149	A cryptographic object function failed.
150	There is no certificate revocation list data source available.
151	There is no cryptographic token available.
152	FIPS mode is not available.

Error code	Error Message
153	There is a conflict with the FIPS mode settings.
154	The password entered does not meet the minimum required strength.
200	There was a failure during initialization of the program.
201	Tokenization of the arguments passed to the runmqakm Program failed.
202	The object identified in the command is not a recognized object.
203	The action passed is not a known -keydb action.
204	The action passed is not a known -cert action.
205	The action passed is not a known -certreq action.
206	There is a tag missing for the requested command.
207	The value passed with the -version tag is not a recognized value.
208	The value passed with the -size tag is not a recognized value.
209	The value passed in with the -dn tag is not in the correct format.
210	The value passed in with the -format tag is not a recognized value.
211	There was an error associated with opening the file.
212	PKCS12 is not supported at this stage.
213	The cryptographic token you are trying to change the password for is not password protected.
214	PKCS12 is not supported at this stage.
215	The password entered does not meet the minimum required strength.
216	FIPS mode is not available.
217	The number of days you have entered as the expiry date is out of the allowed range.
218	Password strength failed the minimum requirements.
219	No Default certificate was found in the requested key database.
220	An invalid trust status was encountered.
221	An unsupported signature algorithm was encountered. At this stage only MD5 and SHA1 are supported.

Error code	Error Message
222	PCKS11 not supported for that particular operation.
223	The action passed is not a known -random action.
224	A length than less than zero is not allowed.
225	When using the -strong tag the minimum length password is 14 characters.
226	When using the -strong tag the maximum length password is 300 characters.
227	The MD5 algorithm is not supported when in FIPS mode.
228	The site tag is not supported for the -cert -list command. This attribute is added for backward compatibility and potential future enhancement.
229	The value associated with the -ca tag is not recognized. The value must be either 'true' or 'false'.
230	The value passed in with the -type tag is not valid.
231	The value passed in with the -expire tag is below the allowed range.
232	The encryption algorithm used or requested is not supported.
233	The target already exists.

IBM MQ for IBM i CL commands

A list of IBM MQ for IBM i CL commands 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
  - CHGMQMPRC, Change IBM MQ Process
  - CPYMQMPRC, Copy IBM MQ Process
  - CRTMQMPRC, Create IBM MQ Process
  - DLTMQMPRC, Delete IBM MQ Process
  - DSPMQMPRC, Display IBM MQ Process
  - WRKMQMPRC, 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

Add Queue Manager Information (ADDMQMINF)

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

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	Character value	Required, Positional 1
PREFIX	Queue Manager Prefix	Character value	Required, Positional 2
MQMDIR	Queue Manager Directory	Character value	Required, Positional 3
MQMLIB	Queue Manager Library	Name	Required, Positional 4
DATAPATH	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.

Examples

None

Error messages

Unknown

Add Queue Manager Journal (ADDMQMJRN)

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
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
JRN	Queue Manager Journal	*Character value*, ***DFT**	Optional, Positional 2
RMTJRNRDB	Remote Relational Database	Character value	Optional, Positional 3
RMTJRNSTS	Remote Journal Status	***ACTIVE**, *INACTIVE	Optional, Positional 4
RMTJRNDLV	Remote Journal Delivery	***SYNC**, *ASYNC	Optional, Positional 5
RMTJRNTIMO	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 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.

Examples

None

Error messages

Unknown

Connect MQ (CCTMQM)

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

Examples

None

Error messages

Unknown

Change Message Queue Manager (CHGMQM)

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

Table 53. Queue manager attributes

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 1
FORCE	Force	***NO**, *YES	Optional, Positional 2

Table 53. Queue manager attributes (continued)

Keyword	Description	Choices	Notes
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
TRGITV	Trigger interval	0-999999999, ***SAME**	Optional, Positional 4
UDLMSGQ	Undelivered message queue	*Character value*, *NONE, ***SAME**	Optional, Positional 5
DFTTMQ	Default transmission queue	*Character value*, *NONE, ***SAME**	Optional, Positional 6
MAXHDL	Maximum handle limit	0-999999999, ***SAME**	Optional, Positional 7
MAXUMSG	Maximum uncommitted messages	1-999999999, ***SAME**	Optional, Positional 8
AUTEVT	Authorization events enabled	***SAME**, *YES, *NO	Optional, Positional 9
INHEVT	Inhibit events enabled	***SAME**, *YES, *NO	Optional, Positional 10
LCLERREVT	Local error events enabled	***SAME**, *YES, *NO	Optional, Positional 11
RMTERREVT	Remote error events enabled	***SAME**, *YES, *NO	Optional, Positional 12
PFREVT	Performance events enabled	***SAME**, *YES, *NO	Optional, Positional 13
STRSTPEVT	Start and stop events enabled	***SAME**, *YES, *NO	Optional, Positional 14
CHAD	Automatic Channel Definition	***SAME**, *YES, *NO	Optional, Positional 15
CHADEV	Automatic Channel Definition events enabled	***SAME**, *YES, *NO	Optional, Positional 16
CHADEXIT	Automatic Channel Definition exit program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 17
	Qualifier 1: Automatic Channel Definition exit program	Name	
	Qualifier 2: Library	Name	
MAXMSGL	Max message length	32768-104857600, ***SAME**	Optional, Positional 18
CCSID	Coded Character Set	*Integer*, ***SAME**	Optional, Positional 19
CLWLDATA	Cluster workload exit data	*Character value*, ***SAME**, *NONE	Optional, Positional 20

Table 53. Queue manager attributes (continued)			
Keyword	**Description**	**Choices**	**Notes**
CLWLEXIT	Cluster workload exit	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 21
	Qualifier 1: Cluster workload exit	Name	
	Qualifier 2: Library	Name	
CLWLLEN	Cluster workload exit length	0-999999999, ***SAME**	Optional, Positional 22
REPOS	Repository name	*Character value*, *NONE, ***SAME**	Optional, Positional 23
REPOSNL	Repository name list	*Character value*, *NONE, ***SAME**	Optional, Positional 24
SSLCRLNL	SSL CRL Namelist	*Character value*, *NONE, ***SAME**	Optional, Positional 25
SSLKEYR	SSL Key Repository	*Character value*, *NONE, ***SAME**, *SYSTEM	Optional, Positional 26
SSLKEYRPWD	SSL Repository Password	*Character value*, *NONE, ***SAME**	Optional, Positional 27
SSLRSTCNT	SSL key reset count	0-999999999, ***SAME**, *NONE	Optional, Positional 28
IPADDRV	IP protocol	***SAME**, *IPv4, *IPv6	Optional, Positional 29
CLWLMRUC	Cluster workload channels	0-999999999, ***SAME**	Optional, Positional 30
CLWLUSEQ	Cluster workload queue use	***SAME**, *LOCAL, *ANY	Optional, Positional 31
LOGGEREVT	Log recovery events enabled	***SAME**, *YES, *NO	Optional, Positional 32
CHLEVT	Channel events enabled	***SAME**, *YES, *NO, *EXCEPTION	Optional, Positional 33
SSLEVT	SSL events enabled	***SAME**, *YES, *NO	Optional, Positional 34
SCHINIT	Channel initiator control	***SAME**, *QMGR, *MANUAL	Optional, Positional 35
SCMDSERV	Command server control	***SAME**, *QMGR, *MANUAL	Optional, Positional 36
MONQ	Queue Monitoring	***SAME**, *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 37
MONCHL	Channel Monitoring	***SAME**, *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 38
MONACLS	Cluster Sender Monitoring	***SAME**, *QMGR, *NONE, *LOW, *MEDIUM, *HIGH	Optional, Positional 39
STATMQI	Queue Manager Statistics	***SAME**, *OFF, *ON	Optional, Positional 40

Table 53. Queue manager attributes (continued)			
Keyword	**Description**	**Choices**	**Notes**
STATQ	Queue Statistics	***SAME**, *NONE, *OFF, *ON	Optional, Positional 41
STATCHL	Channel Statistics	***SAME**, *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 42
STATACLS	Cluster Sender Statistics	***SAME**, *QMGR, *NONE, *LOW, *MEDIUM, *HIGH	Optional, Positional 43
STATINT	Statistics Interval	1-604800, ***SAME**	Optional, Positional 44
ACCTMQI	MQI Accounting	***SAME**, *OFF, *ON	Optional, Positional 45
ACCTQ	Queue Accounting	***SAME**, *NONE, *OFF, *ON	Optional, Positional 46
ACCTINT	Accounting Interval	1-604800, ***SAME**	Optional, Positional 47
ACCTCONO	Accounting Override	***SAME**, *ENABLED, *DISABLED	Optional, Positional 48
ROUTEREC	Trace Route Recording	***SAME**, *MSG, *QUEUE, *DISABLED	Optional, Positional 49
ACTIVREC	Activity Recording	***SAME**, *MSG, *QUEUE, *DISABLED	Optional, Positional 50
MAXPROPLEN	Maximum Property Data Length	0-104857600, ***SAME**, *ANY	Optional, Positional 51
MARKINT	Message mark-browse interval	0-999999999, ***SAME**, *ANY	Optional, Positional 52
PSRTYCNT	PubSub max msg retry count	0-999999999, ***SAME**	Optional, Positional 53
PSNPMMSG	PubSub NPM msg	***SAME**, *DISCARD, *KEEP	Optional, Positional 54
PSNPMRES	PubSub NPM msg response	***SAME**, *NORMAL, *SAFE, *DISCARD, *KEEP	Optional, Positional 55
PSSYNCPT	PubSub syncpoint	***SAME**, *YES, *IFPER	Optional, Positional 56
PSMODE	Pubsub Engine Control	***SAME**, *ENABLED, *DISABLED, *COMPATIBLE	Optional, Positional 57
TREELIFE	Topic Tree Life Time	0-604000, ***SAME**	Optional, Positional 58
CFGEVT	Configuration events enabled	***SAME**, *YES, *NO	Optional, Positional 59
CMDEVT	Command events enabled	***SAME**, *YES, *NO, *NODSP	Optional, Positional 60
ACTVTRC	Activity tracing	*Character value*, *ON, ***SAME**, *OFF	Optional, Positional 61
ACTVCONO	Override activity tracing	*Character value*, *DISABLED, ***SAME**, *ENABLED	Optional, Positional 62

Table 53. Queue manager attributes (continued)			
Keyword	**Description**	**Choices**	**Notes**
CHLAUTH	Channel authentication	*Character value*, *DISABLED, ***SAME**, *ENABLED	Optional, Positional 63
CUSTOM	Custom attribute	*Character value*, *NONE, ***SAME**, 128 character	Optional, Positional 64
DFTCLXQ	Default cluster transmission queue type	***SAME**, *SCTQ, *CHANNEL	Optional, Positional 65
CERTLABL	Certificate label	***SAME**, *DFT	Optional, Positional 66
REVDNS	Reverse lookup of host name	***SAME**, *DISABLED, *ENABLED	Optional, Positional 67
CONNAUTH	Connection authentication object	***SAME**, *NONE, 48 character string	Optional, Positional 68

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 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.

SSL 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.

SSL 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 key repository. If you specify this value you are 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 manual for more details.

SSL 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.

SSL key reset count (SSLRSTCNT)

Specifies when SSL 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 hostname 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
  - MQRC_CHANNEL_STOPPED_BY_USER

***YES**
All channel events are generated.

In addition to those generated by *EXCEPTION the following channel events are also generated:

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED

  with the following reason qualifier:

  - MQRQ_CHANNEL_STOPPED_OK

SSL events enabled (SSLEVT)

Specifies whether SSL events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO**
SSL events are not generated.

***YES**
SSL events are generated.

The following event is generated:

- MQRC_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 switched off.

***ON**
API accounting data collection is switched on.

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 MQCONNX 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 MQCONNX 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 (PSRTYCNT)

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 (PSSYNCPT)

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 ACTVCONO is set to ENABLED, the value of this parameter can be overridden using the options field of the MQCNO structure.

ACTVCONO

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 MQCONNX 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 attribute string
Specify zero or more attributes as pairs of attribute name and value, separated by at least one space. The attribute name and 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. This behavior is implicit in versions of IBM WebSphere MQ, earlier than Version 7.5. In earlier versions, the queue manager attribute `DefClusterXmitQueueType` 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`.

CERTLABL

This attribute specified 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.

Change MQ AuthInfo object (CHGMQMAUTI)

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

Keyword	Description	Choices	Notes
AINAME	AuthInfo name	Character value	Required, Key, Positional 1

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
AUTHTYPE	AuthInfo type	*CRLLDAP, *OCSP, *IDPWOS, *IDPWLDAP	Optional, Positional 3
CONNAME	Connection name	*Character value*, ***SAME**	Optional, Positional 4
TEXT	Text 'description'	*Character value*, ***SAME**, *NONE	Optional, Positional 5
USERNAME	User name	*Character value*, ***SAME**, *NONE	Optional, Positional 6
PASSWORD	User password	*Character value*, ***SAME**, *NONE	Optional, Positional 7
OCSPURL	OCSP Responder URL	*Character value*, ***SAME**	Optional, Positional 8
CHCKCLNT	Authentication checks required	*ASQMGR, *REQUIRED, *REQADM	Optional, Positional 9
CHCKLOCL	Authentication checks required	*NONE, *OPTIONAL, *REQUIRED, *REQADM	Optional, Positional 10
FAILDELAY	Failure delay	*Integer value*	Optional, Positional 11
BASEDNU	Base user DN	*Character value*, ***SAME**	Optional, Positional 12
ADOPTCTX	Context adoption	*Integer value*	Optional, Positional 13
CLASSUSER	LDAP object class	*Character value*, ***SAME**	Optional, Positional 14
USERFIELD	LDAP user record	*Character value*, ***SAME**	Optional, Positional 15
SHORTUSER	User record	*Character value*, ***SAME**	Optional, Positional 16
SECCOMM	LDAP communications	*Character value*, ***SAME**	Optional, Positional 17
AUTHORMD	Authorization method	*Character value*, ***OS** *SEARCHGRP *SEARCHUSR	Optional, Positional 18
BASEDNG	Base DN for groups	*Character value*, ***SAME**	Optional, Positional 19
CLASSGRP	Object class for group	*Character value*, ***SAME**	Optional, Positional 20
FINDGRP	Attribute to find group membership	*Character value*, ***SAME**	Optional, Positional 21
GRPFIELD	Simple name for group	*Character value*, ***SAME**	Optional, Positional 22
NESTGRP	Group nesting	***NO** *YES	Optional, Positional 23
V 8.0.0.3 V 8.0.0.3 AUTHENMD	Authentication method	***OS** 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)

V 8.0.0.3

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)

V 8.0.0.2

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 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 all the groups to which the specified user belongs. The attribute to query is defined by the FINDGRP value, typically *memberOf*.

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)

V 8.0.0.2

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)

V 8.0.0.2 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)

V 8.0.0.2 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*.

If left blank, the queue manager uses *member* or *memberOf* depending on the setting of **AUTHORMD**.

This attribute is valid only for an **AUTHTYPE** of **IDPWLDAP*.

Simple name for group (GRPFIELD)

V 8.0.0.2

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)

V 8.0.0.2

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.

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 Reponder URL. The maximum string length is 256 characters.

Secure comms (SECCOMM)

Whether connectivity to the LDAP server should be done securely using SSL/TLS

YES
Connectivity to the LDAP server is made securely using SSL/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 SSL/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 SSL/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.

Change MQ Channel (CHGMQMCHL)

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" on page 1243 for information about the SEQNUM parameter.)

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Optional, Key, Positional 3
TRPTYPE	Transport type	*LU62, *TCP, ***SAME**	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
TGTMQMNAME	Target Queue Manager	*Character value*, *NONE, ***SAME**	Optional, Positional 6
CONNAME	Connection name	*Character value*, *NONE, ***SAME**	Optional, Positional 7
TPNAME	Transaction Program Name	*Character value*, *BLANK, ***SAME**	Optional, Positional 8
MODENAME	Mode Name	*Character value*, *BLANK, ***SAME**	Optional, Positional 9
TMQNAME	Transmission queue	*Character value*, ***SAME**	Optional, Positional 10
MCANAME	Message channel agent	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 11
	Qualifier 1: Message channel agent	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MCAUSRID	Message channel agent user ID	*Character value*, *NONE, *PUBLIC, ***SAME**	Optional, Positional 12
MCATYPE	Message channel agent Type	*PROCESS, *THREAD, ***SAME**	Optional, Positional 13
BATCHINT	Batch Interval	0-999999999, ***SAME**	Optional, Positional 14
BATCHSIZE	Batch size	1-9999, ***SAME**	Optional, Positional 15
DSCITV	Disconnect interval	0-999999, ***SAME**	Optional, Positional 16
SHORTTMR	Short retry interval	0-999999999, ***SAME**	Optional, Positional 17
SHORTRTY	Short retry count	0-999999999, ***SAME**	Optional, Positional 18
LONGTMR	Long retry interval	0-999999999, ***SAME**	Optional, Positional 19
LONGRTY	Long retry count	0-999999999, ***SAME**	Optional, Positional 20

Keyword	Description	Choices	Notes
SCYEXIT	Security exit	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 21
	Qualifier 1: Security exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSCYEXIT	Security exit	*Character value*, ***SAME**, *NONE	Optional, Positional 22
SCYUSRDATA	Security exit user data	*Character value*, ***SAME**, *NONE	Optional, Positional 23
SNDEXIT	Send exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 24
	Qualifier 1: Send exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSNDEXIT	Send exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 25
SNDUSRDATA	Send exit user data	Values (up to 10 repetitions): *Character value*, ***SAME**, *NONE	Optional, Positional 26
RCVEXIT	Receive exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 27
	Qualifier 1: Receive exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CRCVEXIT	Receive exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 28
RCVUSRDATA	Receive exit user data	Values (up to 10 repetitions): *Character value*, ***SAME**, *NONE	Optional, Positional 29
MSGEXIT	Message exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 30
	Qualifier 1: Message exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGUSRDATA	Message exit user data	Values (up to 10 repetitions): *Character value*, ***SAME**, *NONE	Optional, Positional 31

Keyword	Description	Choices	Notes
MSGRTYEXIT	Message retry exit	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 32
	Qualifier 1: Message retry exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGRTYDATA	Message retry exit data	*Character value*, ***SAME**, *NONE	Optional, Positional 33
MSGRTYNBR	Number of message retries	0-999999999, ***SAME**	Optional, Positional 34
MSGRTYITV	Message retry interval	0-999999999, ***SAME**	Optional, Positional 35
CVTMSG	Convert message	*YES, *NO, ***SAME**	Optional, Positional 36
PUTAUT	Put authority	*DFT, *CTX, ***SAME**	Optional, Positional 37
SEQNUMWRAP	Sequence number wrap	100-999999999, ***SAME**	Optional, Positional 38
MAXMSGLEN	Maximum message length	0-104857600, ***SAME**	Optional, Positional 39
HRTBTINTVL	Heartbeat interval	0-999999999, ***SAME**	Optional, Positional 40
NPMSPEED	Non Persistent Message Speed	*FAST, *NORMAL, ***SAME**	Optional, Positional 41
CLUSTER	Cluster Name	*Character value*, *NONE, ***SAME**	Optional, Positional 42
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SAME**	Optional, Positional 43
NETPRTY	Network Connection Priority	0-9, ***SAME**	Optional, Positional 44

Keyword	Description	Choices	Notes
SSLCIPH	SSL CipherSpec	*Character value*, '*NULL_MD5', '*NULL_SHA', '*RC4_MD5_EXPORT', '*RC4_MD5_US', '*RC4_SHA_US', '*RC2_MD5_EXPORT', '*DES_SHA_EXPORT', '*TRIPLE_DES_SHA_US', '*AES_SHA_US', '*TLS_RSA_WITH_NULL_MD5', '*TLS_RSA_WITH_NULL_SHA', '*TLS_RSA_EXPORT_WITH_RC4_40_MD5', '*TLS_RSA_WITH_RC4_128_MD5', '*TLS_RSA_WITH_RC4_128_SHA', '*TLS_RSA_EXPORT_WITH_RC2_40_MD5', '*TLS_RSA_WITH_DES_CBC_SHA', '*TLS_RSA_WITH_3DES_EDE_CBC_SHA', '*TLS_RSA_WITH_AES_128_CBC_SHA', '*TLS_RSA_WITH_AES_256_CBC_SHA', *NONE, ***SAME**	Optional, Positional 45 CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA is deprecated.
SSLCAUTH	SSL Client Authentication	*REQUIRED, *OPTIONAL, ***SAME**	Optional, Positional 46
SSLPEER	SSL Peer name	*Character value*, *NONE, ***SAME**	Optional, Positional 47
LOCLADDR	Local communication address	*Character value*, *NONE, ***SAME**	Optional, Positional 48
BATCHHB	Batch Heartbeat Interval	0-999999999, ***SAME**	Optional, Positional 49
USERID	Task user identifier	*Character value*, *NONE, ***SAME**	Optional, Positional 50
PASSWORD	Password	*Character value*, *NONE, ***SAME**	Optional, Positional 51
KAINT	Keep Alive Interval	0-99999, ***SAME**, *AUTO	Optional, Positional 52
COMPHDR	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, ***SAME**	Optional, Positional 53

Keyword	Description	Choices	Notes
COMPMSG	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST, ***SAME**	Optional, Positional 54
MONCHL	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SAME**	Optional, Positional 55
STATCHL	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SAME**	Optional, Positional 56
CLWLRANK	Cluster Workload Rank	0-9, ***SAME**	Optional, Positional 57
CLWLPRTY	Cluster Workload Priority	0-9, ***SAME**	Optional, Positional 58
CLWLWGHT	Cluster Channel Weight	1-99, ***SAME**	Optional, Positional 59
SHARECNV	Sharing Conversations	0-999999999, ***SAME**	Optional, Positional 60
PROPCTL	Property Control	*COMPAT, *NONE, *ALL, ***SAME**	Optional, Positional 61
MAXINST	Maximum Instances	0-999999999, ***SAME**	Optional, Positional 62
MAXINSTC	Maximum Instances Per Client	0-999999999, ***SAME**	Optional, Positional 63
CLNTWGHT	Client Channel Weight	0-99, ***SAME**	Optional, Positional 64
AFFINITY	Connection Affinity	*PREFERRED, *NONE, ***SAME**	Optional, Positional 65
BATCHLIM	Batch Data Limit	0-999999, ***SAME**	Optional, Positional 66
DFTRECON	Default client reconnetion	*NO, *YES, *QMGR, *DISABLED, ***SYSDFTCHL**	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.

  On AIX, HP-UX, IBM i, Linux, Solaris, and Windows, 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, *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, *CLUSSDR 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 persistant messages.

The possible values are:

***SAME**
The value of this attribute does not change.

***FAST**
The channel supports fast non persistant messages.

***NORMAL**
The channel does not support fast non persistant 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 *CLUSSDR 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 specifiifed.

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 *CLUSSDR 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 specifiifed.

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.

SSL CipherSpec (SSLCIPH)

SSLCIPH specifies the CipherSpec used in SSL 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.

SSL Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel carries out client authentication over SSL. 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.

SSL Peer name (SSLPEER)

SSLPEER specifies the X500 peer name used in SSL 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 SSL or TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different SSL or 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 (BATCHHB)

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 switched off.

***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 origininating 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.

This 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 MQCONNX, the client is not reconnected automatically.

***YES**
Unless overridden by MQCONNX, the client reconnects automatically.

***QMGR**
Unless overridden by MQCONNX, 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 MQCONNX MQI call.

This parameter is specified for a client connection channel, (CHLTYPE) *CLTCN

Examples

None

Error messages

Unknown

Change Queue Manager Journal (CHGMQMJRN)

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
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
JRN	Queue Manager Journal	*Character value*, ***DFT**	Optional, Positional 2
RMTJRNRDB	Remote Relational Database	Character value	Optional, Positional 3
RMTJRNSTS	Remote Journal Status	***ACTIVE**, *INACTIVE	Optional, Positional 4
RMTJRNDLV	Remote Journal Delivery	***SYNC**, *ASYNC	Optional, Positional 5
RMTJRNTIMO	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 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.

Examples

None

Error messages

Unknown

Change MQ Listener (CHGMQMLSR)

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

Keyword	Description	Choices	Notes
LSRNAME	Listener name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
CONTROL	Listener control	***SAME**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 4
PORT	Port number	0-65535, ***SAME**	Optional, Positional 5
IPADDR	IP Address	*Character value*, *BLANK, ***SAME**	Optional, Positional 6
BACKLOG	Listener backlog	0-999999999, ***SAME**	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.

Examples

None

Error messages

Unknown

Change MQ Namelist (CHGMQMNL)

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

Keyword	Description	Choices	Notes
NAMELIST	Namelist	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
NAMES	List of Names	Values (up to 256 repetitions): *Character value*, *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.

Examples

None

Error messages

Unknown

Change MQ Process (CHGMQMPRC)

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

Keyword	Description	Choices	Notes
PRCNAME	Process name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3

Keyword	Description	Choices	Notes
APPTYPE	Application type	*Integer*, ***SAME**, *CICS®, *MVS, *IMS, *OS2, *DOS, *UNIX, *QMGR, *OS400, *WINDOWS, *CICS_VSE, *WINDOWS_NT, *VMS, *NSK, *VOS, *IMS_BRIDGE, *XCF, *CICS_BRIDGE, *NOTES_AGENT, *BROKER, *JAVA, *DQM	Optional, Positional 4
APPID	Application identifier	*Character value*, ***SAME**	Optional, Positional 5
USRDATA	User data	*Character value*, ***SAME**, *NONE	Optional, Positional 6
ENVDATA	Environment data	*Character value*, ***SAME**, *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:

***SAME**
The attribute is unchanged.

***CICS**
Represents a CICS/400 application.

***MVS**
Represents an MVS application.

***IMS**
Represents an IMS application.

***OS2**
Represents an OS/2 application.

***DOS**
Represents a DOS application.

***UNIX**
Represents a UNIX application.

***QMGR**
Represents a queue manager.

***OS400**
Represents an IBM i application.

***WINDOWS**
Represents a Windows application.

***CICS_VSE**
Represents a CICS/VSE application.

***WINDOWS_NT**
Represents a Windows NT application.

***VMS**
Represents a VMS application.

***NSK**
Represents a Tandem/NSK application.

***VOS**
Represents a VOS application.

***IMS_BRIDGE**
Represents an IMS bridge application.

***XCF**
Represents an XCF application.

***CICS_BRIDGE**
Represents a CICS bridge application.

***NOTES_AGENT**
Represents a Lotus Notes application.

***BROKER**
Represents a broker application.

***JAVA**
Represents a Java application.

***DQM**
Represents a DQM application.

user-value
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.

Examples

None

Error messages

Unknown

Change MQ Queue (CHGMQMQ)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Change MQ Queue (**CHGMQMQ**) command changes the specified attributes of an existing MQ queue.

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
QTYPE	Queue type	Character value	Optional, Positional 3
FORCE	Force	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
PUTENBL	Put enabled	***SAME**, *NO, *YES	Optional, Positional 6
DFTPTY	Default message priority	0-9, ***SAME**	Optional, Positional 7
DFTMSGPST	Default message persistence	***SAME**, *NO, *YES	Optional, Positional 8
PRCNAME	Process name	*Character value*, *NONE, ***SAME**	Optional, Positional 9
TRGENBL	Triggering enabled	***SAME**, *NO, *YES	Optional, Positional 10
GETENBL	Get enabled	***SAME**, *NO, *YES	Optional, Positional 11
SHARE	Sharing enabled	***SAME**, *NO, *YES	Optional, Positional 12
DFTSHARE	Default share option	***SAME**, *NO, *YES	Optional, Positional 13
MSGDLYSEQ	Message delivery sequence	***SAME**, *PTY, *FIFO	Optional, Positional 14
HDNBKTCNT	Harden backout count	***SAME**, *NO, *YES	Optional, Positional 15
TRGTYPE	Trigger type	***SAME**, *FIRST, *ALL, *DEPTH, *NONE	Optional, Positional 16
TRGDEPTH	Trigger depth	1-999999999, ***SAME**	Optional, Positional 17
TRGMSGPTY	Trigger message priority	0-9, ***SAME**	Optional, Positional 18
TRGDATA	Trigger data	*Character value*, *NONE, ***SAME**	Optional, Positional 19
RTNITV	Retention interval	0-999999999, ***SAME**	Optional, Positional 20
MAXDEPTH	Maximum queue depth	0-999999999, ***SAME**	Optional, Positional 21
MAXMSGLEN	Maximum message length	0-104857600, ***SAME**	Optional, Positional 22
BKTTHLD	Backout threshold	0-999999999, ***SAME**	Optional, Positional 23
BKTQNAME	Backout requeue name	*Character value*, *NONE, ***SAME**	Optional, Positional 24
INITQNAME	Initiation queue	*Character value*, *NONE, ***SAME**	Optional, Positional 25
USAGE	Usage	***SAME**, *NORMAL, *TMQ	Optional, Positional 26

Keyword	Description	Choices	Notes
DFNTYPE	Definition type	***SAME**, *TEMPDYN, *PERMDYN	Optional, Positional 27
TGTQNAME	Target object	*Character value*, ***SAME**	Optional, Positional 28
RMTQNAME	Remote queue	*Character value*, ***SAME**, *NONE	Optional, Positional 29
RMTMQMNAME	Remote Message Queue Manager	*Character value*, ***SAME**	Optional, Positional 30
TMQNAME	Transmission queue	*Character value*, *NONE, ***SAME**	Optional, Positional 31
HIGHTHLD	Queue depth high threshold	0-100, ***SAME**	Optional, Positional 32
LOWTHLD	Queue depth low threshold	0-100, ***SAME**	Optional, Positional 33
FULLEVT	Queue full events enabled	***SAME**, *NO, *YES	Optional, Positional 34
HIGHEVT	Queue high events enabled	***SAME**, *NO, *YES	Optional, Positional 35
LOWEVT	Queue low events enabled	***SAME**, *NO, *YES	Optional, Positional 36
SRVITV	Service interval	0-999999999, ***SAME**	Optional, Positional 37
SRVEVT	Service interval events	***SAME**, *HIGH, *OK, *NONE	Optional, Positional 38
DISTLIST	Distribution list support	***SAME**, *NO, *YES	Optional, Positional 39
CLUSTER	Cluster Name	*Character value*, ***SAME**, *NONE	Optional, Positional 40
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SAME**	Optional, Positional 41
DEFBIND	Default Binding	***SAME**, *OPEN, *NOTFIXED, *GROUP	Optional, Positional 42
CLWLRANK	Cluster Workload Rank	0-9, ***SAME**	Optional, Positional 43
CLWLPRTY	Cluster Workload Priority	0-9, ***SAME**	Optional, Positional 44
CLWLUSEQ	Cluster workload queue use	***SAME**, *QMGR, *LOCAL, *ANY	Optional, Positional 45
MONQ	Queue Monitoring	***SAME**, *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 46
STATQ	Queue Statistics	***SAME**, *QMGR, *OFF, *ON	Optional, Positional 47
ACCTQ	Queue Accounting	***SAME**, *QMGR, *OFF, *ON	Optional, Positional 48
NPMCLASS	Non Persistent Message Class	***SAME**, *NORMAL, *HIGH	Optional, Positional 49

Keyword	Description	Choices	Notes
MSGREADAHD	Message Read Ahead	***SAME**, *DISABLED, *NO, *YES	Optional, Positional 50
DFTPUTRESP	Default Put Response	***SAME**, *SYNC, *ASYNC	Optional, Positional 51
PROPCTL	Property Control	***SAME**, *COMPAT, *NONE, *ALL, *FORCE, *V6COMPAT	Optional, Positional 52
TARGTYPE	Target Type	***SAME**, *QUEUE, *TOPIC	Optional, Positional 53
CUSTOM	Custom attribute	*Character value*, *BLANK, ***SAME**	Optional, Positional 54
"CLCHNAME" on page 274	Cluster-sender channel name	*Character value*, *NONE, ***SAME**	Optional, Positional 55

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 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 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 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:

***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 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:

***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 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.

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 switched off.

***ON**
Statistics data collection is switched on for this queue.

Queue Accounting (ACCTQ)

Controls the collection of accouting 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**
Accouting data collection is based upon the setting of the queue manager attribute ACCTQ.

***OFF**
Accounting data collection for this queue is switched off.

***ON**
Accounting data collection is switched on 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 and 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`.

Change MQ Subscription (CHGMQMSUB)

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

Keyword	Description	Choices	Notes
SUBID	Subscription identifier	*Character value*, ***SAME**	Optional, Key, Positional 2
SUBNAME	Subscription name	*Character value*, ***SAME**	Optional, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
TOPICSTR	Topic string	*Character value*, *NONE, ***SAME**	Optional, Positional 4
TOPICOBJ	Topic object	*Character value*, *NONE, ***SAME**	Optional, Positional 5
DEST	Destination	*Character value*, ***SAME**	Optional, Positional 6

Keyword	Description	Choices	Notes
DESTMQM	Destination Queue Manager	*Character value*, *NONE, ***SAME**	Optional, Positional 7
DESTCRRLID	Destination Correlation Id	*Character value*, *NONE, ***SAME**	Optional, Positional 8
PUBACCT	Publish Accounting Token	*Character value*, *NONE, ***SAME**	Optional, Positional 9
PUBAPPID	Publish Application Id	*Character value*, *NONE, ***SAME**	Optional, Positional 10
SUBUSER	Subscription User Id	*Character value*, ***SAME**	Optional, Positional 11
USERDATA	Subscription User Data	*Character value*, *NONE, ***SAME**	Optional, Positional 12
SELECTOR	Selector String	*Character value*, *NONE, ***SAME**	Optional, Positional 13
PSPROP	PubSub Property	***SAME**, *NONE, *COMPAT, *RFH2, *MSGPROP	Optional, Positional 14
DESTCLASS	Destination Class	***SAME**, *MANAGED, *PROVIDED	Optional, Positional 15
VARUSER	Variable User	***SAME**, *ANY, *FIXED	Optional, Positional 16
REQONLY	Request Publications	***SAME**, *YES, *NO	Optional, Positional 17
PUBPTY	Publish Priority	0-9, ***SAME**, *ASPUB, *ASQDEF	Optional, Positional 18
WSCHEMA	Wildcard Schema	***SAME**, *CHAR, *TOPIC	Optional, Positional 19
EXPIRY	Expiry Time	0-999999999, ***SAME**, *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 (DESTCRRLID)

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 Version 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 (WSCHEMA)

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.

Examples

None

Error messages

Unknown

Change MQ Service (CHGMQMSVC)

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

Keyword	Description	Choices	Notes
SVCNAME	Service name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
STRCMD	Start program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 4
	Qualifier 1: Start program	Name	
	Qualifier 2: Library	Name	
STRARG	Start program arguments	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
ENDCMD	End program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 6
	Qualifier 1: End program	Name	
	Qualifier 2: Library	Name	
ENDARG	End program arguments	*Character value*, *BLANK, ***SAME**	Optional, Positional 7
STDOUT	Standard output	*Character value*, *BLANK, ***SAME**	Optional, Positional 8
STDERR	Standard error	*Character value*, *BLANK, ***SAME**	Optional, Positional 9
TYPE	Service type	***SAME**, *CMD, *SVR	Optional, Positional 10
CONTROL	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.

Examples

None

Error messages

Unknown

Change MQ Topic (CHGMQMTOP)

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

Keyword	Description	Choices	Notes
TOPNAME	Topic name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
TOPICSTR	Topic string	*Character value*, *BLANK, ***SAME**	Optional, Positional 4
DURSUB	Durable subscriptions	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 5
MGDDURMDL	Durable model queue	*Character value*, *NONE, ***SAME**	Optional, Positional 6

Keyword	Description	Choices	Notes
MGDNDURMDL	Non-durable model queue	*Character value*, *NONE, ***SAME**	Optional, Positional 7
PUBENBL	Publish	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 8
SUBENBL	Subscribe	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 9
DFTPTY	Default message priority	0-9, ***SAME**, *ASPARENT	Optional, Positional 10
DFTMSGPST	Default message persistence	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 11
DFTPUTRESP	Default Put Response	***SAME**, *ASPARENT, *SYNC, *ASYNC	Optional, Positional 12
WILDCARD	Wildcard behavior	***SAME**, *PASSTHRU, *BLOCK	Optional, Positional 13
PMSGDLV	Persistent message delivery	***SAME**, *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 14
NPMSGDLV	Non-persistent message deliver	***SAME**, *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 15
CUSTOM	Custom attribute	*Character value*, *BLANK, ***SAME**	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 (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:

***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 and value pairs must have the form `NAME(VALUE)` and be specified in uppercase. Single quotes must be escaped with another single quote.

Examples

None

Error messages

Unknown

Clear MQ Pub/Sub Broker (CLRMQMBRK)

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
MQMNAME	Message Queue Manager name	Character value	Required, Positional 1
BRKPARENT	Break Parent link	***NO**, *YES	Optional, Positional 2
CHILDMQM	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.

Examples

None

Error messages

Unknown

Clear MQ Queue (CLRMQMQ)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Clear MQ Queue (CLRMQMQ) 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
QNAME	Queue name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***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.

Examples

None

Error messages

Unknown

Clear MQ Topic String (CLRMQMTOP)

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
TOPICSTR	Topic string	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
CLRTYPE	Clear type	*RETAINED	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.

Examples

None

Error messages

Unknown

Copy MQ AuthInfo object (CPYMQMAUTI)

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

Keyword	Description	Choices	Notes
FROMAI	From AuthInfo name	Character value	Required, Key, Positional 1
TOAI	To AuthInfo name	Character value	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3

Keyword	Description	Choices	Notes
AUTHTYPE	AuthInfo type	*CRLLDAP, *OCSP, *IDPWOS, *IDPWLDAP	Optional, Positional 4
CONNAME	Connection name	*Character value*, ***SAME**	Optional, Positional 5
REPLACE	Replace	***NO**, *YES	Optional, Positional 6
TEXT	Text 'description'	*Character value*, ***SAME**, *NONE	Optional, Positional 7
USERNAME	User name	*Character value*, ***SAME**, *NONE	Optional, Positional 8
PASSWORD	User password	*Character value*, ***SAME**, *NONE	Optional, Positional 9
OCSPURL	OCSP Responder URL	*Character value*, ***SAME**	Optional, Positional 10
CHCKCLNT	Authentication checks required	*ASQMGR, *REQUIRED, *REQADM	Optional, Positional 11
CHCKLOCL	Authentication checks required	*NONE, *OPTIONAL, *REQUIRED, *REQADM	Optional, Positional 12
FAILDELAY	Failure delay	*Integer value*	Optional, Positional 13
BASEDNU	Base user DN	*Character value*, ***SAME**	Optional, Positional 14
ADOPTCTX	Context adoption	*Integer value*	Optional, Positional 15
CLASSUSR	LDAP object class	*Character value*, ***SAME**	Optional, Positional 16
SHORTUSR	Short user name	*Character value*, ***SAME**	Optional, Positional 17
USRFIELD	User field	*Character value*, ***SAME**	Optional, Positional 18
SECCOMM	LDAP communications	*Character value*, ***SAME**	Optional, Positional 19
AUTHORMD	Authorization method	*Character value*, ***OS** *SEARCHGRP *SEARCHUSR	Optional, Positional 20
BASEDNG	Base DN for groups	*Character value*, ***SAME**	Optional, Positional 21
CLASSGRP	Object class for group	*Character value*, ***SAME**	Optional, Positional 22
FINDGRP	Attribute to find group membership	*Character value*, ***SAME**	Optional, Positional 23
GRPFIELD	Simple name for group	*Character value*, ***SAME**	Optional, Positional 24
NESTGRP	Group nesting	***NO** *YES	Optional, Positional 25
V 8.0.0.3 V 8.0.0.3 AUTHENMD	Authentication method	***OS** 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)

V 8.0.0.3

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)

V 8.0.0.2

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 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 all the groups to which the specified user belongs. The attribute to query is defined by the FINDGRP value, typically *memberOf*.

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)

V 8.0.0.2

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)

V 8.0.0.2 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)

V 8.0.0.2 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*.

If left blank, the queue manager uses *member* or *memberOf* depending on the setting of **AUTHORMD**.

This attribute is valid only for an **AUTHTYPE** of **IDPWLDAP*.

Simple name for group (GRPFIELD)

V 8.0.0.2

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)

V 8.0.0.2

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.

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 SSL/TLS

YES
Connectivity to the LDAP server is made securely using SSL/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 SSL/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 SSL/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 Reponder URL. The maximum string length is 256 characters.

Examples

None

Error messages

Unknown

Copy MQ Channel (CPYMQMCHL)

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

Keyword	Description	Choices	Notes
FROMCHL	From channel	Character value	Required, Key, Positional 1
TOCHL	To channel	Character value	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Optional, Key, Positional 4
REPLACE	Replace	***NO**, *YES	Optional, Positional 5
TRPTYPE	Transport type	*LU62, *TCP, ***SAME**	Optional, Positional 6
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 7
TGTMQMNAME	Target Queue Manager	*Character value*, *NONE, ***SAME**	Optional, Positional 8
CONNAME	Connection name	*Character value*, *NONE, ***SAME**	Optional, Positional 9
TPNAME	Transaction Program Name	*Character value*, *BLANK, ***SAME**	Optional, Positional 10
MODENAME	Mode Name	*Character value*, *BLANK, ***SAME**	Optional, Positional 11
TMQNAME	Transmission queue	*Character value*, ***SAME**	Optional, Positional 12

Keyword	Description	Choices	Notes
MCANAME	Message channel agent	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 13
	Qualifier 1: Message channel agent	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MCAUSRID	Message channel agent user ID	*Character value*, *NONE, *PUBLIC, ***SAME**	Optional, Positional 14
MCATYPE	Message channel agent Type	*PROCESS, *THREAD, ***SAME**	Optional, Positional 15
BATCHINT	Batch Interval	0-999999999, ***SAME**	Optional, Positional 16
BATCHSIZE	Batch size	1-9999, ***SAME**	Optional, Positional 17
DSCITV	Disconnect interval	0-999999, ***SAME**	Optional, Positional 18
SHORTTMR	Short retry interval	0-999999999, ***SAME**	Optional, Positional 19
SHORTRTY	Short retry count	0-999999999, ***SAME**	Optional, Positional 20
LONGTMR	Long retry interval	0-999999999, ***SAME**	Optional, Positional 21
LONGRTY	Long retry count	0-999999999, ***SAME**	Optional, Positional 22
SCYEXIT	Security exit	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 23
	Qualifier 1: Security exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSCYEXIT	Security exit	*Character value*, ***SAME**, *NONE	Optional, Positional 24
SCYUSRDATA	Security exit user data	*Character value*, ***SAME**, *NONE	Optional, Positional 25
SNDEXIT	Send exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 26
	Qualifier 1: Send exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSNDEXIT	Send exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 27
SNDUSRDATA	Send exit user data	Values (up to 10 repetitions): *Character value*, ***SAME**, *NONE	Optional, Positional 28

Keyword	Description	Choices	Notes
RCVEXIT	Receive exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 29
	Qualifier 1: Receive exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CRCVEXIT	Receive exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 30
RCVUSRDATA	Receive exit user data	Values (up to 10 repetitions): *Character value*, ***SAME**, *NONE	Optional, Positional 31
MSGEXIT	Message exit	Single values: ***SAME**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 32
	Qualifier 1: Message exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGUSRDATA	Message exit user data	Values (up to 10 repetitions): *Character value*, ***SAME**, *NONE	Optional, Positional 33
MSGRTYEXIT	Message retry exit	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 34
	Qualifier 1: Message retry exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGRTYDATA	Message retry exit data	*Character value*, ***SAME**, *NONE	Optional, Positional 35
MSGRTYNBR	Number of message retries	0-999999999, ***SAME**	Optional, Positional 36
MSGRTYITV	Message retry interval	0-999999999, ***SAME**	Optional, Positional 37
CVTMSG	Convert message	*YES, *NO, ***SAME**	Optional, Positional 38
PUTAUT	Put authority	*DFT, *CTX, ***SAME**	Optional, Positional 39
SEQNUMWRAP	Sequence number wrap	100-999999999, ***SAME**	Optional, Positional 40
MAXMSGLEN	Maximum message length	0-104857600, ***SAME**	Optional, Positional 41
HRTBTINTVL	Heartbeat interval	0-999999999, ***SAME**	Optional, Positional 42
NPMSPEED	Non Persistent Message Speed	*FAST, *NORMAL, ***SAME**	Optional, Positional 43

Keyword	Description	Choices	Notes
CLUSTER	Cluster Name	*Character value*, *NONE, ***SAME**	Optional, Positional 44
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SAME**	Optional, Positional 45
NETPRTY	Network Connection Priority	0-9, ***SAME**	Optional, Positional 46
SSLCIPH	SSL CipherSpec	*Character value*, '*NULL_MD5', '*NULL_SHA', '*RC4_MD5_EXPORT', '*RC4_MD5_US', '*RC4_SHA_US', '*RC2_MD5_EXPORT', '*DES_SHA_EXPORT', '*TRIPLE_DES_SHA_US', '*AES_SHA_US', '*TLS_RSA_WITH_NULL_MD5', '*TLS_RSA_WITH_NULL_SHA', '*TLS_RSA_EXPORT_WITH_RC4_40_MD5', '*TLS_RSA_WITH_RC4_128_MD5', '*TLS_RSA_WITH_RC4_128_SHA', '*TLS_RSA_EXPORT_WITH_RC2_40_MD5', '*TLS_RSA_WITH_DES_CBC_SHA', '*TLS_RSA_WITH_3DES_EDE_CBC_SHA', '*TLS_RSA_WITH_AES_128_CBC_SHA', '*TLS_RSA_WITH_AES_256_CBC_SHA', *NONE, ***SAME**	Optional, Positional 47 CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA is deprecated.
SSLCAUTH	SSL Client Authentication	*REQUIRED, *OPTIONAL, ***SAME**	Optional, Positional 48
SSLPEER	SSL Peer name	*Character value*, *NONE, ***SAME**	Optional, Positional 49
LOCLADDR	Local communication address	*Character value*, *NONE, ***SAME**	Optional, Positional 50
BATCHHB	Batch Heartbeat Interval	0-999999999, ***SAME**	Optional, Positional 51
USERID	Task user identifier	*Character value*, *NONE, ***SAME**	Optional, Positional 52
PASSWORD	Password	*Character value*, *NONE, ***SAME**	Optional, Positional 53
KAINT	Keep Alive Interval	0-99999, ***SAME**, *AUTO	Optional, Positional 54

Keyword	Description	Choices	Notes
COMPHDR	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, ***SAME**	Optional, Positional 55
COMPMSG	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST, ***SAME**	Optional, Positional 56
MONCHL	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SAME**	Optional, Positional 57
STATCHL	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SAME**	Optional, Positional 58
CLWLRANK	Cluster Workload Rank	0-9, ***SAME**	Optional, Positional 59
CLWLPRTY	Cluster Workload Priority	0-9, ***SAME**	Optional, Positional 60
CLWLWGHT	Cluster Channel Weight	1-99, ***SAME**	Optional, Positional 61
SHARECNV	Sharing Conversations	0-999999999, ***SAME**	Optional, Positional 62
PROPCTL	Property Control	*COMPAT, *NONE, *ALL, ***SAME**	Optional, Positional 63
MAXINST	Maximum Instances	0-999999999, ***SAME**	Optional, Positional 64
MAXINSTC	Maximum Instances Per Client	0-999999999, ***SAME**	Optional, Positional 65
CLNTWGHT	Client Channel Weight	0-99, ***SAME**	Optional, Positional 66
AFFINITY	Connection Affinity	*PREFERRED, *NONE, ***SAME**	Optional, Positional 67
BATCHLIM	Batch Data Limit	0-999999, ***SAME**	Optional, Positional 68
DFTRECON	Default client reconnetion	*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 AIX, HP-UX, IBM i, Linux, Solaris, and Windows, 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, *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 (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 (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 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, *CLUSSDR 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 persistant messages.

The possible values are:

***SAME**
The value of this attribute does not change.

***FAST**
The channel supports fast non persistant messages.

***NORMAL**
The channel does not support fast non persistant 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 *CLUSSDR 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 specififed.

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 *CLUSSDR 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 specifiifed.

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.

SSL CipherSpec (SSLCIPH)

SSLCIPH specifies the CipherSpec used in SSL 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.

SSL Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel carries out client authentication over SSL. 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.

SSL Peer name (SSLPEER)

SSLPEER specifies the X500 peer name used in SSL 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 SSL or TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different SSL or 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 (BATCHHB)

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 switched off.

***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 origininating 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.

This 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 MQCONNX, the client is not reconnected automatically.

***YES**
Unless overridden by MQCONNX, the client reconnects automatically.

***QMGR**
Unless overridden by MQCONNX, 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 MQCONNX MQI call.

This parameter is specified for a client connection channel, (CHLTYPE) *CLTCN

Examples

None

Error messages

Unknown

Copy MQ Listener (CPYMQMLSR)

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

Keyword	Description	Choices	Notes
FROMLSR	From Listener	Character value	Required, Key, Positional 1
TOLSR	To Listener	Character value	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
CONTROL	Listener control	***SAME**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 6
PORT	Port number	0-65535, ***SAME**	Optional, Positional 7
IPADDR	IP Address	*Character value*, *BLANK, ***SAME**	Optional, Positional 8
BACKLOG	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.

Examples

None

Error messages

Unknown

Copy MQ Namelist (CPYMQMNL)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Copy MQ Namelist (CPYMQMNL) command copies an MQ namelist.

Parameters

Keyword	Description	Choices	Notes
FROMNL	From Namelist	Character value	Required, Key, Positional 1
TONL	To Namelist	Character value	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
NAMES	List of Names	Values (up to 256 repetitions): *Character value*, *BLANKS, ***SAME**, *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.

Examples

None

Error messages

Unknown

Copy MQ Process (CPYMQMPRC)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Copy MQ Process (CPYMQMPRC) 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

Keyword	Description	Choices	Notes
FROMPRC	From process	Character value	Required, Key, Positional 1
TOPRC	To process	Character value	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
APPTYPE	Application type	*Integer*, ***SAME**, *CICS, *MVS, *IMS, *OS2, *DOS, *UNIX, *QMGR, *OS400, *WINDOWS, *CICS_VSE, *WINDOWS_NT, *VMS, *NSK, *VOS, *IMS_BRIDGE, *XCF, *CICS_BRIDGE, *NOTES_AGENT, *BROKER, *JAVA, *DQM	Optional, Positional 6
APPID	Application identifier	*Character value*, ***SAME**	Optional, Positional 7
USRDATA	User data	*Character value*, ***SAME**, *NONE	Optional, Positional 8
ENVDATA	Environment data	*Character value*, ***SAME**, *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:

***SAME**
The attribute is unchanged.

***CICS**
Represents a CICS/400 application.

***MVS**
Represents an MVS application.

***IMS**
Represents an IMS application.

***OS2**
Represents an OS/2 application.

***DOS**
Represents a DOS application.

***UNIX**
Represents a UNIX application.

***QMGR**
Represents a queue manager.

***OS400**
Represents an IBM i application.

***WINDOWS**
Represents a Windows application.

***CICS_VSE**
Represents a CICS/VSE application.

***WINDOWS_NT**
Represents a Windows NT application.

***VMS**
Represents a VMS application.

***NSK**
Represents a Tandem/NSK application.

***VOS**
Represents a VOS application.

***IMS_BRIDGE**
Represents an IMS bridge application.

***XCF**
Represents an XCF application.

***CICS_BRIDGE**
Represents a CICS bridge application.

***NOTES_AGENT**
Represents a Lotus Notes application.

***BROKER**
Represents a broker application.

***JAVA**
Represents a Java application.

***DQM**
Represents a DQM application.

user-value
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.

Examples

None

Error messages

Unknown

Copy MQ Queue (CPYMQMQ)

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

Keyword	Description	Choices	Notes
FROMQ	From queue name	Character value	Required, Key, Positional 1
TOQ	To queue name	Character value	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
QTYPE	Queue type	Character value	Optional, Positional 4
REPLACE	Replace	***NO**, *YES	Optional, Positional 5
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 6
PUTENBL	Put enabled	***SAME**, *NO, *YES	Optional, Positional 7
DFTPTY	Default message priority	0-9, ***SAME**	Optional, Positional 8
DFTMSGPST	Default message persistence	***SAME**, *NO, *YES	Optional, Positional 9
PRCNAME	Process name	*Character value*, *NONE, ***SAME**	Optional, Positional 10
TRGENBL	Triggering enabled	***SAME**, *NO, *YES	Optional, Positional 11
GETENBL	Get enabled	***SAME**, *NO, *YES	Optional, Positional 12
SHARE	Sharing enabled	***SAME**, *NO, *YES	Optional, Positional 13
DFTSHARE	Default share option	***SAME**, *NO, *YES	Optional, Positional 14
MSGDLYSEQ	Message delivery sequence	***SAME**, *PTY, *FIFO	Optional, Positional 15
HDNBKTCNT	Harden backout count	***SAME**, *NO, *YES	Optional, Positional 16
TRGTYPE	Trigger type	***SAME**, *FIRST, *ALL, *DEPTH, *NONE	Optional, Positional 17
TRGDEPTH	Trigger depth	1-999999999, ***SAME**	Optional, Positional 18
TRGMSGPTY	Trigger message priority	0-9, ***SAME**	Optional, Positional 19
TRGDATA	Trigger data	*Character value*, *NONE, ***SAME**	Optional, Positional 20
RTNITV	Retention interval	0-999999999, ***SAME**	Optional, Positional 21
MAXDEPTH	Maximum queue depth	0-999999999, ***SAME**	Optional, Positional 22
MAXMSGLEN	Maximum message length	0-104857600, ***SAME**	Optional, Positional 23
BKTTHLD	Backout threshold	0-999999999, ***SAME**	Optional, Positional 24
BKTQNAME	Backout requeue name	*Character value*, *NONE, ***SAME**	Optional, Positional 25
INITQNAME	Initiation queue	*Character value*, *NONE, ***SAME**	Optional, Positional 26

Keyword	Description	Choices	Notes
USAGE	Usage	***SAME**, *NORMAL, *TMQ	Optional, Positional 27
DFNTYPE	Definition type	***SAME**, *TEMPDYN, *PERMDYN	Optional, Positional 28
TGTQNAME	Target object	*Character value*, ***SAME**	Optional, Positional 29
RMTQNAME	Remote queue	*Character value*, ***SAME**, *NONE	Optional, Positional 30
RMTMQMNAME	Remote Message Queue Manager	*Character value*, ***SAME**	Optional, Positional 31
TMQNAME	Transmission queue	*Character value*, *NONE, ***SAME**	Optional, Positional 32
HIGHTHLD	Queue depth high threshold	0-100, ***SAME**	Optional, Positional 33
LOWTHLD	Queue depth low threshold	0-100, ***SAME**	Optional, Positional 34
FULLEVT	Queue full events enabled	***SAME**, *NO, *YES	Optional, Positional 35
HIGHEVT	Queue high events enabled	***SAME**, *NO, *YES	Optional, Positional 36
LOWEVT	Queue low events enabled	***SAME**, *NO, *YES	Optional, Positional 37
SRVITV	Service interval	0-999999999, ***SAME**	Optional, Positional 38
SRVEVT	Service interval events	***SAME**, *HIGH, *OK, *NONE	Optional, Positional 39
DISTLIST	Distribution list support	***SAME**, *NO, *YES	Optional, Positional 40
CLUSTER	Cluster Name	*Character value*, ***SAME**, *NONE	Optional, Positional 41
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SAME**	Optional, Positional 42
DEFBIND	Default Binding	***SAME**, *OPEN, *NOTFIXED, *GROUP	Optional, Positional 43
CLWLRANK	Cluster Workload Rank	0-9, ***SAME**	Optional, Positional 44
CLWLPRTY	Cluster Workload Priority	0-9, ***SAME**	Optional, Positional 45
CLWLUSEQ	Cluster workload queue use	***SAME**, *QMGR, *LOCAL, *ANY	Optional, Positional 46
MONQ	Queue Monitoring	***SAME**, *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 47
STATQ	Queue Statistics	***SAME**, *QMGR, *OFF, *ON	Optional, Positional 48
ACCTQ	Queue Accounting	***SAME**, *QMGR, *OFF, *ON	Optional, Positional 49

Keyword	Description	Choices	Notes
NPMCLASS	Non Persistent Message Class	***SAME**, *NORMAL, *HIGH	Optional, Positional 50
MSGREADAHD	Message Read Ahead	***SAME**, *DISABLED, *NO, *YES	Optional, Positional 51
DFTPUTRESP	Default Put Response	***SAME**, *SYNC, *ASYNC	Optional, Positional 52
PROPCTL	Property Control	***SAME**, *COMPAT, *NONE, *ALL, *FORCE, *V6COMPAT	Optional, Positional 53
TARGTYPE	Target Type	***SAME**, *QUEUE, *TOPIC	Optional, Positional 54
CUSTOM	Custom attribute	*Character value*, *BLANK, ***SAME**	Optional, Positional 55
CLCHNAME	Cluster-sender channel name	*Character value*, *NONE, ***SAME**	Optional, Positional 56

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 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:

***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 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:

***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 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.

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 switched off.

***ON**
Statistics data collection is switched on for this queue.

Queue Accounting (ACCTQ)

Controls the collection of accouting 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**
Accouting data collection is based upon the setting of the queue manager attribute ACCTQ.

***OFF**
Accounting data collection for this queue is switched off.

***ON**
Accounting data collection is switched on 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 and 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`.

Copy MQ Subscription (CPYMQMSUB)

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
FROMSUBID	From subscription identifier	*Character value*, ***SAME**	Optional, Key, Positional 3
FROMSUB	From subscription	*Character value*, ***SAME**	Optional, Key, Positional 2
TOSUB	To subscription	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 4
REPLACE	Replace	***NO**, *YES	Optional, Positional 5
TOPICSTR	Topic string	*Character value*, *NONE, ***SAME**	Optional, Positional 6

Keyword	Description	Choices	Notes
TOPICOBJ	Topic object	*Character value*, *NONE, ***SAME**	Optional, Positional 7
DEST	Destination	*Character value*, *NONE, ***SAME**	Optional, Positional 8
DESTMQM	Destination Queue Manager	*Character value*, *NONE, ***SAME**	Optional, Positional 9
DESTCRRLID	Destination Correlation Id	*Character value*, *NONE, ***SAME**	Optional, Positional 10
PUBACCT	Publish Accounting Token	*Character value*, *NONE, ***SAME**	Optional, Positional 11
PUBAPPID	Publish Application Id	*Character value*, *NONE, ***SAME**	Optional, Positional 12
SUBUSER	Subscription User Id	*Character value*, *CURRENT, ***SAME**	Optional, Positional 13
USERDATA	Subscription User Data	*Character value*, *NONE, ***SAME**	Optional, Positional 14
SELECTOR	Selector String	*Character value*, *NONE, ***SAME**	Optional, Positional 15
PSPROP	PubSub Property	***SAME**, *NONE, *COMPAT, *RFH2, *MSGPROP	Optional, Positional 16
DESTCLASS	Destination Class	***SAME**, *MANAGED, *PROVIDED	Optional, Positional 17
SUBSCOPE	Subscription Scope	***SAME**, *ALL, *QMGR	Optional, Positional 18
VARUSER	Variable User	***SAME**, *ANY, *FIXED	Optional, Positional 19
REQONLY	Request Publications	***SAME**, *YES, *NO	Optional, Positional 20
PUBPTY	Publish Priority	0-9, ***SAME**, *ASPUB, *ASQDEF	Optional, Positional 21
WSCHEMA	Wildcard Schema	***SAME**, *CHAR, *TOPIC	Optional, Positional 22
EXPIRY	Expiry Time	0-999999999, ***SAME**, *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 (DESTCRRLID)

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 Version 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.

***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 (WSCHEMA)

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.

Examples

None

Error messages

Unknown

Copy MQ Service (CPYMQMSVC)

Where allowed to run:	Threadsafe:
All environments (*ALL)	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

Keyword	Description	Choices	Notes
FROMSVC	From Service	Character value	Required, Key, Positional 1
TOSVC	To Service	Character value	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
STRCMD	Start program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 6
	Qualifier 1: Start program	Name	
	Qualifier 2: Library	Name	
STRARG	Start program arguments	*Character value*, *BLANK, ***SAME**	Optional, Positional 7
ENDCMD	End program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 8
	Qualifier 1: End program	Name	
	Qualifier 2: Library	Name	
ENDARG	End program arguments	*Character value*, *BLANK, ***SAME**	Optional, Positional 9
STDOUT	Standard output	*Character value*, *BLANK, ***SAME**	Optional, Positional 10
STDERR	Standard error	*Character value*, *BLANK, ***SAME**	Optional, Positional 11
TYPE	Service type	***SAME**, *CMD, *SVR	Optional, Positional 12
CONTROL	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.

Examples

None

Error messages

Unknown

Copy MQ Topic (CPYMQMTOP)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Copy MQ Topic (CPYMQMTOP) 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

Keyword	Description	Choices	Notes
FROMTOP	From topic	Character value	Required, Key, Positional 1
TOTOP	To topic	Character value	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
TOPICSTR	Topic string	*Character value*, *BLANK, ***SAME**	Optional, Positional 6
DURSUB	Durable subscriptions	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 7
MGDDURMDL	Durable model queue	*Character value*, *NONE, ***SAME**	Optional, Positional 8
MGDNDURMDL	Non-durable model queue	*Character value*, *NONE, ***SAME**	Optional, Positional 9
PUBENBL	Publish	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 10
SUBENBL	Subscribe	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 11
DFTPTY	Default message priority	0-9, ***SAME**, *ASPARENT	Optional, Positional 12
DFTMSGPST	Default message persistence	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 13
DFTPUTRESP	Default Put Response	***SAME**, *ASPARENT, *SYNC, *ASYNC	Optional, Positional 14
WILDCARD	Wildcard behavior	***SAME**, *PASSTHRU, *BLOCK	Optional, Positional 15
PMSGDLV	Persistent message delivery	***SAME**, *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 16
NPMSGDLV	Non-persistent message deliver	***SAME**, *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 17
CUSTOM	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 (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:

***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 and value pairs must have the form `NAME(VALUE)` and be specified in uppercase. Single quotes must be escaped with another single quote.

Examples

None

Error messages

Unknown

Create Message Queue Manager (CRTMQM)

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

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	Character value	Required, Positional 1
TEXT	Text 'description'	*Character value*, ***BLANK**	Optional, Positional 2
TRGITV	Trigger interval	0-999999999, **999999999**	Optional, Positional 3
UDLMSGQ	Undelivered message queue	*Character value*, ***NONE**	Optional, Positional 4
DFTTMQ	Default transmission queue	*Character value*, ***NONE**	Optional, Positional 5
MAXHDL	Maximum handle limit	0-999999999, **256**	Optional, Positional 6
MAXUMSG	Maximum uncommitted messages	1-999999999, **10000**	Optional, Positional 7
DFTQMGR	Default Queue Manager	*YES, ***NO**	Optional, Positional 8
MQMLIB	Queue Manager Library	*Name*, ***AUTO**	Optional, Positional 9
MQMDIRP	Data Directory Prefix	*Character value*, ***DFT**	Optional, Positional 10
ASP	ASP Number	1-32, ***SYSTEM**, *ASPDEV	Optional, Positional 11
ASPDEV	ASP device	*Character value*, *ASP	Optional, Positional 12
THRESHOLD	Journal receiver threshold	100000-1000000000, ***DFT**, *MIN, *MAX	Optional, Positional 13
JRNBUFSIZ	Journal buffer size	32000-15761440, ***DFT**	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 utilized 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 utilized 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.

Examples

None

Error messages

Unknown

Create MQ AuthInfo object (CRTMQMAUTI)

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
AINAME	AuthInfo name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, *DFT	Required, Key, Positional 2
AUTHTYPE	AuthInfo type	*CRLLDAP, *OCSP, *IDPWOS, *IDPWLDAP	Required, Key, Positional 3
CONNAME	Connection name	*Character value*, *SYSDFTAI	Optional, Positional 4
REPLACE	Replace	***NO**, *YES	Optional, Positional 5
TEXT	Text 'description'	*Character value*, ***SYSDFTAI**, *NONE	Optional, Positional 6
USERNAME	User name	*Character value*, ***SYSDFTAI**, *NONE	Optional, Positional 7
PASSWORD	User password	*Character value*, ***SYSDFTAI**, *NONE	Optional, Positional 8
OCSPURL	OCSP Responder URL	*Character value*, ***SAME**	Optional, Positional 9
CHCKCLNT	Authentication checks required	*ASQMGR, *REQUIRED, *REQADM	Optional, Positional 10
CHCKLOCL	Authentication checks required	*NONE, *OPTIONAL, *REQUIRED, *REQADM	Optional, Positional 11
FAILDELAY	Failure delay	*Integer value*	Optional, Positional 12
BASEDNU	Base user DN	*Character value*, ***SAME**	Optional, Positional 13
ADOPTCTX	Context adoption	*Integer value*	Optional, Positional 14
CLASSUSR	LDAP object class	*Character value*, ***SAME**	Optional, Positional 15

Keyword	Description	Choices	Notes
SHORTUSR	Short user name	*Character value*, ***SAME**	Optional, Positional 16
USRFIELD	User field	*Character value*, ***SAME**	Optional, Positional 17
SECCOMM	LDAP communications	*Character value*, ***SAME**	Optional, Positional 18
AUTHORMD	Authorization method	*Character value*, ***OS** *SEARCHGRP *SEARCHUSR	Optional, Positional 19
BASEDNG	Base DN for groups	*Character value*, ***SAME**	Optional, Positional 20
CLASSGRP	Object class for group	*Character value*, ***SAME**	Optional, Positional 21
FINDGRP	Attribute to find group membership	*Character value*, ***SAME**	Optional, Positional 22
GRPFIELD	Simple name for group	*Character value*, ***SAME**	Optional, Positional 23
NESTGRP	Group nesting	***NO** *YES	Optional, Positional 24
V 8.0.0.3 V 8.0.0.3 AUTHENMD	Authentication method	***OS** 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 **IDPWLDAP*.

Authentication method (AUTHENMD)

V 8.0.0.3

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)

V 8.0.0.2

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 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 all the groups to which the specified user belongs. The attribute to query is defined by the FINDGRP value, typically *memberOf*.

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)

V 8.0.0.2

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)

V 8.0.0.2 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)

V 8.0.0.2 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*.

If left blank, the queue manager uses *member* or *memberOf* depending on the setting of **AUTHORMD**.

This attribute is valid only for an **AUTHTYPE** of **IDPWLDAP*.

Simple name for group (GRPFIELD)

V 8.0.0.2

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)

V 8.0.0.2

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.

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 SSL/TLS

YES
Connectivity to the LDAP server is made securely using SSL/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 SSL/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 SSL/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.

Create MQ Channel (CRTMQMCHL)

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
CHLNAME	Channel name	Character value	Required, Key, Positional 1
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TRPTYPE	Transport type	*LU62, *TCP, ***SYSDFTCHL**	Optional, Positional 5
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTCHL**	Optional, Positional 6
TGTMQMNAME	Target Queue Manager	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 7
CONNAME	Connection name	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 8

Keyword	Description	Choices	Notes
TPNAME	Transaction Program Name	*Character value*, *BLANK, ***SYSDFTCHL**	Optional, Positional 9
MODENAME	Mode Name	*Character value*, *BLANK, ***SYSDFTCHL**	Optional, Positional 10
TMQNAME	Transmission queue	*Character value*, ***SYSDFTCHL**	Optional, Positional 11
MCANAME	Message channel agent	Single values: ***SYSDFTCHL**, *NONE Other values: *Qualified object name*	Optional, Positional 12
	Qualifier 1: Message channel agent	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MCAUSRID	Message channel agent user ID	*Character value*, *NONE, *PUBLIC, ***SYSDFTCHL**	Optional, Positional 13
MCATYPE	Message channel agent Type	*PROCESS, *THREAD, ***SYSDFTCHL**	Optional, Positional 14
BATCHINT	Batch Interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 15
BATCHSIZE	Batch size	1-9999, ***SYSDFTCHL**	Optional, Positional 16
DSCITV	Disconnect interval	0-999999, ***SYSDFTCHL**	Optional, Positional 17
SHORTTMR	Short retry interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 18
SHORTRTY	Short retry count	0-999999999, ***SYSDFTCHL**	Optional, Positional 19
LONGTMR	Long retry interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 20
LONGRTY	Long retry count	0-999999999, ***SYSDFTCHL**	Optional, Positional 21
SCYEXIT	Security exit	Single values: ***SYSDFTCHL**, *NONE Other values: *Qualified object name*	Optional, Positional 22
	Qualifier 1: Security exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSCYEXIT	Security exit	*Character value*, ***SYSDFTCHL**, *NONE	Optional, Positional 23
SCYUSRDATA	Security exit user data	*Character value*, ***SYSDFTCHL**, *NONE	Optional, Positional 24

Keyword	Description	Choices	Notes
SNDEXIT	Send exit	Single values: ***SYSDFTCHL**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 25
	Qualifier 1: Send exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSNDEXIT	Send exit	Single values: ***SYSDFTCHL**, *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 26
SNDUSRDATA	Send exit user data	Values (up to 10 repetitions): *Character value*, ***SYSDFTCHL**, *NONE	Optional, Positional 27
RCVEXIT	Receive exit	Single values: ***SYSDFTCHL**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 28
	Qualifier 1: Receive exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CRCVEXIT	Receive exit	Single values: ***SYSDFTCHL**, *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 29
RCVUSRDATA	Receive exit user data	Values (up to 10 repetitions): *Character value*, ***SYSDFTCHL**, *NONE	Optional, Positional 30
MSGEXIT	Message exit	Single values: ***SYSDFTCHL**, *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 31
	Qualifier 1: Message exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGUSRDATA	Message exit user data	Values (up to 10 repetitions): *Character value*, ***SYSDFTCHL**, *NONE	Optional, Positional 32

Keyword	Description	Choices	Notes
MSGRTYEXIT	Message retry exit	Single values: ***SYSDFTCHL**, *NONE Other values: *Qualified object name*	Optional, Positional 33
	Qualifier 1: Message retry exit	Name	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGRTYDATA	Message retry exit data	*Character value*, ***SYSDFTCHL**, *NONE	Optional, Positional 34
MSGRTYNBR	Number of message retries	0-999999999, ***SYSDFTCHL**	Optional, Positional 35
MSGRTYITV	Message retry interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 36
CVTMSG	Convert message	*YES, *NO, ***SYSDFTCHL**	Optional, Positional 37
PUTAUT	Put authority	*DFT, *CTX, ***SYSDFTCHL**	Optional, Positional 38
SEQNUMWRAP	Sequence number wrap	100-999999999, ***SYSDFTCHL**	Optional, Positional 39
MAXMSGLEN	Maximum message length	0-104857600, ***SYSDFTCHL**	Optional, Positional 40
HRTBTINTVL	Heartbeat interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 41
NPMSPEED	Non Persistent Message Speed	*FAST, *NORMAL, ***SYSDFTCHL**	Optional, Positional 42
CLUSTER	Cluster Name	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 43
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 44
NETPRTY	Network Connection Priority	0-9, ***SYSDFTCHL**	Optional, Positional 45

Keyword	Description	Choices	Notes
SSLCIPH	SSL CipherSpec	*Character value*, '*NULL_MD5', '*NULL_SHA', '*RC4_MD5_EXPORT', '*RC4_MD5_US', '*RC4_SHA_US', '*RC2_MD5_EXPORT', '*DES_SHA_EXPORT', '*TRIPLE_DES_SHA_US', '*AES_SHA_US', '*TLS_RSA_WITH_NULL_MD5', '*TLS_RSA_WITH_NULL_SHA', '*TLS_RSA_EXPORT_WITH_RC4_40_MD5', '*TLS_RSA_WITH_RC4_128_MD5', '*TLS_RSA_WITH_RC4_128_SHA', '*TLS_RSA_EXPORT_WITH_RC2_40_MD5', '*TLS_RSA_WITH_DES_CBC_SHA', '*TLS_RSA_WITH_3DES_EDE_CBC_SHA', '*TLS_RSA_WITH_AES_128_CBC_SHA', '*TLS_RSA_WITH_AES_256_CBC_SHA', *NONE, ***SYSDFTCHL**	Optional, Positional 46 CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA is deprecated.
SSLCAUTH	SSL Client Authentication	*REQUIRED, *OPTIONAL, ***SYSDFTCHL**	Optional, Positional 47
SSLPEER	SSL Peer name	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 48
LOCLADDR	Local communication address	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 49
BATCHHB	Batch Heartbeat Interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 50
USERID	Task user identifier	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 51
PASSWORD	Password	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 52
KAINT	Keep Alive Interval	*Integer*, *AUTO, ***SYSDFTCHL**	Optional, Positional 53
COMPHDR	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, ***SYSDFTCHL**	Optional, Positional 54

Keyword	Description	Choices	Notes
COMPMSG	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST, ***SYSDFTCHL**	Optional, Positional 55
MONCHL	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SYSDFTCHL**	Optional, Positional 56
STATCHL	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SYSDFTCHL**	Optional, Positional 57
CLWLRANK	Cluster Workload Rank	0-9, ***SYSDFTCHL**	Optional, Positional 58
CLWLPRTY	Cluster Workload Priority	0-9, ***SYSDFTCHL**	Optional, Positional 59
CLWLWGHT	Cluster Channel Weight	1-99, ***SYSDFTCHL**	Optional, Positional 60
SHARECNV	Sharing Conversations	0-999999999, ***SYSDFTCHL**	Optional, Positional 61
PROPCTL	Property Control	*COMPAT, *NONE, *ALL, ***SYSDFTCHL**	Optional, Positional 62
MAXINST	Maximum Instances	0-999999999, ***SYSDFTCHL**	Optional, Positional 63
MAXINSTC	Maximum Instances Per Client	0-999999999, ***SYSDFTCHL**	Optional, Positional 64
CLNTWGHT	Client Channel Weight	0-99, ***SYSDFTCHL**	Optional, Positional 65
AFFINITY	Connection Affinity	*PREFERRED, *NONE, ***SYSDFTCHL**	Optional, Positional 66
BATCHLIM	Batch Data Limit	0-999999, ***SYSDFTCHL**	Optional, Positional 67
DFTRECON	Default client reconnetion	*NO, *YES, *QMGR, *DISABLED, ***SYSDFTCHL**	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 AIX, HP-UX, IBM i, Linux, Solaris, and Windows, 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:

***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 (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:

***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 Persistant Messages.

The possible values are:

***SYSDFTCHL**
The value of this attribute does not change.

***FAST**
The channel supports fast non persistant messages.

***NORMAL**
The channel does not support fast non persistant 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 *CLUSSDR 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 specifiifed.

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 *CLUSSDR 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 specififed.

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.

SSL CipherSpec (SSLCIPH)

SSLCIPH specifies the CipherSpec used in SSL 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.

SSL Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel should carry out client authentication over SSL. 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.

SSL Peer name (SSLPEER)

SSLPEER specifies the X500 peer name used in SSL 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 SSL or TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different SSL or 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 (BATCHHB)

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 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:

***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 switched off.

***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 origininating 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.

This 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 Seqence 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 MQCONNX, the client is not reconnected automatically.

***YES**
Unless overridden by MQCONNX, the client reconnects automatically.

***QMGR**
Unless overridden by MQCONNX, 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 MQCONNX MQI call.

This parameter is specified for a client connection channel, (CHLTYPE) *CLTCN

Examples

None

Error messages

Unknown

Create MQ Listener (CRTMQMLSR)

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
LSRNAME	Listener name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Positional 3
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTLSR**	Optional, Positional 4
CONTROL	Listener control	***SYSDFTLSR**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 5
PORT	Port number	0-65535, ***SYSDFTLSR**	Optional, Positional 6
IPADDR	IP Address	*Character value*, *BLANK, ***SYSDFTLSR**	Optional, Positional 7
BACKLOG	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.

Examples

None

Error messages

Unknown

Create MQ Namelist (CRTMQMNL)

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

Keyword	Description	Choices	Notes
NAMELIST	Namelist	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Positional 3
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTNL**	Optional, Positional 4
NAMES	List of Names	Values (up to 256 repetitions): *Character value*, *BLANKS, ***SYSDFTNL**, *NONE	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.

Examples

None

Error messages

Unknown

Create MQ Process (CRTMQMPRC)

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
PRCNAME	Process name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Positional 3
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTPRC**	Optional, Positional 4
APPTYPE	Application type	*Integer*, ***SAME**, *CICS, *MVS, *IMS, *OS2, *DOS, *UNIX, *QMGR, *OS400, *WINDOWS, *CICS_VSE, *WINDOWS_NT, *VMS, *NSK, *VOS, *IMS_BRIDGE, *XCF, *CICS_BRIDGE, *NOTES_AGENT, *BROKER, *JAVA, *DQM	Optional, Positional 5
APPID	Application identifier	*Character value*, ***SYSDFTPRC**	Optional, Positional 6
USRDATA	User data	*Character value*, ***SYSDFTPRC**, *NONE	Optional, Positional 7
ENVDATA	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:

***SYSDFTPRC**
The value for this attribute is taken from the system default process.

***CICS**
Represents a CICS/400 application.

***MVS**
Represents an MVS application.

***IMS**
Represents an IMS application.

***OS2**
Represents an OS/2 application.

***DOS**
Represents a DOS application.

***UNIX**
Represents a UNIX application.

***QMGR**
Represents a queue manager.

***OS400**
Represents an IBM i application.

***WINDOWS**
Represents a Windows application.

***CICS_VSE**
Represents a CICS/VSE application.

***WINDOWS_NT**
Represents a Windows NT application.

***VMS**
Represents a VMS application.

***NSK**
Represents a Tandem/NSK application.

***VOS**
Represents a VOS application.

***IMS_BRIDGE**
Represents an IMS bridge application.

***XCF**
Represents an XCF application.

***CICS_BRIDGE**
Represents a CICS bridge application.

***NOTES_AGENT**
Represents a Lotus Notes application.

***BROKER**
Represents a broker application.

***JAVA**
Represents a Java application.

***DQM**
Represents a DQM application.

user-value
User-defined application type in the range 65536 through 999999999.

The values within this range are not tested, and any other value is accepted.

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.

Examples

None

Error messages

Unknown

Create MQ Queue (CRTMQMQ)

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

Keyword	Description	Choices	Notes
QNAME	Queue name	Character value	Required, Key, Positional 1
QTYPE	Queue type	*ALS, *LCL, *MDL, *RMT	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTQ**	Optional, Positional 5
PUTENBL	Put enabled	***SYSDFTQ**, *NO, *YES	Optional, Positional 6
DFTPTY	Default message priority	0-9, ***SYSDFTQ**	Optional, Positional 7
DFTMSGPST	Default message persistence	***SYSDFTQ**, *NO, *YES	Optional, Positional 8
PRCNAME	Process name	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 9
TRGENBL	Triggering enabled	***SYSDFTQ**, *NO, *YES	Optional, Positional 10
GETENBL	Get enabled	***SYSDFTQ**, *NO, *YES	Optional, Positional 11
SHARE	Sharing enabled	***SYSDFTQ**, *NO, *YES	Optional, Positional 12
DFTSHARE	Default share option	***SYSDFTQ**, *NO, *YES	Optional, Positional 13

Keyword	Description	Choices	Notes
MSGDLYSEQ	Message delivery sequence	***SYSDFTQ**, *PTY, *FIFO	Optional, Positional 14
HDNBKTCNT	Harden backout count	***SYSDFTQ**, *NO, *YES	Optional, Positional 15
TRGTYPE	Trigger type	***SYSDFTQ**, *FIRST, *ALL, *DEPTH, *NONE	Optional, Positional 16
TRGDEPTH	Trigger depth	1-999999999, ***SYSDFTQ**	Optional, Positional 17
TRGMSGPTY	Trigger message priority	0-9, ***SYSDFTQ**	Optional, Positional 18
TRGDATA	Trigger data	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 19
RTNITV	Retention interval	0-999999999, ***SYSDFTQ**	Optional, Positional 20
MAXDEPTH	Maximum queue depth	0-999999999, ***SYSDFTQ**	Optional, Positional 21
MAXMSGLEN	Maximum message length	0-104857600, ***SYSDFTQ**	Optional, Positional 22
BKTTHLD	Backout threshold	0-999999999, ***SYSDFTQ**	Optional, Positional 23
BKTQNAME	Backout requeue name	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 24
INITQNAME	Initiation queue	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 25
USAGE	Usage	***SYSDFTQ**, *NORMAL, *TMQ	Optional, Positional 26
DFNTYPE	Definition type	***SYSDFTQ**, *TEMPDYN, *PERMDYN	Optional, Positional 27
TGTQNAME	Target object	*Character value*, ***SYSDFTQ**	Optional, Positional 28
RMTQNAME	Remote queue	*Character value*, ***SYSDFTQ**, *NONE	Optional, Positional 29
RMTMQMNAME	Remote Message Queue Manager	*Character value*, ***SYSDFTQ**	Optional, Positional 30
TMQNAME	Transmission queue	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 31
HIGHTHLD	Queue depth high threshold	0-100, ***SYSDFTQ**	Optional, Positional 32
LOWTHLD	Queue depth low threshold	0-100, ***SYSDFTQ**	Optional, Positional 33
FULLEVT	Queue full events enabled	***SYSDFTQ**, *NO, *YES	Optional, Positional 34
HIGHEVT	Queue high events enabled	***SYSDFTQ**, *NO, *YES	Optional, Positional 35

Keyword	Description	Choices	Notes
LOWEVT	Queue low events enabled	***SYSDFTQ**, *NO, *YES	Optional, Positional 36
SRVITV	Service interval	0-999999999, ***SYSDFTQ**	Optional, Positional 37
SRVEVT	Service interval events	***SYSDFTQ**, *HIGH, *OK, *NONE	Optional, Positional 38
DISTLIST	Distribution list support	***SYSDFTQ**, *NO, *YES	Optional, Positional 39
CLUSTER	Cluster Name	*Character value*, ***SYSDFTQ**, *NONE	Optional, Positional 40
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 41
DEFBIND	Default Binding	***SYSDFTQ**, *OPEN, *NOTFIXED, *GROUP	Optional, Positional 42
CLWLRANK	Cluster Workload Rank	0-9, ***SYSDFTQ**	Optional, Positional 43
CLWLPRTY	Cluster Workload Priority	0-9, ***SYSDFTQ**	Optional, Positional 44
CLWLUSEQ	Cluster workload queue use	***SYSDFTQ**, *QMGR, *LOCAL, *ANY	Optional, Positional 45
MONQ	Queue Monitoring	***SYSDFTQ**, *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 46
STATQ	Queue Statistics	***SYSDFTQ**, *QMGR, *OFF, *ON	Optional, Positional 47
ACCTQ	Queue Accounting	***SYSDFTQ**, *QMGR, *OFF, *ON	Optional, Positional 48
NPMCLASS	Non Persistent Message Class	***SYSDFTQ**, *NORMAL, *HIGH	Optional, Positional 49
MSGREADAHD	Message Read Ahead	***SYSDFTQ**, *DISABLED, *NO, *YES	Optional, Positional 50
DFTPUTRESP	Default Put Response	*SYSDFTQ, ***SYNC**, *ASYNC	Optional, Positional 51
PROPCTL	Property Control	***SYSDFTQ**, *COMPAT, *NONE, *ALL, *FORCE, *V6COMPAT	Optional, Positional 52
TARGTYPE	Target Type	***SYSDFTQ**, *QUEUE, *TOPIC	Optional, Positional 53
CUSTOM	Custom attribute	*Character value*, *BLANK, ***SYSDFTQ**	Optional, Positional 54
CLCHNAME	Cluster-sender channel name	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 55

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 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.

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 switched off.

***ON**
Statistics data collection is switched on 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 switched off.

***ON**
Accounting data collection is switched on 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 and 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 and 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`.

Create MQ Subscription (CRTMQMSUB)

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
SUBNAME	Subscription name	Character value	Required, Key, Positional 1

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Key, Positional 3
TOPICSTR	Topic string	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 4
TOPICOBJ	Topic object	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 5
DEST	Destination	*Character value*, ***SYSDFTSUB**	Optional, Positional 6
DESTMQM	Destination Queue Manager	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 7
DESTCRRLID	Destination Correlation Id	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 8
PUBACCT	Publish Accounting Token	*Character value*, *CURRENT, ***SYSDFTSUB**	Optional, Positional 9
PUBAPPID	Publish Application Id	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 10
SUBUSER	Subscription User Id	*Character value*, *CURRENT, ***SYSDFTSUB**	Optional, Positional 11
USERDATA	Subscription User Data	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 12
SELECTOR	Selector String	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 13
PSPROP	PubSub Property	***SYSDFTSUB**, *NONE, *COMPAT, *RFH2, *MSGPROP	Optional, Positional 14
DESTCLASS	Destination Class	***SYSDFTSUB**, *MANAGED, *PROVIDED	Optional, Positional 15
SUBSCOPE	Subscription Scope	***SYSDFTSUB**, *ALL, *QMGR	Optional, Positional 16
VARUSER	Variable User	***SYSDFTSUB**, *ANY, *FIXED	Optional, Positional 17
REQONLY	Request Publications	***SYSDFTSUB**, *YES, *NO	Optional, Positional 18
PUBPTY	Publish Priority	0-9, ***SYSDFTSUB**, *ASPUB, *ASQDEF	Optional, Positional 19
WSCHEMA	Wildcard Schema	***SYSDFTSUB**, *TOPIC, *CHAR	Optional, Positional 20
EXPIRY	Expiry Time	0-999999999, ***SYSDFTSUB**, *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 (DESTCRRLID)

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 Version 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 (WSCHEMA)

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.

Examples

None

Error messages

Unknown

Create MQ Service (CRTMQMSVC)

Where allowed to run:	Threadsafe:
All environments (*ALL)	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

Keyword	Description	Choices	Notes
SVCNAME	Service name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Positional 3
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTSVC**	Optional, Positional 4
STRCMD	Start program	Single values: ***SYSDFTSVC**, *NONE Other values: *Qualified object name*	Optional, Positional 5
	Qualifier 1: Start program	Name	
	Qualifier 2: Library	Name	
STRARG	Start program arguments	*Character value*, *BLANK, ***SYSDFTSVC**	Optional, Positional 6
ENDCMD	End program	Single values: ***SYSDFTSVC**, *NONE Other values: *Qualified object name*	Optional, Positional 7
	Qualifier 1: End program	Name	
	Qualifier 2: Library	Name	
ENDARG	End program arguments	*Character value*, *BLANK, ***SYSDFTSVC**	Optional, Positional 8
STDOUT	Standard output	*Character value*, *BLANK, ***SYSDFTSVC**	Optional, Positional 9
STDERR	Standard error	*Character value*, *BLANK, ***SYSDFTSVC**	Optional, Positional 10
TYPE	Service type	***SYSDFTSVC**, *CMD, *SVR	Optional, Positional 11
CONTROL	Service control	***SYSDFTSVC**, *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.

Examples

None

Error messages

Unknown

Create MQ Topic (CRTMQMTOP)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Create MQ Topic (CRTMQMTOP) command creates a new MQ topic object, specifying those attributes that are different from the default.

Parameters

Keyword	Description	Choices	Notes
TOPNAME	Topic name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Positional 3
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTTOP**	Optional, Positional 4
TOPICSTR	Topic string	*Character value*, *BLANK, ***SYSDFTTOP**	Optional, Positional 5
DURSUB	Durable subscriptions	***SYSDFTTOP**, *ASPARENT, *YES, *NO	Optional, Positional 6

Keyword	Description	Choices	Notes
MGDDURMDL	Durable model queue	*Character value*, *NONE, ***SYSDFTTOP**	Optional, Positional 7
MGDNDURMDL	Non-durable model queue	*Character value*, *NONE, ***SYSDFTTOP**	Optional, Positional 8
PUBENBL	Publish	***SYSDFTTOP**, *ASPARENT, *YES, *NO	Optional, Positional 9
SUBENBL	Subscribe	***SYSDFTTOP**, *ASPARENT, *YES, *NO	Optional, Positional 10
DFTPTY	Default message priority	0-9, ***SYSDFTTOP**, *ASPARENT	Optional, Positional 11
DFTMSGPST	Default message persistence	***SYSDFTTOP**, *ASPARENT, *YES, *NO	Optional, Positional 12
DFTPUTRESP	Default Put Response	***SYSDFTTOP**, *ASPARENT, *SYNC, *ASYNC	Optional, Positional 13
WILDCARD	Wildcard behavior	***SYSDFTTOP**, *PASSTHRU, *BLOCK	Optional, Positional 14
PMSGDLV	Persistent message delivery	***SYSDFTTOP**, *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 15
NPMSGDLV	Non-persistent message deliver	***SYSDFTTOP**, *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 16
CUSTOM	Custom attribute	*Character value*, *BLANK, ***SYSDFTTOP**	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 and value pairs must have the form NAME(VALUE) and be specified in uppercase. Single quotes must be escaped with another single quote.

Examples

None

Error messages

Unknown

Convert MQ Data Type (CVTMQMDTA)

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

Keyword	Description	Choices	Notes
FROMFILE	Input file	Qualified object name	Required, Positional 1
	Qualifier 1: Input file	Name	
	Qualifier 2: Library	*Name*, ***LIBL**, *CURLIB	
FROMMBR	Member containing input	Name	Required, Positional 2
TOFILE	File to receive output	Qualified object name	Required, Positional 3
	Qualifier 1: File to receive output	Name	
	Qualifier 2: Library	*Name*, ***LIBL**, *CURLIB	
TOMBR	Member to receive output	*Name*, ***FROMMBR**	Optional, Positional 4
RPLTOMBR	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.

Examples

None

Error messages

Unknown

Delete Message Queue Manager (DLTMQM)

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
MQMNAME	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).

Examples

None

Error messages

Unknown

Delete MQ AuthInfo object (DLTMQMAUTI)

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
AINAME	AuthInfo name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Delete MQ Pub/Sub Broker (DLTMQMBRK)

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
MQMNAME	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.

Examples

None

Error messages

Unknown

Delete MQ Channel (DLTMQMCHL)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Delete MQ Channel (DLTMQMCHL) command deletes the specified channel definition.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, ***NONCLT**, *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

Examples

None

Error messages

Unknown

Delete MQ Listener (DLTMQMLSR)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Delete MQ Listener object (DSPMQMLSR) command deletes an existing MQ listener object.

Parameters

Keyword	Description	Choices	Notes
LSRNAME	Listener name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Delete MQ Namelist (DLTMQMNL)

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

Keyword	Description	Choices	Notes
NAMELIST	Namelist	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Delete MQ Process (DLTMQMPRC)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Delete MQ Process (DLTMQMPRC) command deletes an existing MQ process definition.

Parameters

Keyword	Description	Choices	Notes
PRCNAME	Process name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Delete MQ Queue (DLTMQMQ)

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

Keyword	Description	Choices	Notes
QNAME	Queue name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Delete MQ Subscription (DLTMQMSUB)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Delete MQ Subscription (DLTMQMSUB) command deletes an existing MQ subscription.

Parameters

Keyword	Description	Choices	Notes
SUBID	Subscription identifier	*Character value*, ***NONE**	Optional, Positional 1
SUBNAME	Subscription name	*Character value*, ***NONE**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Delete MQ Service (DLTMQMSVC)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Delete MQ Service object (DLTMQMSVC) command deletes an existing MQ service object.

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Delete MQ Topic (DLTMQMTOP)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Delete MQ Topic (DLTMQMTOP) command deletes an existing MQ topic object.

Parameters

Keyword	Description	Choices	Notes
TOPNAME	Topic name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Dump MQ Configuration (DMPMQMCFG)

Where allowed to run:	Threadsafe:
All environments (*ALL)	Yes

The Dump MQ Configuration (DMPMQMCFG) command is used to dump the configuration objects and authorities for a queue manager.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***ALL**	Optional, Positional 1
OBJ	Object name	*Character value*, ***ALL**	Optional, Positional 2
OBJTYPE	Object type	***ALL**, *AUTHINFO, *CHL, *CLTCN, *COMMINFO, *LSR, *NMLIST, *PRC, *Q, *MQM, *SVC, *SUB, *TOPIC	Optional, Positional 3
EXPTYPE	Export type	***ALL**, *OBJECT, *AUTHREC, *CHLAUTH	Optional, Positional 4
EXPATTR	Export attributes	***NONDEF**, *ALL	Optional, Positional 5
WARN	Warnings	***NO**, *YES	Optional, Positional 6
OUTPUT	Output	***MQSC**, *ONELINE, *SETMQAUT, *GRTMQMAUT	Optional, Positional 7
CLIENT	Client connection	***NO**, *YES, *CHL	Optional, Positional 8
CLIENTCHL	MQSC Channel Definition	*Character value*, ***NONE**	Optional, Positional 9
MSGSEQNUM	Message sequence number	1-999999999, ***NORESET**	Optional, Positional 10
RPLYQ	Reply Queue	*Character value*, **'SYSTEM.DEFAULT.MODEL.QUEUE'**	Optional, Positional 11
RMTMQMNAME	Remote Message Queue Manager	*Character value*, ***NONE**	Optional, Positional 12
TOFILE	File to receive output	*Qualified object name*	Optional, Positional 13
	Qualifier 1: File to receive output	*Name*	
	Qualifier 2: Library	*Name*, ***LIBL**	
TOMBR	Member to receive output	*Name*	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 Windows or UNIX platforms.

***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 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 Preparing queue managers for remote administration and Preparing channels and transmission queues 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.

Error messages

Unknown

Disconnect MQ (DSCMQM)

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

Examples

None

Error messages

Unknown

Display Message Queue Manager (DSPMQM)

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

Keyword	Description	Choices	Notes
OUTPUT	Output	*****, *PRINT	Optional, Positional 1
MQMNAME	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).

Examples

None

Error messages

Unknown

Display MQ Object Authority (DSPMQMAUT)

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

Keyword	Description	Choices	Notes
OBJ	Object name	Character value	Required, Positional 1
OBJTYPE	Object type	*Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC, *RMTMQMNAME	Required, Positional 2
USER	User name	*Name*, ***PUBLIC**	Optional, Positional 3
OUTPUT	Output	*****, *PRINT	Optional, Positional 4

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 5
SRVCOMP	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.

Display MQ AuthInfo object (DSPMQMAUTI)

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

Keyword	Description	Choices	Notes
AINAME	AuthInfo name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
OUTPUT	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.

Examples

None

Error messages

Unknown

Display MQ Pub/Sub Broker (DSPMQMBRK)

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

Keyword	Description	Choices	Notes
MQMNAME	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.

Examples

None

Error messages

Unknown

Display MQ Channel (DSPMQMCHL)

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

Keyword	Description	Choices	Notes
CHLNAME	Channel name	Character value	Required, Positional 1
OUTPUT	Output	*****, *PRINT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, ***NONCLT**, *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

Examples

None

Error messages

Unknown

Display MQ Command Server (DSPMQMCSVR)

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

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Display MQ Listener (DSPMQMLSR)

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
LSRNAME	Listener name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
OUTPUT	Output	*****, *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.

Examples

None

Error messages

Unknown

Display MQ Namelist (DSPMQMNL)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Display MQ Namelist (DSPMQMNL) command displays an MQ namelist.

Parameters

Keyword	Description	Choices	Notes
NAMELIST	Namelist	Character value	Required, Positional 1
OUTPUT	Output	*****, *PRINT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Display MQ Object Names (DSPMQMOBJN)

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
OBJ	Object name	*Character value*, *ALL	Required, Positional 1
OBJTYPE	Object type	***ALLMQM**, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *CHL, *CLTCN, *SVC, *MQM, *NMLIST, *PRC, *LSR, *TOPIC	Optional, Positional 2
OUTPUT	Output	*****, *PRINT	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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 displayed.

The possible values are:

***DFT**
The default queue manager.

queue-manager-name
Specify the name of the queue manager.

Examples

None

Error messages

Unknown

Display MQ Process (DSPMQMPRC)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Display MQ Process (DSPMQMPRC) command displays the attributes of an existing MQ process definition.

Parameters

Keyword	Description	Choices	Notes
PRCNAME	Process name	Character value	Required, Positional 1
OUTPUT	Output	*****, *PRINT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Display MQ Queue (DSPMQMQ)

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
QNAME	Queue name	Character value	Required, Positional 1
OUTPUT	Output	*****, *PRINT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Display MQ Route Information (DSPMQMRTE)

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

Keyword	Description	Choices	Notes
QNAME	Target object	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
CRRLID	Correlation Identifier	*Character value*, ***NONE**	Optional, Positional 3
MSGPST	Message Persistence	*YES, ***NO**, *QUEUE	Optional, Positional 4
MSGPRTY	Message Priority	0-9, ***QUEUE**	Optional, Positional 5
OPTION	Report Option	Single values: ***DFT**, *NONE Other values (up to 6 repetitions): *ACTIVITY, *COA, *COD, *DISCARD, *EXCEPTION, *EXPIRATION	Optional, Positional 6
RPLYQ	Reply Queue	*Character value*, ***DFT**	Optional, Positional 7
RPLYMQM	Reply Queue Manager	*Character value*, ***DFT**	Optional, Positional 8
EXPIRY	Message Expiry	0-999999999, ***DFT**	Optional, Positional 9
EXPRPT	Pass Expiry	***YES**, *NO	Optional, Positional 10
RTEINF	Route Accumulation	*YES, ***NO**	Optional, Positional 11
RPLYMSG	Reply Message	*YES, ***NO**	Optional, Positional 12
DLVRMSG	Deliver Message	*YES, ***NO**	Optional, Positional 13
FWDMSG	Forward Message	***SUPPORT**, *ALL	Optional, Positional 14
MAXACTS	Maximum Activities	1-999999999, ***NOMAX**	Optional, Positional 15
DETAIL	Route Detail	*LOW, ***MEDIUM**, *HIGH	Optional, Positional 16
BROWSE	Browse Only	*YES, ***NO**	Optional, Positional 17
DSPMSG	Display Message	***YES**, *NO	Optional, Positional 18
TGTMQM	Target Queue Manager	*Character value*, ***DFT**	Optional, Positional 19

Keyword	Description	Choices	Notes
DSPINF	Display Information	Single values: ***ALL**, *SUMMARY, *NONE Other values (up to 6 repetitions): *ACTGRP, *ID, *MSGGRP, *MSGDELTA, *OPGRP, *TRGRP	Optional, Positional 20
WAIT	Wait Time	0-999999999, ***DFT**	Optional, Positional 21
BIND	Bind Option	***OPEN**, *NOTFIXED	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 successfullly 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 numer 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.

Examples

None

Error messages

Unknown

Display Queue Manager Status (DSPMQMSTS)

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
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 1
OUTPUT	Output	*****, *PRINT	Optional, Positional 2
“STARTDA” on page 480	Start Date		Optional, Positional 3
“STARTTI” on page 480	Start Time		Optional, Positional 4

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.

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).

Examples

Error messages

Unknown

Display MQ Service (DSPMQMSVC)

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
SVCNAME	Service name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
OUTPUT	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.

Examples

None

Error messages

Unknown

Display MQM Security Policy (DSPMQMSPL)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Display MQM Security Policies (DSPMQMSPL) command displays security policies, that are used by IBM MQ 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
OUTPUT	Output	*, *PRINT	Optional, Positional 1

Keyword	Description	Choices	Notes
POLICY	Policy name	Character value	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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).

Display MQ Subscription (DSPMQMSUB)

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
SUBID	Subscription identifier	*Character value*, ***NONE**	Optional, Positional 1
SUBNAME	Subscription name	*Character value*, ***NONE**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
OUTPUT	Output	*****, *PRINT	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.

Examples

None

Error messages

Unknown

Display MQ Topic (DSPMQMTOP)

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

Keyword	Description	Choices	Notes
TOPNAME	Topic name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
OUTPUT	Output	*****, *PRINT	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.

Examples

None

Error messages

Unknown

Display MQ Version (DSPMQMVER)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Display MQ Version (DSPMQMVER) command provides the current MQ version.

Parameters

Keyword	Description	Choices	Notes
OUTPUT	Output	*****, *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.

Examples

None

Error messages

Unknown

End Message Queue Manager (ENDMQM)

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 CHGCMDDFT (Change Command Default) command.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
OPTION	Option	***CNTRLD**, *IMMED, *WAIT, *PREEMPT	Optional, Positional 2
INSTANCE	Instance To End	***ALL**, *STANDBY	Optional, Positional 3
ALWSWITCH	Allow Switchover	***NO**, *YES	Optional, Positional 4
RECONN	Reconnect	***NO**, *YES	Optional, Positional 5
ENDCCTJOB	End connected jobs	***NO**, *YES	Optional, Positional 6

Keyword	Description	Choices	Notes
RCDMQMIMG	Record MQ Object Image	*NO, ***YES**	Optional, Positional 7
TIMEOUT	Timeout interval (seconds)	0-3600, **30**	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 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 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.

Examples

None

Error messages

Unknown

End MQ Pub/Sub Broker (ENDMQMBRK)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The End IBM MQ Broker (ENDMQMBRK) command is used to stop a broker.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	Character value	Required, Positional 1
OPTION	Option	***CNTRLD**, *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.

Examples

None

Error messages

Unknown

End MQ Channel (ENDMQMCHL)

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
CHLNAME	Channel name	Character value	Required, Positional 1
OPTION	Option	***CNTRLD**, *IMMED, *ABNORMAL	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
STATUS	Channel status	***STOPPED**, *INACTIVE	Optional, Positional 4
CONNAME	Connection name	*Character value*, ***NONE**	Optional, Positional 5
RQMNAME	Remote queue manager	*Character value*, ***NONE**	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 succesful 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.

Examples

None

Error messages

Unknown

End Queue Manager Connection (ENDMQMCONN)

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

Keyword	Description	Choices	Notes
CONN	Connection Identifier	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

End MQ Command Server (ENDMQMCSVR)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The End MQ Command Server (ENDMQMCSVR) command stops the MQ command server for the specified local queue manager.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	Character value	Required, Positional 1
OPTION	Option	***CNTRLD**, *IMMED	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.

Examples

None

Error messages

Unknown

End MQ Listeners (ENDMQMLSR)

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
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
PORT	Port number	1-65535, *ALL	Optional, Positional 2
OPTION	Option	***CNTRLD**, *WAIT, *FORCE	Optional, Positional 3
LSRNAME	Listener name	*Character value*, ***NONE**	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.

Examples

None

Error messages

Unknown

End MQ Service (ENDMQMSVC)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The End MQ Service (ENDMQMSVC) command ends an MQ service.

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	Character value	Required, Positional 1
MQMNAME	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.

Examples

None

Error messages

Unknown

Grant MQ Object Authority (GRTMQMAUT)

Where allowed to run:	Threadsafe:
All environments (*ALL)	Yes

The Grant MQ Authority (GRTMQMAUT) 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 GRTMQMAUT 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

Keyword	Description	Choices	Notes
OBJ	Object name	Character value	Required, Positional 1
OBJTYPE	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC, *RMTMQMNAME	Required, Positional 2
USER	User names	Single values: *PUBLIC, Other values (up to 50 repetitions): *Name*	Required, Positional 3

Keyword	Description	Choices	Notes
AUT	Authority	Values (up to 22 repetitions): *ALTUSR, *BROWSE, *CONNECT, *GET, *INQ, *PUT, *SET, *PUB, *SUB, *RESUME, *PASSALL, *PASSID, *SETALL, *SETID, *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *ALL, *ALLADM, *ALLMQI, *NONE, *CTRL, *CTRLX, *SYSTEM	Required, Positional 4
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 5
SRVCOMP	Service Component name	*Character value*, ***DFT**	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, *ADMCRT, *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.

***ADMCRT**
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).

***ADMCRT**
Create objects of the specified type.

***ADMDLT**
Delete the specified object.

***ADMDSP**
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.

Examples

None

Error messages

Unknown

Ping MQ Channel (PNGMQMCHL)

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
CHLNAME	Channel name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
DATACNT	Data count	16-32768, **64**	Optional, Positional 3
CNT	Count	1-16, **1**	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.

Examples

None

Error messages

Unknown

Record MQ Object Image (RCDMQMIMG)

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
OBJ	Object name	*Character value*, *ALL	Required, Positional 1
OBJTYPE	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *MQM, *NMLIST, *PRC, *CHL, *CLTCN, *LSR, *SVC, *SYNCFILE, *TOPIC	Required, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
DSPJRNDTA	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.

Examples

None

Error messages

Unknown

Re-create MQ Object (RCRMQMOBJ)

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

Keyword	Description	Choices	Notes
OBJ	Object name	*Character value*, *ALL	Required, Positional 1
OBJTYPE	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *MQM, *NMLIST, *PRC, *CHL, *CLTCN, *LSR, *SVC, *SYNCFILE, *CLCHLTAB, *TOPIC	Required, Positional 2
MQMNAME	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.

Examples

None

Error messages

Unknown

Refresh IBM MQ Authority (RFRMQMAUT)

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

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
TYPE	Refresh Type	***AUTHSERV**, *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 SSL 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.

Examples

None

Error messages

Unknown

Refresh MQ Cluster (RFRMQMCL)

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

Keyword	Description	Choices	Notes
CLUSTER	Cluster Name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
REPOS	Refresh Repository	***NO**, *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.

Examples

None

Error messages

Unknown

V 8.0.0.4 Refresh Message Queue Manager (RFRMQM)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Refresh Message Queue manager (RFRMQM) performs special operations on queue managers.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Required, Positional 1
TYPE	Refresh Type	***CONFIGEV**, **PROXYSUB*	Required, Positional 2
OBJECT	Object Type	***ALL**, *Specified objects*	Optional, Positional 3
NAME	Object Name	***ALL**, *generic- object-name*, *object-name*	Optional, Positional 4
INCLINT	Include Interval	***NONE**, *include- interval*	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).

Remove Queue Manager Info. (RMVMQMINF)

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
MQMNAME	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).

Examples

None

Error messages

Unknown

Remove Queue Manager Journal (RMVMQMJRN)

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
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
JRN	Queue Manager Journal	*Character value*, ***DFT**	Optional, Positional 2
RMTJRNRDB	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.

Examples

None

Error messages

Unknown

Resume Cluster Queue Manager (RSMMQMCLQM)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

Use the RSMMQMCLQM 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
CLUSTER	Cluster Name	Character value	Optional, Positional 1
CLUSNL	Cluster Name List	Character value	Optional, Positional 2

Keyword	Description	Choices	Notes
MQMNAME	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.

Examples

None

Error messages

Unknown

Reset MQ Channel (RSTMQMCHL)

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 uset 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
CHLNAME	Channel name	Character value	Required, Positional 1

Keyword	Description	Choices	Notes
MSGSEQNUM	Message sequence number	1-999999999, **1**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Reset Cluster (RSTMQMCL)

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
CLUSTER	Cluster Name	Character value	Required, Positional 1
QMNAME	Queue Manager Name for removal	*Character value*, *QMID	Required, Positional 2
ACTION	Action	*FRCRMV	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4
QUEUES	Remove Queues	***NO**, *YES	Optional, Positional 5
QMID	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.

Examples

None

Error messages

Unknown

Resolve MQ Channel (RSVMQMCHL)

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

Keyword	Description	Choices	Notes
CHLNAME	Channel name	Character value	Required, Positional 1
OPTION	Resolve option	*CMT, *BCK	Required, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***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.

Examples

None

Error messages

Unknown

RUNMQSC (RUNMQSC)

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

Keyword	Description	Choices	Notes
MQMNAME	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.

Examples

None

Error messages

Unknown

Revoke MQ Object Authority (RVKMQMAUT)

Where allowed to run:	Threadsafe:
All environments (*ALL)	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

Keyword	Description	Choices	Notes
OBJ	Object name	Character value	Required, Positional 1
OBJTYPE	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC, *RMTMQMNAME	Required, Positional 2
USER	User names	Single values: *PUBLIC, Other values (up to 50 repetitions): *Name*	Required, Positional 3
AUT	Authority	Values (up to 22 repetitions): *ALTUSR, *BROWSE, *CONNECT, *GET, *INQ, *PUT, *SET, *PUB, *SUB, *RESUME, *PASSALL, *PASSID, *SETALL, *SETID, *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *ALL, *ALLADM, *ALLMQI, *REMOVE, *CTRL, *CTRLX, *SYSTEM	Required, Positional 4
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 5
SRVCOMP	Service Component name	*Character value*, ***DFT**	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, *ADMCRT, *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.

***ADMCRT**
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).

***ADMCRT**
Create objects of the specified type.

***ADMDLT**
Delete the specified object.

***ADMDSP**
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.

Examples

None

Error messages

Unknown

Set MQM Security Policy (SETMQMSPL)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Set MQM Security Policy (SETMQMSPL) command sets security policies, that are used by IBM MQ 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
POLICY	Policy name	Character value	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, *DFT	Required, Key, Positional 2
SIGNAL	Signal algorithm	*NONE, *MD5, *SHA1, *SHA256, *SHA384, *SHA512	Optional, Positional 3
ENCALG	Encryption algorithm	*NONE, *RC2, *DES, *TRIPLEDES, *AES128, *AES256	Optional, Positional 4
SIGNER	Authorized signers	***NONE**, *Character value*	Optional, Positional 5
RECIP	Intended recipients	***NONE**, *Character value*	Optional, Positional 6
TOLERATE	Tolerate unprotected	***NO**, *YES	Optional, Positional 7
REMOVE	Remove policy	***NO**, *YES	Optional, Positional 8

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.

Signal algorithm (SIGNALG)

Specifies the digital signature algorithm from one of the following values:

***NONE**
Messages are not signed.

***MD5**
Messages are signed using the MD5 message digest algorithm.

***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.

***RC2**
Messages are encrypted using the RC2 Rivest Cipher algorithm.

***DES**
Messages are encrypted using the DES Data Encryption Standard algorithm.

***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).

Suspend Cluster Queue Manager (SPDMQMCLQM)

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

Keyword	Description	Choices	Notes
CLUSTER	Cluster Name	Character value	Optional, Positional 1
CLUSNL	Cluster Name List	Character value	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
MODE	Mode	***QUIESCE**, *FORCE	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.

Examples

None

Error messages

Unknown

Start Message Queue Manager (STRMQM)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Start Message Queue Manager (STRMQM) command starts the local queue manager.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
RDEFSYS	Redefine system objects	*YES, ***NO**	Optional, Positional 2
FIXDIRS	Fix directories	*YES, ***NO**	Optional, Positional 3
STRSTSDTL	Startup Status Detail	***ALL**, *MIN	Optional, Positional 4
STRSVC	Service startup	***YES**, *NO	Optional, Positional 5
REPLAY	Perform replay only	*YES, ***NO**	Optional, Positional 6
ACTIVATE	Activate backup	*YES, ***NO**	Optional, Positional 7
STANDBY	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 (STRSTSDTL)

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.

Examples

None

Error messages

Unknown

Start MQ Pub/Sub Broker (STRMQMBRK)

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

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	Character value	Required, Positional 1
PARENTMQM	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 CLRMQMBRK 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.

Start MQ Channel (STRMQMCHL)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Start MQ Channel (STRMQMCHL) command starts an MQ channel.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	Character value	Required, Positional 1
MQMNAME	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.

Examples

None

Error messages

Unknown

Start MQ Channel Initiator (STRMQMCHLI)

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
QNAME	Queue name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Start MQ Command Server (STRMQMCSVR)

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
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Start IBM MQ DLQ Handler (STRMQMDLQ)

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

Keyword	Description	Choices	Notes
UDLMSGQ	Undelivered message queue	*Character value*, *DFT, *NONE	Required, Positional 1
SRCMBR	Member containing input	*Name*, *FIRST	Required, Positional 2
SRCFILE	Input file	Qualified object name	Optional, Positional 3
	Qualifier 1: Input file	*Name*, **QTXTSRC**	
	Qualifier 2: Library	*Name*, ***LIBL**, *CURLIB	
MQMNAME	Message Queue Manager name	*Character value*, ***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.

Examples

None

Error messages

Unknown

Start MQ Listener (STRMQMLSR)

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
PORT	Port number	1-65535, ***DFT**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
IPADDR	IP Address	*Character value*, ***DFT**	Optional, Positional 3
BACKLOG	Listener backlog	0-999999999, ***DFT**	Optional, Positional 4
LSRNAME	Listener name	*Character value*, ***NONE**	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.

Examples

None

Error messages

Unknown

Start IBM MQ Commands (STRMQMMQSC)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Start IBM MQ 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
SRCMBR	Member containing input	*Name*, *FIRST	Required, Positional 1
SRCFILE	Input file	Qualified object name	Optional, Positional 2
	Qualifier 1: Input file	*Name*, **QMQSC**	
	Qualifier 2: Library	*Name*, ***LIBL**, *CURLIB	
OPTION	Option	***RUN**, *VERIFY, *MVS	Optional, Positional 3
WAIT	Wait time	1-999999	Optional, Positional 4
LCLMQMNAME	Local Message Queue Manager	Character value	Optional, Positional 5
MQMNAME	Message Queue Manager name	*Character value*, ***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.

Examples

None

Error messages

Unknown

Start MQ Service (STRMQMSVC)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Start MQ Service (STRMQMSVC) command starts an MQ service.

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Start MQ Trigger Monitor (STRMQMTRM)

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
INITQNAME	Initiation queue	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Trace MQ (TRCMQM)

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
TRCEARLY	Trace early	***NO**, *YES	Optional, Positional 1
SET	Trace option setting	***ON**, *OFF, *STS, *END	Optional, Positional 2
OUTPUT	Output	***MQM**, *MQMFMT, *PEX, *ALL	Optional, Positional 3
TRCLEVEL	Trace level	***DFT**, *DETAIL, *PARMS	Optional, Positional 4
TRCTYPE	Trace types	Single values: ***ALL** Other values (up to 14 repetitions): *API, *CMTRY, *COMMS, *CSDATA, *CSFLOW, *LQMDATA, *LQMFLOW, *OTHDATA, *OTHFLOW, *RMTDATA, *RMTFLOW, *SVCDATA, *SVCFLOW, *VSNDATA	Optional, Positional 5

Keyword	Description	Choices	Notes
EXCLUDE	Exclude types	Single values: ***NONE** Other values (up to 14 repetitions): *API, *CMTRY, *COMMS, *CSDATA, *CSFLOW, *LQMDATA, *LQMFLOW, *OTHDATA, *OTHFLOW, *RMTDATA, *RMTFLOW, *SVCDATA, *SVCFLOW, *VSNDATA	Optional, Positional 6
INTERVAL	Trace interval	1-32000000, ***NONE**	Optional, Positional 7
MAXSTG	Maximum storage to use	1-16, ***DFT**	Optional, Positional 8
DATASIZE	Trace data size	1-99999999, ***DFT**, *ALL, *NONE	Optional, Positional 9
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 10
JOB	Job information	Values (up to 8 repetitions): *Element list*	Optional, Positional 11
	Element 1: Job name	Qualified job name	
	Qualifier 1: Job name	Generic name, name	
	Qualifier 2: User	*Character value*, **X''**	
	Qualifier 3: Number	*Character value*, **X''**	
	Element 2: Thread identifier	*Character value*, ***NONE**, *INITIAL	
STRCTL	Trace start control	Values (up to 8 repetitions): *Character value*, ***NONE**	Optional, Positional 12
ENDCTL	Trace end control	Values (up to 8 repetitions): *Character value*, ***NONE**	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 identfier 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 identifer 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.

Examples

None

Error messages

Unknown

Work with MQ Queue Manager (WRKMQM)

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

Keyword	Description	Choices	Notes
MQMNAME	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).

Examples

None

Error messages

Unknown

Work with MQ Authority (WRKMQMAUT)

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
OBJ	Object/Profile name	*Character value*, ***ALL**	Optional, Positional 1
OBJTYPE	Object type	*Q, *PRC, *MQM, *NMLIST, *AUTHINFO, *LSR, *SVC, *CHL, *CLTCN, ***ALL**, *TOPIC, *RMTMQMNAME	Optional, Positional 2
OUTPUT	Output	*****, *PRINT	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4
SRVCOMP	Service Component name	*Character value*, ***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.

Examples

None

Error messages

Unknown

Work with MQ Authority Data (WRKMQMAUTD)

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

Keyword	Description	Choices	Notes
OBJ	Object/Profile name	Character value	Required, Positional 1
OBJTYPE	Object type	*Q, *PRC, *MQM, *NMLIST, *AUTHINFO, *CHL, *CLTCN, *SVC, *LSR, *TOPIC	Required, Positional 2
USER	User name	*Name*, *PUBLIC, ***ALL**	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4
SRVCOMP	Service Component name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

Work with AuthInfo objects (WRKMQMAUTI)

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
AINAME	AuthInfo name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *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.

Examples

None

Error messages

Unknown

Work with MQ Channels (WRKMQMCHL)

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

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*, ***ALL**	Optional, Positional 1
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN, ***ALL**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
STATUS	Channel status	***ALL**, *INACTIVE, *STOPPED, *BINDING, *RETRYING, *RUNNING, *SWITCHING	Optional, Positional 4

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 5
	Element 1: Filter keyword	*AFFINITY, *ALTDATE, *ALTTIME, *BATCHHB, *BATCHINT, *BATCHLIM, *BATCHSIZE, *CLNTWGHT, *CLUSNL, *CLUSTER, *CLWLPRTY, *CLWLRANK, *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, *TGTMQMNAME, *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.

***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 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.

***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.

***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 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.

***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 SSL.

The filter value is one of the following:

***REQUIRED**
Client authentication is required.

***OPTIONAL**
Client authentication is optional.

***SSLCIPH**
The CipherSpec using in SSL channel negotiation.

The filter value is the name of the CipherSpec.

***SSLPEER**
The X500 peer name used in SSL 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 switched off.

***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.

Examples

None

Error messages

Unknown

Work with MQ Channel Status (WRKMQMCHST)

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

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*, ***ALL**	Optional, Positional 1
CONNAME	Connection name	*Character value*, ***ALL**	Optional, Positional 2
TMQNAME	Transmission queue name	*Character value*, ***ALL**	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4
CHLSTS	Channel status	***ALL**, *SAVED, *CURRENT	Optional, Positional 5
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 6
	Element 1: Filter keyword	*CHLSTS, *CHLTYPE, *COMPHDR, *COMPMSG, *CONNAME, *INDOUBT, *INDMSGS, *INDSEQNO, *LSTSEQNO, *MONCHL, *RMTMQMNAME, *RMTVERSION, *SHARECNV, *STATUS, *SUBSTATE, *TMQNAME, *XQMSGSA, *LSTMSGDATE, *LSTMSGTIME, *MSGS	
	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 (CONNAME)

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 an SSL 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.

Examples

None

Error messages

Unknown

Work with MQ Clusters (WRKMQMCL)

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

*Table 54. **WRKMQMCL** parameters*

Keyword	Description	Choices	Notes
CLUSQMGR	Cluster Queue Manager name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Table 54. ***WRKMQMCL*** *parameters (continued)*			
Keyword	**Description**	**Choices**	**Notes**
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *BATCHHB, *BATCHINT, *BATCHLIM, *BATCHSIZE, *CHLNAME, *CLUSDATE, *CLUSQMGR, *CLUSTER, *CLUSTIME, *CLWLPRTY, *CLWLRANK, *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.

***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.

***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 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.

***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 SSL.

The filter value is one of the following:

***REQUIRED**
Client authentication is required.

***OPTIONAL**
Client authentication is optional.

***SSLCIPH**
The CipherSpec using in SSL channel negotiation.

The filter value is the name of the CipherSpec.

***SSLPEER**
The X500 peer name used in SSL 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 switched off.

***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.

Work with MQ Cluster Queues (WRKMQMCLQ)

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

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
CLUSTER	Cluster name	*Character value*, ***ALL**	Optional, Positional 3

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 4
	Element 1: Filter keyword	*ALTDATE, *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.

Examples

None

Error messages

Unknown

Work with MQ Connections (WRKMQMCONN)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Work with MQ Connections (WRKMQMCONN) 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
CONN	Connection Identifier	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*APPLDESC, *APPLTAG, *APPLTYPE, *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.

***APPLTYPE**
The type of applcation connected to the queue manager.

The filter value is one of the following:

***CICS**
CICS/400 application.

***MVS**
MVS application.

***IMS**
IMS application.

***OS2**
OS/2 application.

***DOS**
DOS application.

***UNIX**
UNIX application.

***QMGR**
Queue manager application.

***OS400**
IBM i application.

***WINDOWS**
Windows application.

***CICS_VSE**
CICS/VSE application.

***WINDOWS_NT**
Windows NT application.

***VMS**
VMS application.

***NSK**
Tandem/NSK application.

***VOS**
VOS application.

***IMS_BRIDGE**
IMS bridge application.

***XCF**
XCF application.

***CICS_BRIDGE**
CICS bridge application.

***NOTES_AGENT**
Lotus Notes application.

***BROKER**
Broker application.

***JAVA**
Java application.

***DQM**
DQM application.

***CHINIT**
Channel initiator.

***SYSTEM_EXT**
System extension application.

user-value
User-defined application.

The filter value is the integer application type.

***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.

Examples

None

Error messages

Unknown

Work Queue Manager Journals (WRKMQMJRN)

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
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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).

Examples

None

Error messages

Unknown

Work with MQ Listeners (WRKMQMLSR)

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
OPTION	Option	***STATUS**, *OBJECT	Optional, Positional 1
LSRNAME	Listener name	*Character value*, ***ALL**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 4
	Element 1: Filter keyword	*ALTDATE, *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.

***ALTTIME**
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.

Examples

None

Error messages

Unknown

Work with MQ Messages (WRKMQMMSG)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Work with MQ Messages (WRKMQMMSG) 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
QNAME	Queue name	Character value	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
FIRST	First Message	1-30000, **1**	Optional, Positional 3
MAXMSG	Maximum number of messages	1-30000, **48**	Optional, Positional 4
MAXMSGLEN	Maximum message size	128-999999, **1024**	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.

Examples

None

Error messages

Unknown

Work with MQ Namelist (WRKMQMNL)

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

Keyword	Description	Choices	Notes
NAMELIST	Namelist	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *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:

***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.

***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.

Examples

None

Error messages

Unknown

Work with MQ Processes (WRKMQMPRC)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Work with MQ Processes (WRKMQMPRC) 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

Keyword	Description	Choices	Notes
PRCNAME	Process name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *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.

***ALTTIME**
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 the application to start.

The filter value is one of the following:

***CICS**
CICS/400 application.

***MVS**
MVS application.

***IMS**
IMS application.

***OS2**
OS/2 application.

***DOS**
DOS application.

***UNIX**
UNIX application.

***QMGR**
Queue manager application.

***OS400**
IBM i application.

***WINDOWS**
Windows application.

***CICS_VSE**
CICS/VSE application.

***WINDOWS_NT**
Windows NT application.

***VMS**
VMS application.

***NSK**
Tandem/NSK application.

***VOS**
VOS application.

***IMS_BRIDGE**
IMS bridge application.

***XCF**
XCF application.

***CICS_BRIDGE**
CICS bridge application.

***NOTES_AGENT**
Lotus Notes application.

***BROKER**
Broker application.

***JAVA**
Java application.

***DQM**
DQM application.

user-value
User-defined application.

The filter value is the integer application type.

***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.

Examples

None

Error messages

Unknown

Work with MQ Queues (WRKMQMQ)

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
QNAME	Queue name	*Character value*, ***ALL**	Optional, Positional 1
QTYPE	Queue type	***ALL**, *ALS, *LCL, *MDL, *RMT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
CLUSTER	Cluster name	*Character value*, ***ALL**	Optional, Positional 4
CLUSNL	Cluster namelist name	*Character value*, ***ALL**	Optional, Positional 5

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	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 switched off.

***ON**
Accounting data collection is switched on for this queue.

***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.

***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.

***CLWLRANK**
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 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.

***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 codepage 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 switched off.

***ON**
Statistics data collection is switched on 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.

Examples

None

Error messages

Unknown

Work with Queue Status (WRKMQMQSTS)

Where allowed to run
All environments (*ALL)

Threadsafe
Yes

The Work with Queue Status (WRKMQMQSTS) 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
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1

Keyword	Description	Choices	Notes
QNAME	Queue name	Character value	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*APPLDESC, *APPLTAG, *BROWSE, *CHLNAME, *CONNAME, *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.

***CONNAME**
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.

Examples

None

Error messages

Unknown

Work with MQM Security Policies (WRKMQMSPL)

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 IBM MQ 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 IBM MQ Advanced Message Security license must be installed when the queue manager was started for this to occur.

Parameters

Keyword	Description	Choices	Notes
OUTPUT	Output	*****, *PRINT	Optional, Positional 1
MQMNAME	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**
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.

Work with MQ Subscriptions (WRKMQMSUB)

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

Keyword	Description	Choices	Notes
SUBNAME	Subscription name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*DEST, *DESTCLASS, *DESTCRRLID, *DESTMQM, *EXPIRY, *PSPROP, *PUBACCT, *PUBAPPID, *PUBPTY, *REQONLY, *SELECTOR, *SELTYPE, *SUBSCOPE, *SUBID, *TOPICOBJ, *TOPICSTR, *USERDATA, *VARUSER, *WSCHEMA	
	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.

***DESTCRRLID**
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 Version 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.

***WSCHEMA**
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.

Examples

None

Error messages

Unknown

Work with MQ Service object (WRKMQMSVC)

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

Keyword	Description	Choices	Notes
SVCNAME	Service name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *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.

Examples

None

Error messages

Unknown

Work with MQ Topics (WRKMQMTOP)

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

Keyword	Description	Choices	Notes
TOPNAME	Topic name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *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 topices 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.

***ALTTIME**
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.

***DFTPUTRESP**
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.

Examples

None

Error messages

Unknown

Work with MQ Transactions (WRKMQMTRN)

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

Keyword	Description	Choices	Notes
TYPE	Transaction type	***ALL**, *EXT, *INT, *MQI, *XA, *OS400	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	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.

Examples

None

Error messages

Unknown

MQSC reference

Use MQSC commands to manage queue manager objects, including the queue manager itself, queues, process definitions, channels, client connection channels, listeners, services, namelists, clusters, and authentication information objects.

For an overview of using MQSC commands for administering IBM MQ, see Performing local administration tasks using MQSC commands.

MQSC commands use certain special characters to have certain meanings. For more information about these special characters and how to use them, see “Generic values and characters with special meanings” on page 627.

To find out how you can build scripts using MQSC commands, see “Building command scripts” on page 628.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

For the full list of MQSC commands, see “The MQSC commands” on page 630.

Related concepts
“IBM MQ Control commands” on page 8
Find out how to use the IBM MQ control commands.

“Programmable command formats reference” on page 1327
Programmable Command Formats (PCFs) define command and reply messages that can be exchanged between a program and any queue manager (that supports PCFs) in a network. PCFs simplify queue manager administration and other network administration.

Related reference
“IBM MQ for IBM i CL commands” on page 187
A list of IBM MQ for IBM i CL commands grouped according to command type:

Generic values and characters with special meanings

The following information describes generic values, and characters that have special meaning when you build MQSC commands.

Wherever a parameter can have a generic value, it is entered ending with an asterisk (*), for example ABC*. A generic value means all values beginning with; so ABC* means all values beginning with ABC.

If characters that require quotation marks are used in the value, the asterisk must be placed inside the quotation marks, thus `'abc*'`. The asterisk must be the last or only character in the value.

The question mark (?) and colon (:) are not allowed in generic values.

Character	Description
	Blanks are used as separators. Multiple blanks are equivalent to a single blank, except in strings that are enclosed in apostrophes ('). Any trailing blanks in those string attributes which are based on MQCHARV types are treated as significant.
,	Commas are used as separators. Multiple commas are equivalent to a single comma, except in strings that are enclosed in apostrophes (').
'	An apostrophe indicates the beginning or end of a string. IBM MQ leaves all characters that are enclosed in quotation marks exactly as they are entered. The containing apostrophes are not included when calculating the length of the string.
''	Single quotation marks inside a string are treated by IBM MQ as one character when calculating the length of the string and the string is not terminated.
=	On z/OS, an equals sign indicates the start of a parameter value which is ended by a comma or blank.
(	An open parenthesis indicates the beginning of a parameter value or list of values.
)	A close parenthesis indicates the end of a parameter value or list of values.
:	A colon indicates an inclusive range. For example (1:5) means (1,2,3,4,5). This notation can be used only in **TRACE** commands.
*	An asterisk means all. For example, **DISPLAY TRACE (*)** means display all traces, and **DISPLAY QUEUE (PAY*)** means display all queues with names that begin with PAY.

When you need to use any of these special characters in a field (for example as part of a description), you must enclose the whole string in single quotation marks.

Building command scripts

Use this information to learn how to build command scripts.

You might want to build the MQSC commands into a script when you use:

- The CSQINP1, CSQINP2, and CSQINPX initialization data sets or the CSQUTIL batch utility on z/OS.
- The STRMQM command on IBM i.
- The `runmqsc` command on UNIX, Linux, and Windows systems.

When you do this, follow these rules:

- Each command must start on a new line.
- On each platform, there might be platform-specific rules about the line length and record format. If scripts are to be readily portable to different platforms, the significant length of each line should be restricted to 72 characters.
  - On z/OS, scripts are held in a fixed-format data set, with a record length of 80. Only columns 1 through 72 can contain meaningful information; columns 73 through 80 are ignored.
  - On AIX, HP-UX, Linux, IBM i, Solaris, and Windows, each line can be of any length up to a maximum of 2048 characters.
  - On other UNIX systems, each line can be of any length up to and including 80 characters.
- A line must not end in a keyboard control character (for example, a tab).
- If the last nonblank character on a line is:
  - A minus sign (-), this indicates that the command is to be continued from the start of the next line.

– A plus sign (+), this indicates that the command is to be continued from the first nonblank character in the next line. If you use + to continue a command remember to leave at least one blank before the next parameter (except on z/OS where this is not necessary).

Either of these can occur within a parameter, data value, or a string enclosed in quotation marks. For example,

```
   'Fr+
    ed'
```

and

```
   'Fr-
ed'
```

(where the 'e' of the second line of the second example is in the first position of the line) are both equivalent to

```
   'Fred'
```

MQSC commands that are contained within an Escape PCF (Programmable Command Format) command cannot be continued in this way. The entire command must be contained within a single Escape command. (For information about the PCF commands, see Introduction to Programmable Command Formats).

- + and - values used at the ends of lines are discarded when the command is reassembled into a single string.
- On AIX, HP-UX, Linux, IBM i, Solaris Solaris, and Windows you can use a semicolon character (;) to terminate a command, even if you have entered a plus sign (+) at the end of the previous line. You can also use the semicolon in the same way on z/OS for commands issued from the CSQUTIL batch utility program.
- A line starting with an asterisk (*) in the first position is ignored. This can be used to insert comments into the file.

  A blank line is also ignored.

  If a line ends with a continuation character (- or +), the command continues with the next line that is not a comment line or a blank line.
- When running MQSC commands interactively, you end the interactive session by typing the END command. This applies to:
  – UNIX, Linux, and Windows systems, where you start the interactive session by entering `runmqsc`
  – IBM i systems, where you start the interactive session from the WRKMQM panel
- On Windows, if certain special characters such as the pound sign ('£') and the logical NOT ('¬') are used in a command script (for example, as part of an object description), they are displayed differently in the output from a command such as DISPLAY QLOCAL.

z/OS Using commands on z/OS

MQSC commands can be issued from various sources, depending on the command.

Commands can be issued from the following sources:

- The z/OS console or equivalent
- The initialization input data sets CSQINP1, CSQINP2, CSQINPT and CSQINPX
- The CSQUTIL batch utility
- Suitably authorized applications, sending commands as messages to the SYSTEM.COMMAND.INPUT queue

z/OS For further details, see Issuing commands,

However, not all commands can be issued from all these sources. Commands can be classified according to whether they can be issued from:

1
CSQINP1

2
CSQINP2

C
The z/OS console

R
The command server and command queue, by means of , for example, CSQUTIL, CSQINPT, CSQINPX, or applications

Within the command descriptions that follow, these sources are identified by the use of the characters 1, 2, C, and R in the z/OS column of the tables for each command description.

The MQSC commands

Use this topic as a reference to the MQSC commands.

This section describes, in alphabetical order, all the MQSC commands that can be issued by operators and administrators.

"ALTER AUTHINFO" on page 633
"ALTER BUFFPOOL" on page 644
"ALTER CFSTRUCT" on page 646
"ALTER CHANNEL" on page 653
"ALTER CHANNEL (MQTT)" on page 705
"ALTER COMMINFO" on page 708
"ALTER LISTENER" on page 712
"ALTER NAMELIST" on page 715
"ALTER PROCESS" on page 718
"ALTER PSID" on page 722
"ALTER QMGR" on page 724
"ALTER queues" on page 756
z/OS "ALTER SECURITY" on page 785
"ALTER SERVICE" on page 787
"ALTER SMDS" on page 789
"ALTER STGCLASS" on page 790
"ALTER SUB" on page 793
"ALTER TOPIC" on page 797
"ALTER TRACE" on page 806
"ARCHIVE LOG" on page 807
"BACKUP CFSTRUCT" on page 810
"CLEAR QLOCAL" on page 811
"CLEAR TOPICSTR" on page 813
"DEFINE AUTHINFO" on page 815
"DEFINE BUFFPOOL" on page 826
"DEFINE CFSTRUCT" on page 829
"DEFINE CHANNEL" on page 836
"DEFINE CHANNEL (MQTT)" on page 890
"DEFINE COMMINFO" on page 893
"DEFINE LISTENER" on page 897
"DEFINE LOG" on page 900

“DEFINE MAXSMSGS” on page 902
“DEFINE NAMELIST” on page 903
“DEFINE PROCESS” on page 906
“DEFINE PSID” on page 911
“DEFINE queues” on page 913
“DEFINE SERVICE” on page 945
“DEFINE STGCLASS” on page 948
“DEFINE SUB” on page 952
“DEFINE TOPIC” on page 958
“DELETE AUTHINFO” on page 968
“DELETE BUFFPOOL” on page 971
“DELETE CFSTRUCT” on page 972
“DELETE CHANNEL” on page 972
“DELETE CHANNEL (MQTT)” on page 975
“DELETE COMMINFO” on page 975
“DELETE LISTENER” on page 976
“DELETE NAMELIST” on page 976
“DELETE PROCESS” on page 979
“DELETE PSID” on page 980
“DELETE queues” on page 981
“DELETE SERVICE” on page 986
“DELETE STGCLASS” on page 988
“DELETE SUB” on page 986
“DELETE TOPIC” on page 989
“DISPLAY ARCHIVE” on page 991
“DISPLAY AUTHINFO” on page 993
“DISPLAY CFSTATUS” on page 1004
“DISPLAY CFSTRUCT” on page 1011
“DISPLAY CHANNEL” on page 1015
“DISPLAY CHANNEL (MQTT)” on page 1030
“DISPLAY CHINIT” on page 1034
“DISPLAY CHLAUTH” on page 1035
“DISPLAY CHSTATUS” on page 1042
“DISPLAY CHSTATUS (MQTT)” on page 1066
“DISPLAY CLUSQMGR” on page 1070
“DISPLAY CMDSERV” on page 1079
“DISPLAY COMMINFO” on page 1079
“DISPLAY CONN” on page 1082
“DISPLAY GROUP” on page 1098
“DISPLAY LISTENER” on page 1099
“DISPLAY LOG” on page 1102
“DISPLAY LSSTATUS” on page 1104
“DISPLAY MAXSMSGS” on page 1107
“DISPLAY NAMELIST” on page 1108
“DISPLAY PROCESS” on page 1114
“DISPLAY PUBSUB” on page 1118
“DISPLAY QMGR” on page 1122
“DISPLAY QMSTATUS” on page 1134
“DISPLAY QSTATUS” on page 1137
“DISPLAY QUEUE” on page 1149
“DISPLAY SBSTATUS” on page 1164

z/OS "DISPLAY SECURITY" on page 1169
"DISPLAY SERVICE" on page 1170
"DISPLAY SMDS" on page 1173
"DISPLAY SMDSCONN" on page 1175
"DISPLAY STGCLASS" on page 1179
"DISPLAY SUB" on page 1183
"DISPLAY SVSTATUS" on page 1190
"DISPLAY SYSTEM (display system information) on z/OS" on page 1193
"DISPLAY THREAD" on page 1199
"DISPLAY TOPIC" on page 1201
"DISPLAY TPSTATUS" on page 1210
"DISPLAY TRACE" on page 1217
"DISPLAY USAGE" on page 1220
"MOVE QLOCAL" on page 1222
"PING CHANNEL" on page 1224
"PING QMGR" on page 1227
"RECOVER CFSTRUCT" on page 1230
"REFRESH CLUSTER" on page 1232
"REFRESH QMGR" on page 1235
"REFRESH SECURITY" on page 1238
"RESET CFSTRUCT" on page 1243
"RESET CHANNEL" on page 1243
"RESET CLUSTER" on page 1246
"RESET QMGR" on page 1248
"RESET QSTATS" on page 1250
"RESET SMDS" on page 1252
"RESET TPIPE" on page 1254
"RESOLVE CHANNEL" on page 1256
"RESOLVE INDOUBT" on page 1258
"RESUME QMGR" on page 1260
"RVERIFY SECURITY" on page 1262
"SET ARCHIVE" on page 1263
"SET CHLAUTH" on page 1273
"SET LOG" on page 1282
"SET SYSTEM" on page 1287
"START CHANNEL" on page 1290
"START CHANNEL (MQTT)" on page 1293
"START CHINIT" on page 1294
"START CMDSERV" on page 1295
"START LISTENER" on page 1296
"START QMGR" on page 1299
"START SERVICE" on page 1300
"START SMDSCONN" on page 1300
"START TRACE" on page 1301
"STOP CHANNEL" on page 1307
"STOP CHANNEL (MQTT)" on page 1311
"STOP CHINIT" on page 1312
"STOP CMDSERV" on page 1313
"STOP CONN" on page 1314
"STOP LISTENER" on page 1315
"STOP QMGR" on page 1317

"STOP SERVICE" on page 1319
"STOP SMDSCONN" on page 1320
"STOP TRACE" on page 1321
"SUSPEND QMGR" on page 1324

Related information

Clustering: Using REFRESH CLUSTER best practices

ALTER AUTHINFO

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.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

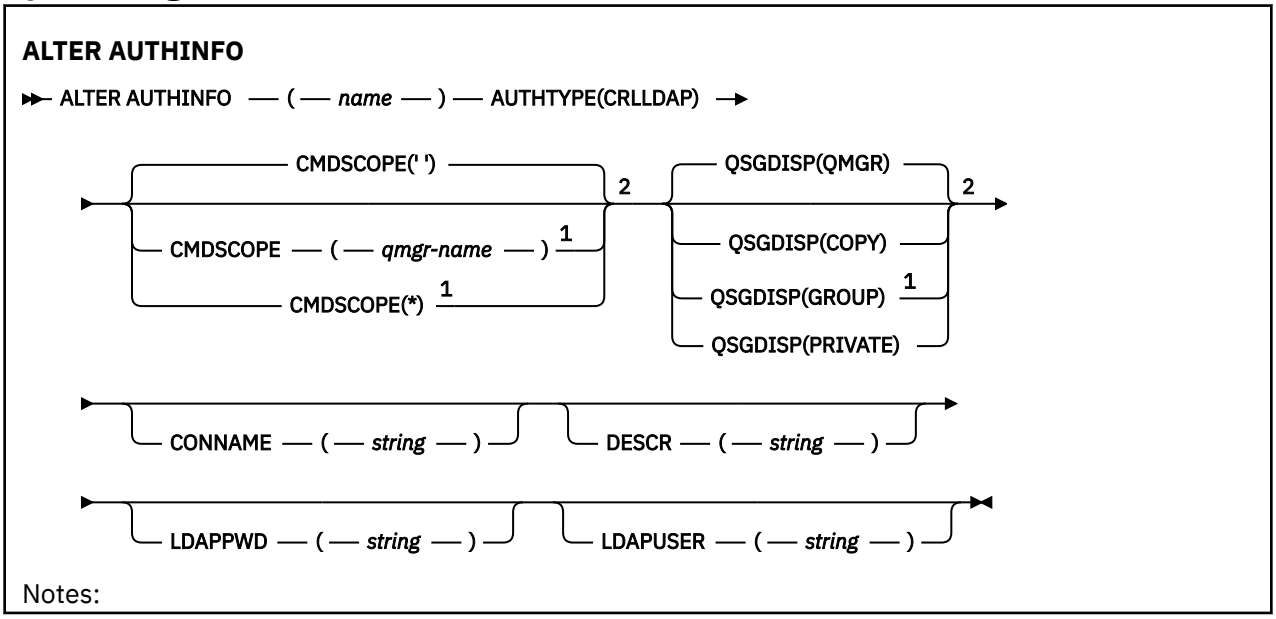
Parameters not specified in the ALTER AUTHINFO command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- 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 636

Synonym: ALT AUTHINFO

Syntax diagram for TYPE(CRLLDAP)

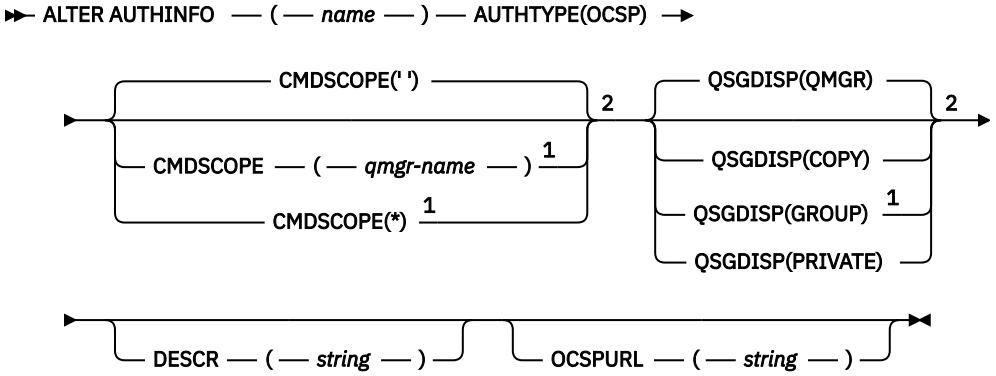


Notes:

[1] 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.
[2] Valid only on z/OS.

Syntax diagram for TYPE(OCSP)

ALTER AUTHINFO

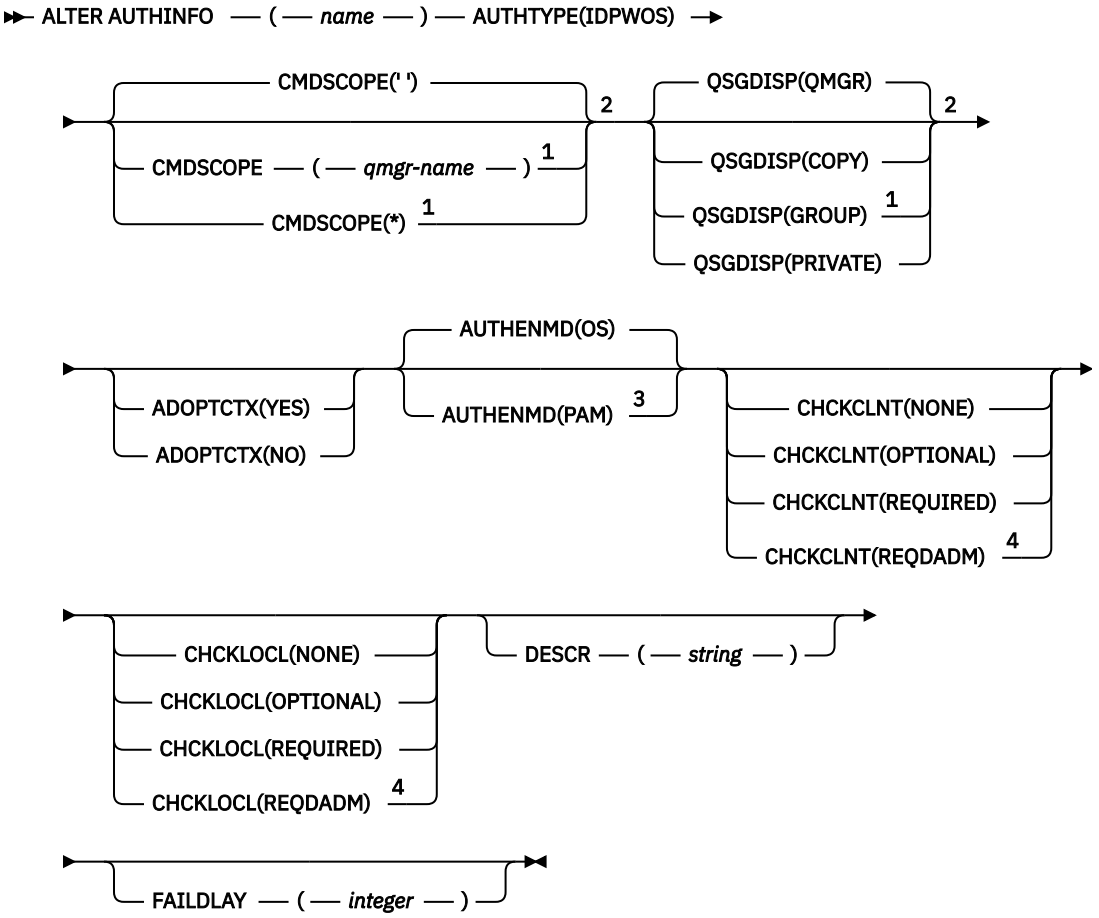


Notes:

[1] 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.
[2] Valid only on z/OS.

Syntax diagram for TYPE(IDPWOS)

ALTER AUTHINFO



Notes:

[1] 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.

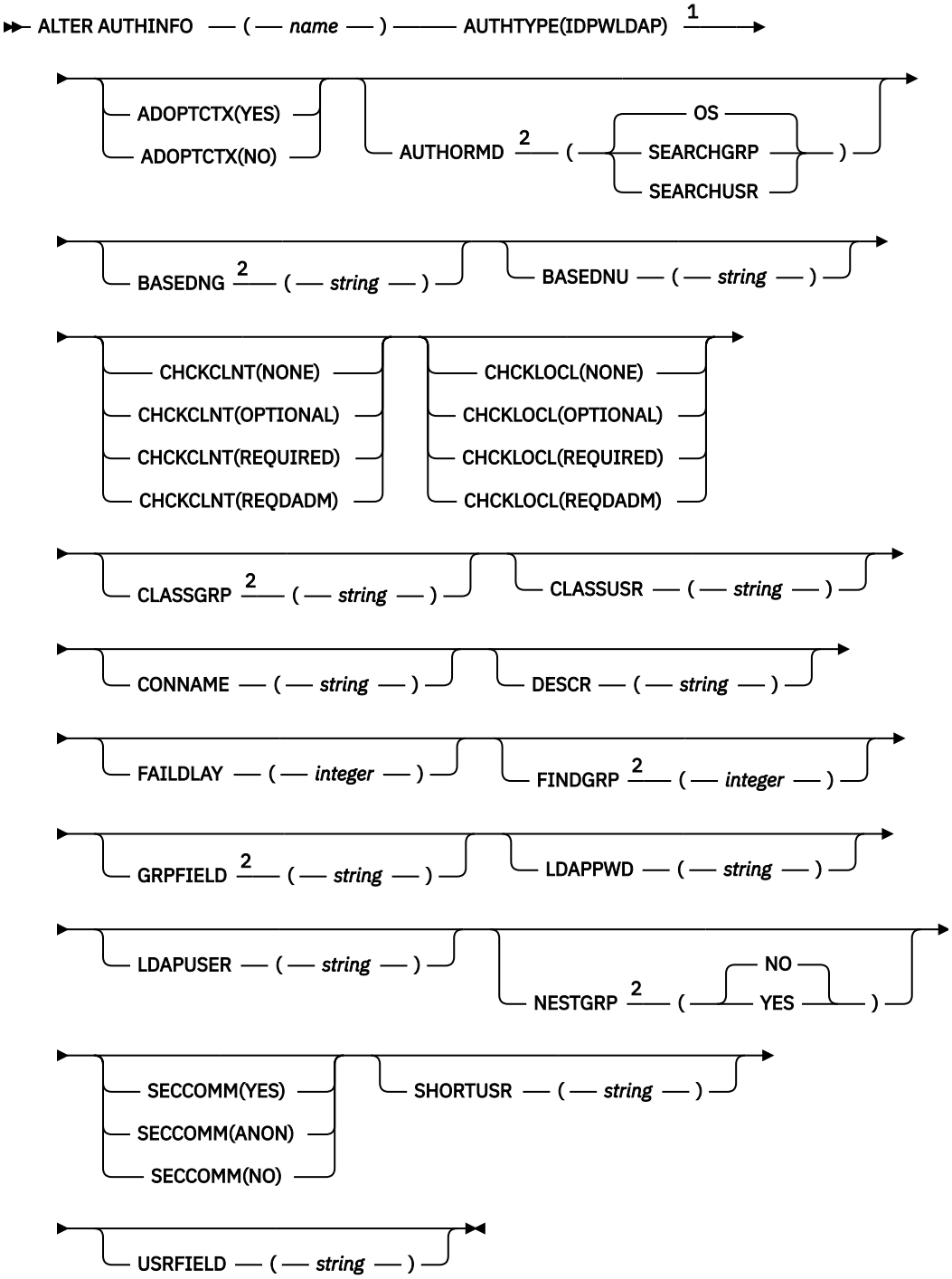
[2] Valid only on z/OS.

[3] Not valid on z/OS and PAM value can be set only on UNIX platforms.

[4] Not valid on z/OS.

Syntax diagram for TYPE(IDPWLDAP)

ALTER AUTHINFO



Notes:

[1] Not valid on z/OS.
[2] Not valid on Windows.

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 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*.

V 8.0.0.3 **AUTHENMD**
Authentication method. Whether to use the operating system or Pluggable Authentication Method (PAM) to authenticate user passwords.

OS
Use the traditional UNIX platforms password verification method.

PAM
Use the PAM to authenticate the user password.

You can set the PAM value only on UNIX and Linux platforms.

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*.

V 8.0.0.2 **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 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 all the groups to which the specified user belongs. The attribute to query is defined by the FINDGRP value, typically *memberOf*.

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* V 8.0.0.2, and is not applicable to Windows platforms.

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.

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.

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.

V 8.0.0.2 **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.

This attribute is valid only for an **AUTHTYPE** of *IDPWLDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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
No user ID and password checks are made. If any user ID or password is supplied by a client application, the credentials are ignored.

OPTIONAL
Client applications are not required to provide a user ID and password.

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 will only be allowed to continue if the user ID and password are valid.

This option might be useful during migration, for example.

REQUIRED
All client applications must provide a user ID and password in the MQCSP structure. This user ID and password is 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.

REQDADM
All client applications using a privileged user ID must provide a user ID and password in the MQCSP structure. Any locally bound applications using a non-privileged user ID are not required to provide a user ID and password and are treated as with the OPTIONAL setting.

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.

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.

A privileged user is one that has full administrative authorities for IBM MQ. See Privileged users for more information.

z/OS This setting is not valid 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 message AMQ9793 on platforms other than z/OS, and message 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* . The possible values are:

NONE
No user ID and password checks are made. If any user ID or password is supplied by a locally bound application, the credentials are ignored.

OPTIONAL
Locally bound applications are not required to provide a user ID and password.

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 will only be allowed to continue if the user ID and password are valid.

This option might be useful during migration, for example.

REQUIRED
All locally bound applications must provide a user ID and password in the MQCSP structure. This 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.

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 a user ID and password in the MQCSP structure. Any locally bound applications using a non-privileged user ID are not required to provide a user ID and password and are treated as with the OPTIONAL setting.

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.

A privileged user is one that has full administrative authorities for IBM MQ. See Privileged users for more information.

z/OS This setting is not allowed on z/OS systems.

V 8.0.0.2 **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 an **AUTHTYPE** of *IDPWLDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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 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.

CONNAME(*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.

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 is 264 characters on IBM i, UNIX systems, and Windows, and 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" on page 993).

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 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 valid only for an **AUTHTYPE** of *IDPWOS* and *IDPWLDAP* .

V 8.0.0.2 **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 left blank, if:

- AUTHORMD =*SEARCHGRP*, the **FINDGRP** attribute defaults to *memberOf*
- AUTHORMD =*SEARCHUSR*, the **FINDGRP** attribute defaults to *member*

V 8.0.0.2 **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.

This attribute is valid only for an **AUTHTYPE** of *IDPWLDAP* V 8.0.0.2 , and is not applicable to Windows platforms.

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.

This 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 IBM i, UNIX systems, and Windows, and 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.

On IBM i, UNIX systems, and Windows, the maximum accepted line length is defined to be BUFSIZ, which can be found in stdio.h.

This attribute is valid only for an **AUTHTYPE** of *CRLLDAP* and *IDPWLDAP* .

V 8.0.0.2 **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.

This attribute is valid only for an **AUTHTYPE** of *IDPWLDAP* V 8.0.0.2 , and is not applicable to Windows platforms.

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).

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 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.

QSGDISP	ALTER

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 attempt to refresh local copies on page set zero:
	`DEFINE AUTHINFO(name)`
	`REPLACE QSGDISP(COPY)`
	The ALTER for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with QSGDISP(QMGR) or QSGDISP(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 QSGDISP(QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

SECCOMM
Whether connectivity to the LDAP server should be done securely using SSL/TLS

YES
Connectivity to the LDAP server is made securely using SSL/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 SSL/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 SSL/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.

This 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 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 ALTER BUFFPOOL

Use the MQSC command ALTER BUFFPOOL to dynamically change the settings of a predefined buffer pool.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

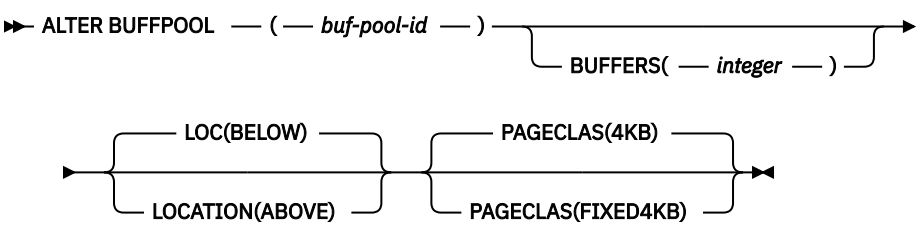
Parameters not specified in the ALTER BUFFPOOL command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for ALTER BUFFPOOL” on page 645
- “Parameter descriptions for ALTER BUFFPOOL” on page 645

Synonym: ALT BP

ALTER BUFFPOOL



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 is processed 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 15. If **OPMODE** is set to `OPMODE=(NEWFUNC, 800)`, 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 or LOC(BELOW or ABOVE)

The **LOCATION** or **LOC** parameter specifies where the memory used by the specified buffer pool is located.

LOCATION and **LOC** are synonyms and either, but not both, can be used.

This memory location can be either `ABOVE` (64 bit) or `BELOW` (31 bit) the bar, with `BELOW` being the default.

`ABOVE` can only be specified if `OPMODE(NEWFUNC, 800)` is in effect. `BELOW` can be specified regardless of `OPMODE(NEWFUNC, 800)` being used and has the same effect as not specifying the **LOCATION** parameter.

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 utilized.

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.

FIXED4KB can only be specified if OPMODE(NEWFUNC, 800) is in effect whereas 4KB can be specified regardless of the setting of OPMODE(NEWFUNC, 800).

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 up, 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.

z/OS ALTER CFSTRUCT

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

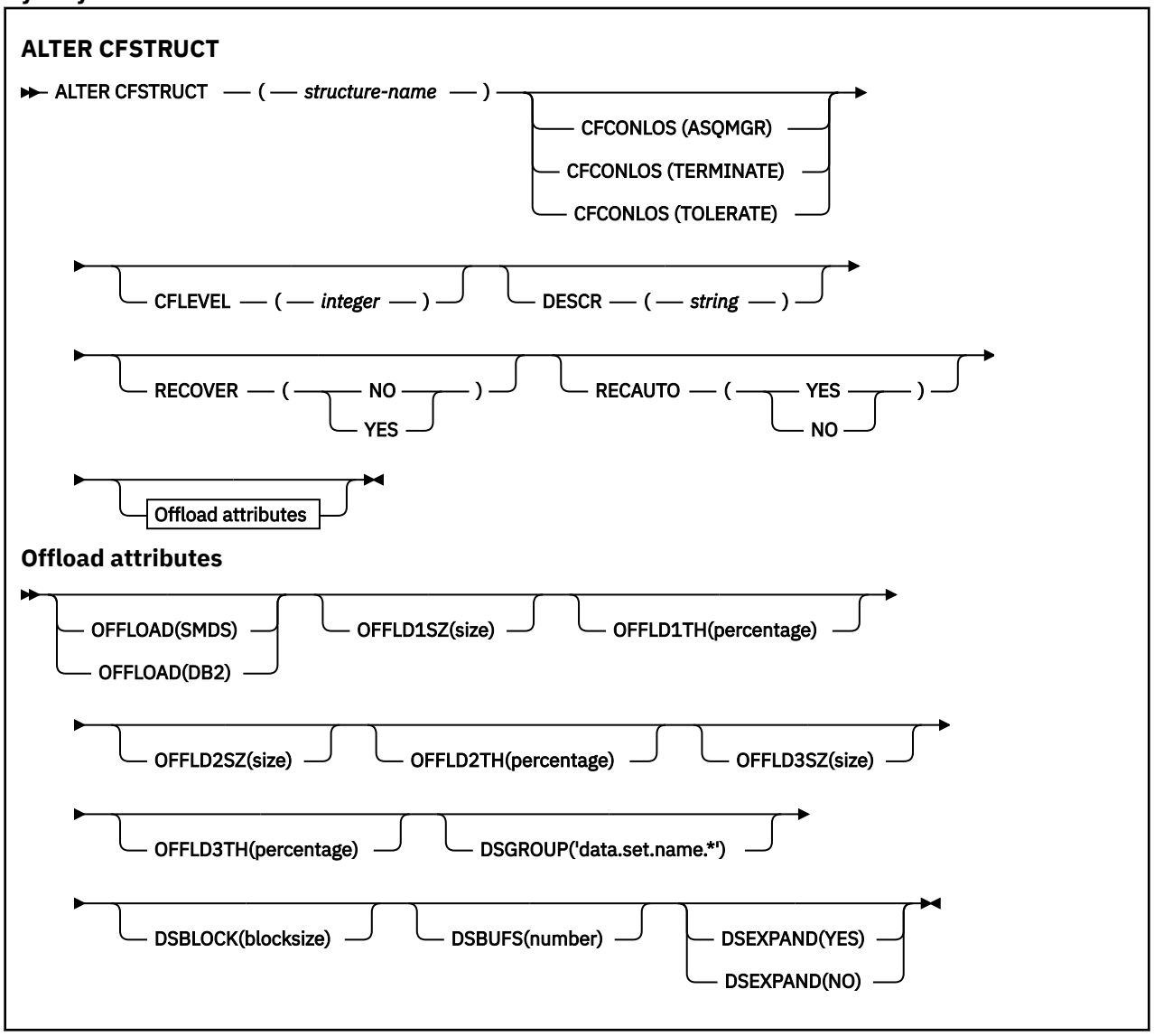
Parameters not specified in the ALTER CFSTRUCT command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram

- "Usage notes" on page 647
- "Parameter descriptions for ALTER CFSTRUCT" on page 647

Synonym: ALT CFSTRUCT



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.

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 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 only increase the value of **CFLEVEL** to 5 if all the queue managers in the queue-sharing group are at command level 710 or greater and have IBM WebSphere MQ 7.1.0 new functions enabled with **OPMODE**.

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.

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)**.

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.

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 or N
CF application structure recovery is not supported.

YES or Y
CF application structure recovery is supported.

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 or Y
The structure and associated shared message data sets which also need recovery are automatically recovered.

NO or N
The structure is not automatically recovered. 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

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

Parameters not specified in the ALTER CHANNEL command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

Synonym: ALT CHL

- "Usage notes" on page 653
- "Parameter descriptions for ALTER CHANNEL" on page 653

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 CONNAME, you must reset the sequence number at both ends of the channel. (See "RESET CHANNEL" on page 1243 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.

Table 55. DEFINE and ALTER CHANNEL parameters

Parameter	SDR	SVR	RCVR	RQSTR	CLNTCONN	SVRCONN	CLUSSDR	CLUSRCVR	V 8.0.0.4 AMQP
AFFINITY					✓				

Table 55. DEFINE and ALTER CHANNEL parameters (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNTCONN	SVRCONN	CLUSSDR	CLUSRCVR	V 8.0.0.4 AMQP
V 8.0.0.4 V 8.0.0.4 AMQPKA									✓
BATCHHB	✓	✓					✓	✓	
BATCHINT	✓	✓					✓	✓	
BATCHLIM	✓	✓					✓	✓	
BATCHSZ	✓	✓	✓	✓			✓	✓	
CERTLABL	✓	✓	✓	✓	✓	✓		✓	✓
channel-name	✓	✓	✓	✓	✓	✓	✓	✓	✓
CHLTYPE	✓	✓	✓	✓	✓	✓	✓	✓	✓
CLNTWGHT					✓				
CLUSNL							✓	✓	
CLUSTER							✓	✓	
CLWLPRTY							✓	✓	
CLWLRANK							✓	✓	
CLWLWGHT							✓	✓	
z/OS z/OS CMDSCOPE	✓	✓	✓	✓	✓	✓	✓	✓	
COMPHDR	✓	✓	✓	✓	✓	✓	✓	✓	
COMPMSG	✓	✓	✓	✓	✓	✓	✓	✓	
CONNAME	✓	✓		✓	✓		✓	✓	
CONVERT	✓	✓					✓	✓	
DEFCDISP	✓	✓	✓	✓		✓			
DEFRECON					✓				
DESCR	✓	✓	✓	✓	✓	✓	✓	✓	✓
DISCINT	✓	✓				✓	✓	✓	
HBINT	✓	✓	✓	✓	✓	✓	✓	✓	
KAINT	✓	✓	✓	✓	✓	✓	✓	✓	

Table 55. DEFINE and ALTER CHANNEL parameters (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNTCONN	SVRCONN	CLUSSDR	CLUSRCVR	V 8.0.0.4 AMQP
LIKE	✓	✓	✓	✓	✓	✓	✓	✓	
LOCLADDR	✓	✓		✓	✓		✓	✓	✓
LONGRTY	✓	✓					✓	✓	
LONGTMR	✓	✓					✓	✓	
MAXINST						✓			✓
MAXINSTC						✓			
MAXMSGL	✓	✓	✓	✓	✓	✓	✓	✓	✓
MCANAME	✓	✓		✓			✓	✓	
MCATYPE	✓	✓		✓			✓	✓	
MCAUSER			✓	✓		✓		✓	✓
MODENAME	✓	✓		✓	✓		✓	✓	
MONCHL	✓	✓	✓	✓		✓	✓	✓	
MRDATA			✓	✓				✓	
MREXIT			✓	✓				✓	
MRRTY			✓	✓				✓	
MRTMR			✓	✓				✓	
MSGDATA	✓	✓	✓	✓			✓	✓	
MSGEXIT	✓	✓	✓	✓			✓	✓	
NETPRTY								✓	
NPMSPEED	✓	✓	✓	✓			✓	✓	
PASSWORD	✓	✓		✓	✓		✓	✓	
V 8.0.0.4 V 8.0.0.4 PORT									✓
PROPCTL	✓	✓					✓	✓	
PUTAUT			✓	✓		✓		✓	
QMNAME					✓				

Table 55. DEFINE and ALTER CHANNEL parameters (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNTCONN	SVRCONN	CLUSSDR	CLUSRCVR	V 8.0.0.4 AMQP
z/OS z/OS QSGDISP	✓	✓	✓	✓	✓	✓	✓	✓	
RCVDATA	✓	✓	✓	✓	✓	✓	✓	✓	
RCVEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
SCYDATA	✓	✓	✓	✓	✓	✓	✓	✓	
SCYEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
SENDDATA	✓	✓	✓	✓	✓	✓	✓	✓	
SENDEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
SEQWRAP	✓	✓	✓	✓			✓	✓	
SHARECNV					✓	✓			
SHORTRTY	✓	✓					✓	✓	
SHORTTMR	✓	✓					✓	✓	
SSLCAUTH		✓	✓	✓		✓		✓	
SSLCIPH [1]	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSLPEER	✓	✓	✓	✓	✓	✓	✓	✓	✓
STATCHL	✓	✓	✓	✓			✓	✓	
TPNAME	✓	✓		✓	✓	✓	✓	✓	
V 8.0.0.4 V 8.0.0.4 TPROOT									✓
TRPTYPE	✓	✓	✓	✓	✓	✓	✓	✓	
V 8.0.0.4 V 8.0.0.4 USECLTID									✓
USEDLQ	✓	✓	✓	✓			✓	✓	
USERID	✓	✓		✓	✓		✓		
XMITQ	✓	✓							

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 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.

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.

distributed V 8.0.0.4 **AMQPKA(*integer*)**
The keep alive time for an AMQP channel in seconds. 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.

This 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 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.

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.
- z/OS 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 distributed platforms, the maximum number of uncommitted messages allowed at the sending queue manager (or one if this value is zero or less).
- z/OS 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 distributed platforms, 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** parameter.

Note that inbound channels (including receiver, 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.

In all other cases, the queue manager **CERTLABL** parameter determines the certificate sent. In particular, the following only ever receive the certificate configured by the **CERTLABL** parameter of the queue manager, regardless of the channel-specific label setting:

- All current Java and JMS clients.
- Versions of IBM MQ prior to Version 8.0.

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. 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. This facility applies to AIX, HP-UX, Linux, IBM i, Solaris, and Windows only; for further 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 z/OS 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 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 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 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 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(*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 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.

z/OS **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.

z/OS 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.

CONNAME(***string*** **)**
Connection name.

For cluster-receiver channels (when specified) CONNAME 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.

On distributed platforms, 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.

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.

Providing multiple connection names in a list was first supported in IBM WebSphere MQ 7.0.1. It changes the syntax of the CONNAME parameter. Earlier clients and queue managers connect using the first connection name in the list, and do not read the rest of the connection names in the list. In order for the earlier clients and queue managers to parse the new syntax, you must specify a port number on the first connection name in the list. Specifying a port number avoids problems when connecting to the channel from a client or queue manager that is running at a level earlier than IBM WebSphere MQ 7.0.1.

On AIX, HP-UX, IBM i, Linux, Solaris, and Windows, 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.

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

- 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.

- On IBM i, Windows, UNIX and Linux systems, CONNAME is the name of the CPI-C communications side object or, if the TPNAME is not blank, CONNAME is the fully qualified name of the partner logical unit.

  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.

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 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, 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.

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 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

Unless overridden by MQCONNX, the client is not reconnected automatically.

YES

Unless overridden by MQCONNX, the client reconnects automatically.

QMGR
Unless overridden by MQCONNX, 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 MQCONNX MQI call.

Table 56. 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	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 server-connection and client-connection channels working in the channel mode before IBM WebSphere MQ 7.0, heartbeats flow only when a server MCA is waiting for an MQGET command with the WAIT option specified, which it has issued on behalf of a client application.

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, 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.

On distributed 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 MQ Explorer.

Keepalive must also be switched on within TCP/IP itself. Refer to your TCP/IP documentation for information about configuring keepalive. On AIX, use the **no** command. On HP-UX, use the **ndd** command. 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.

On distributed platforms, 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 Solaris, for example, flows to (and is implemented by) z/OS queue managers that are in, or join, the cluster.

On distributed platforms, 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.

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 the 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:

Table 57. Examples of how the LOCLADDR parameter can be used

LOCLADDR	Meaning
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 58. How the IP stack to be used for communication is determined

Protocols supported	CONNAME	LOCLADDR	Action of channel
IPv4 only	IPv4 address [1]		Channel binds to IPv4 stack
	IPv6 address [2]		Channel fails to resolve CONNAME
	IPv4 and 6 host name [3]		Channel binds to IPv4 stack
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve CONNAME
	IPv4 and 6 host name	IPv4 address	Channel binds to IPv4 stack
	Any address [4]	IPv6 address	Channel fails to resolve LOCLADDR
	IPv4 address	IPv4 and 6 host name	Channel binds to IPv4 stack
	IPv6 address	IPv4 and 6 host name	Channel fails to resolve CONNAME
	IPv4 and 6 host name	IPv4 and 6 host name	Channel binds to IPv4 stack

Table 58. How the IP stack to be used for communication is determined (continued)

Protocols supported	CONNAME	LOCLADDR	Action of channel
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 IPADDRV
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve CONNAME
	IPv4 and 6 host name	IPv4 address	Channel binds to IPv4 stack
	IPv4 address	IPv6 address	Channel maps CONNAME to IPv6 [5]
	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 IPADDRV
IPv6 only	IPv4 address		Channel maps CONNAME to IPv6 [5]
	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 LOCLADDR
	IPv4 address	IPv6 address	Channel maps CONNAME to IPv6 [5]
	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 CONNAME to IPv6 [5]
	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

Table 58. How the IP stack to be used for communication is determined (continued)

Protocols supported	CONNAME	LOCLADDR	Action of channel

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*)
V 8.0.0.4 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.

V 8.0.0.4 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 of 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 distributed platforms, specify a value in the range zero through to the maximum message length for the queue manager.

z/OS 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).

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.

On distributed platforms, 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 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.
- For TCP/IP z/OS, other than z/OS, the user ID from the `inetd.conf` entry, or the user that started the listener.
- For SNA z/OS, other than z/OS, 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 64 characters on Windows and 12 characters on other platforms. 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.

On AIX, HP-UX, Linux, Solaris, 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.

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.

On AIX, HP-UX, Linux, Solaris, 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.

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:

- On UNIX and Linux systems, it is of the form:

```
libraryname(functionname)
```

  The maximum length of the string is 128 characters.
- 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.
- On IBM i, it is of the form:

```
progname libname
```

  where *program name* 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.

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.

On distributed platforms, 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.

V 8.0.0.4 **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.

Message properties	Result
The message contains a property with a prefix of **mcd.**, **jms.**, **usr.** or **mqext.**	All optional message properties (where the **Support** 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 it 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 **Support** 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 **Support** 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 *content='properties'* 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.

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 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.

On distributed platforms, this parameter is valid only for channels with a channel type (CHLTYPE) of RCVR, RQSTR, or CLUSRCVR.

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.

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).

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 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.

QSGDISP	ALTER

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 attempt to refresh local copies on page set zero:
	`DEFINE CHANNEL(channel-name) CHLTYPE(type)`
	`REPLACE QSGDISP(COPY)`
	The ALTER for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with QSGDISP(QMGR) or QSGDISP(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 QSGDISP(QMGR). 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.

On AIX, HP-UX, Linux, Solaris, 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.

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.

On AIX, HP-UX, Linux, Solaris, 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.

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.

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.

On AIX, HP-UX, Linux, Solaris, 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.

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.

On AIX, HP-UX, Linux, Solaris, 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.

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.

SSLCAUTH
Defines whether IBM MQ requires a certificate from the SSL client. The initiating end of the channel acts as the SSL client, so this parameter applies to the end of the channel that receives the initiation flow, which acts as the SSL 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 SSL client.

OPTIONAL
The peer SSL client system might still send a certificate. If it does, the contents of this certificate are validated as normal.

SSLCIPH *(string)*
SSLCIPH specifies the CipherSpec that is used on the channel. The maximum length is 32 characters. This parameter is valid on all channel types which use transport type TRPTYPE(TCP).If the SSLCIPH parameter is blank, no attempt is made to use SSL on the channel.

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 "SSL Cipher Suite". See the SSLCIPH description in "ALTER CHANNEL (MQTT)" .

Specify the name of the CipherSpec you are using. The CipherSpecs that can be used with IBM MQ SSL support are shown in the following table. If a specific named CipherSpec is being used, the **SSLCIPH** values at the two ends of a channel must specify the same named CipherSpec.

Note: IBM i z/OS On z/OS IBM MQ for z/OS; IBM i IBM i; you can also specify the two digit hexadecimal code of a CipherSpec, whether or not it appears in the table. On IBM i, installation of AC3 is a prerequisite for the use of SSL.

Platform support "1" on page 684	CipherSpec name	Protocol used	MAC algorithm	Encryption algorithm	Encryption bits	FIPS "2" on page 684	Suite B
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Yes	No
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_256_CBC_SHA"3" on page 684	TLS 1.0	SHA-1	AES	256	Yes	No
All	ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No
All	ECDHE_ECDSA_AES_256_CBC_SHA384 "3" on page 684	TLS 1.2	SHA-384	AES	256	Yes	No

Platform support "1" on page 684	CipherSpec name	Protocol used	MAC algorithm	Encryption algorithm	Encryption bits	FIPS "2" on page 684	Suite B
distributed	ECDHE_ECDSA_AES_128_GCM_SHA256 "4" on page 684	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	128 bit
distributed	ECDHE_ECDSA_AES_256_GCM_SHA384 "3" on page 684 "4" on page 684	TLS 1.2	AEAD AES-128 GCM	AES	256	Yes	192 bit
All	ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No
All	ECDHE_RSA_AES_256_CBC_SHA384 "3" on page 684	TLS 1.2	SHA-384	AES	256	Yes	No
All	ECDHE_RSA_AES_128_GCM_SHA256 "4" on page 684	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	No
All	ECDHE_RSA_AES_256_GCM_SHA384 "3" on page 684 "4" on page 684	TLS 1.2	AEAD AES-128 GCM	AES	SHA384	Yes	No
IBM i "5" on page 684	ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	AEAD AES-128 GCM	AES	SHA256	Yes	No
IBM i	ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	AEAD AES-128 GCM	3DES	SHA256	Yes	No
IBM i	ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	AEAD AES-128 GCM	3DES	SHA256	Yes	No
IBM i	ECDHE_RSA_RC4_128_SHA256	TLS 1.2	AEAD AES-128 GCM	RSA	SHA256	Yes	No
IBM i	ECDHE_RSA_NULL_SHA256	TLS 1.2	AEAD AES-128 GCM	RSA	SHA256	Yes	No
IBM i	ECDHE_ECDSA_NULL_SHA256	TLS 1.2	AEAD AES-128 GCM	ECDSA	SHA256	Yes	No
IBM i	ECDHE_ECDSA_AES_256_GCM_SHA384 "3" on page 684 "4" on page 684	TLS 1.2	AEAD AES-128 GCM	AES	SHA384	Yes	No
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No

Platform support "1" on page 684	CipherSpec name	Protocol used	MAC algorithm	Encryption algorithm	Encryption bits	FIPS "2" on page 684	Suite B
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_256_CBC_SHA256 "3" on page 684	TLS 1.2	SHA-256	AES	256	Yes	No
distributed	TLS_RSA_WITH_AES_128_GCM_SHA256 "4" on page 684	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	No
distributed	TLS_RSA_WITH_AES_256_GCM_SHA384 "3" on page 684 "4" on page 684	TLS 1.2	AEAD AES-128 GCM	AES	256	Yes	No

Notes:

1. If no specific platform is noted, the CipherSpec is available on all platforms.
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. This CipherSpec cannot be used to secure a connection from the MQ Explorer to a queue manager unless the appropriate unrestricted policy files are applied to the JRE used by the Explorer.
4. Following a recommendation by NIST, GCM CipherSpecs have a restriction which means that after 2^32 TLS records are sent, using the same session key, the connection is terminated with message AMQ9288.

   To prevent this error from happening: avoid using GCM Ciphers, enable secret key reset, or start your IBM MQ queue manager with the environment variable GSK_ENFORCE_GCM_RESTRICTION=GSK_FALSE set.

   Important: The GCM restriction is active, regardless of the FIPS mode being used.
5. The CipherSpecs listed as supported on IBM i apply to Versions 7.2 and 7.3 of IBM i.

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 z/OS, UNIX, Linux, and Windows systems, 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.
- On UNIX, Linux, and Windows systems, 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 SSL 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 SSL or TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different SSL or 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 start up. (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 59. Attribute types supported by SSLPEER.

A two column table descibing the attributes supported by the SSLPEER attribute

Summary attribute	**Description**
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 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 Windows, IBM i, UNIX and Linux platforms

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
It 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.

On z/OS, 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.

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.

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.

On Windows SNA Server, z/OS 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.

V 8.0.0.4 **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

TRPTYPE
Transport type to be used.

On AIX, IBM i, HP-UX, Linux, Solaris, and Windows z/OS, and z/OS, 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
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
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

V 8.0.0.4 **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.

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.

On distributed platforms 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.

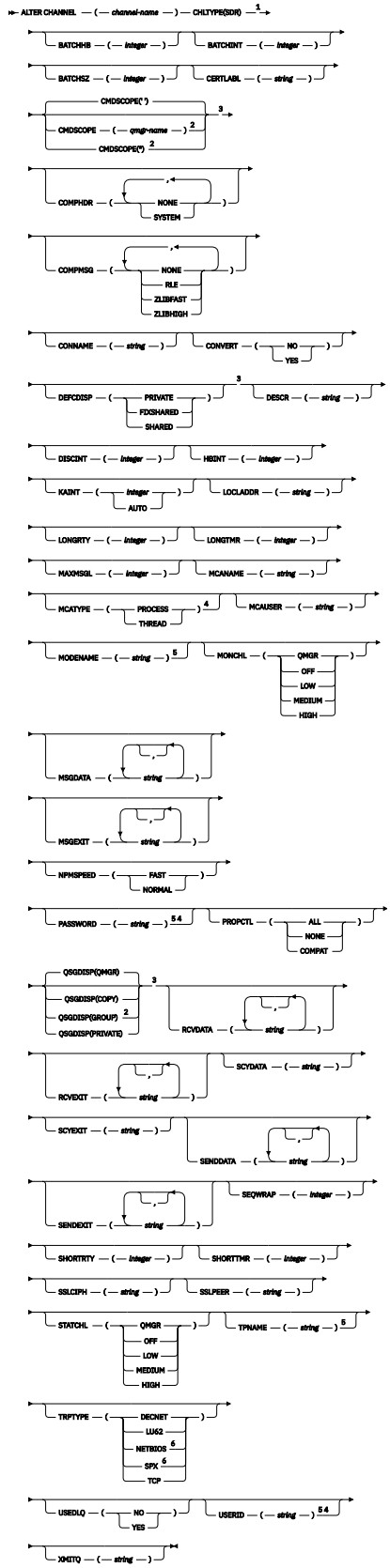
This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR. For these channel types, this parameter is required.

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.

ALTER CHANNEL



Notes:

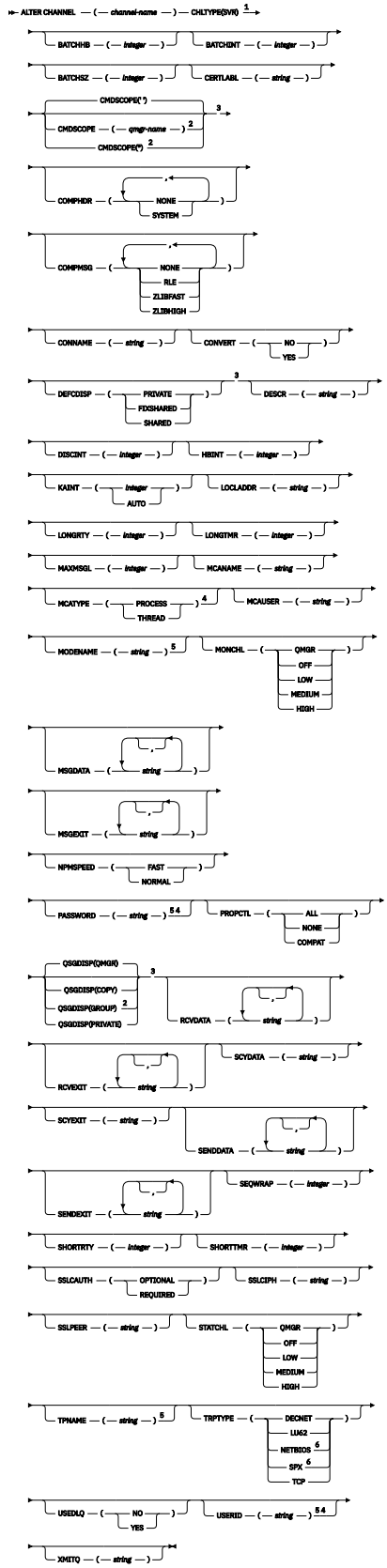
[1] This parameter must follow immediately after the channel name except on z/OS.
[2] Valid only on z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Not valid on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Valid only Windows.

The parameters are described in "ALTER CHANNEL" on page 653.

Server channel

Syntax diagram for a server channel when using the ALTER CHANNEL command.

ALTER CHANNEL



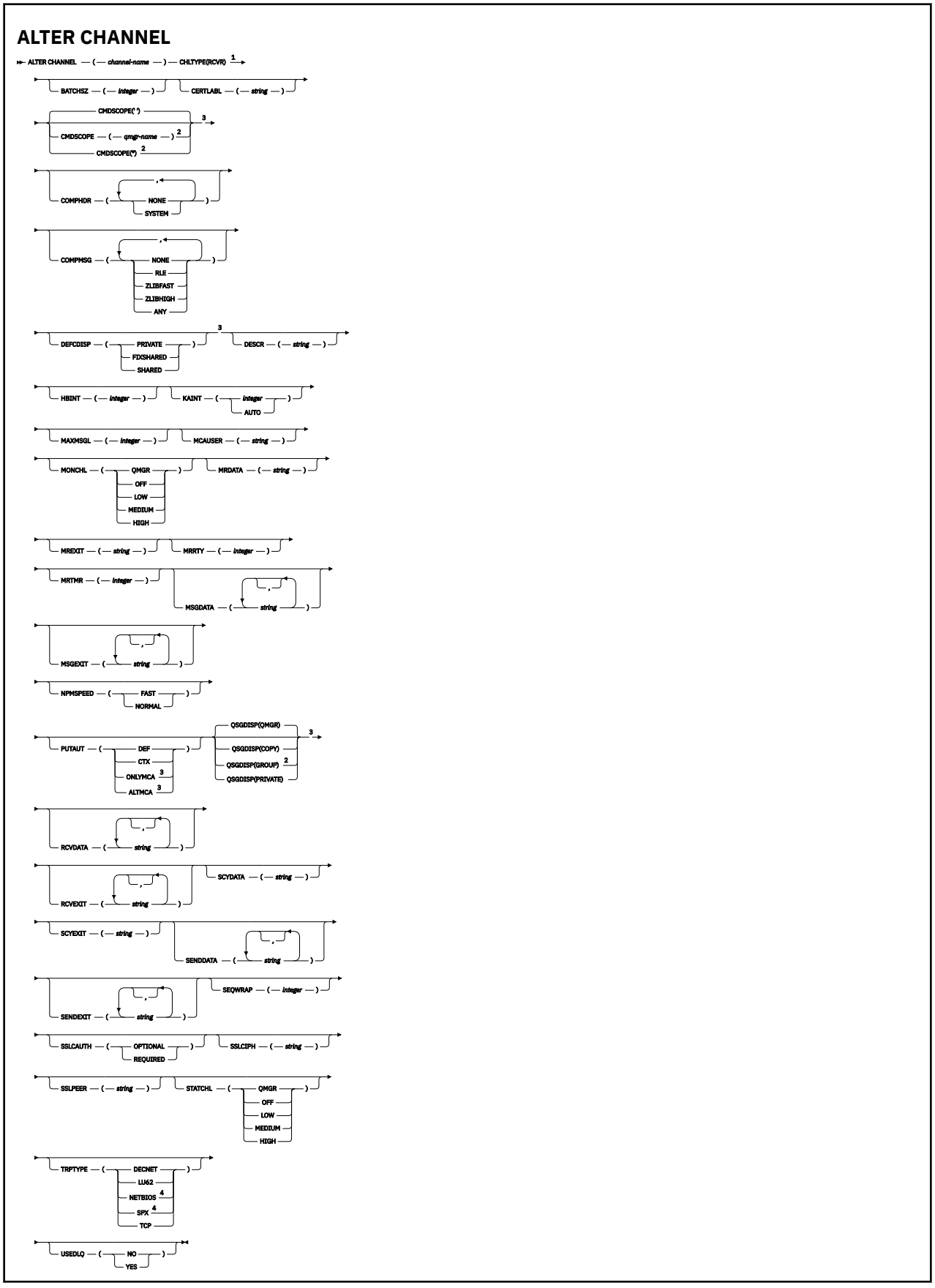
Notes:

[1] This parameter must follow immediately after the channel name except 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] Not valid on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Valid only on Windows.

The parameters are described in "ALTER CHANNEL" on page 653.

Receiver channel

Syntax diagram for a receiver channel when using the ALTER CHANNEL command.



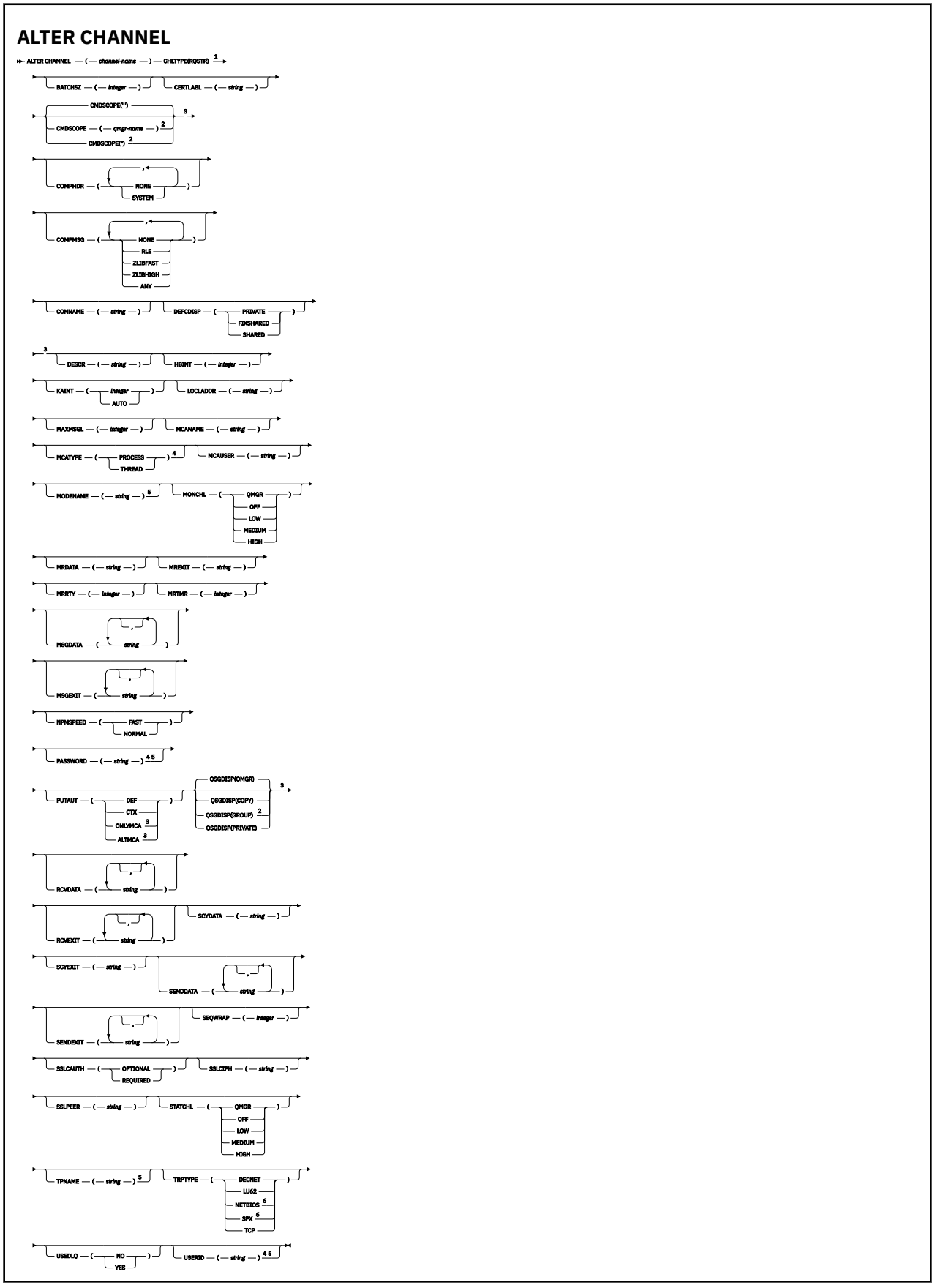
Notes:

[1] This parameter must follow immediately after the channel name except 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 Windows.

The parameters are described in “ALTER CHANNEL” on page 653.

Requester channel

Syntax diagram for a requester channel when using the ALTER CHANNEL command.



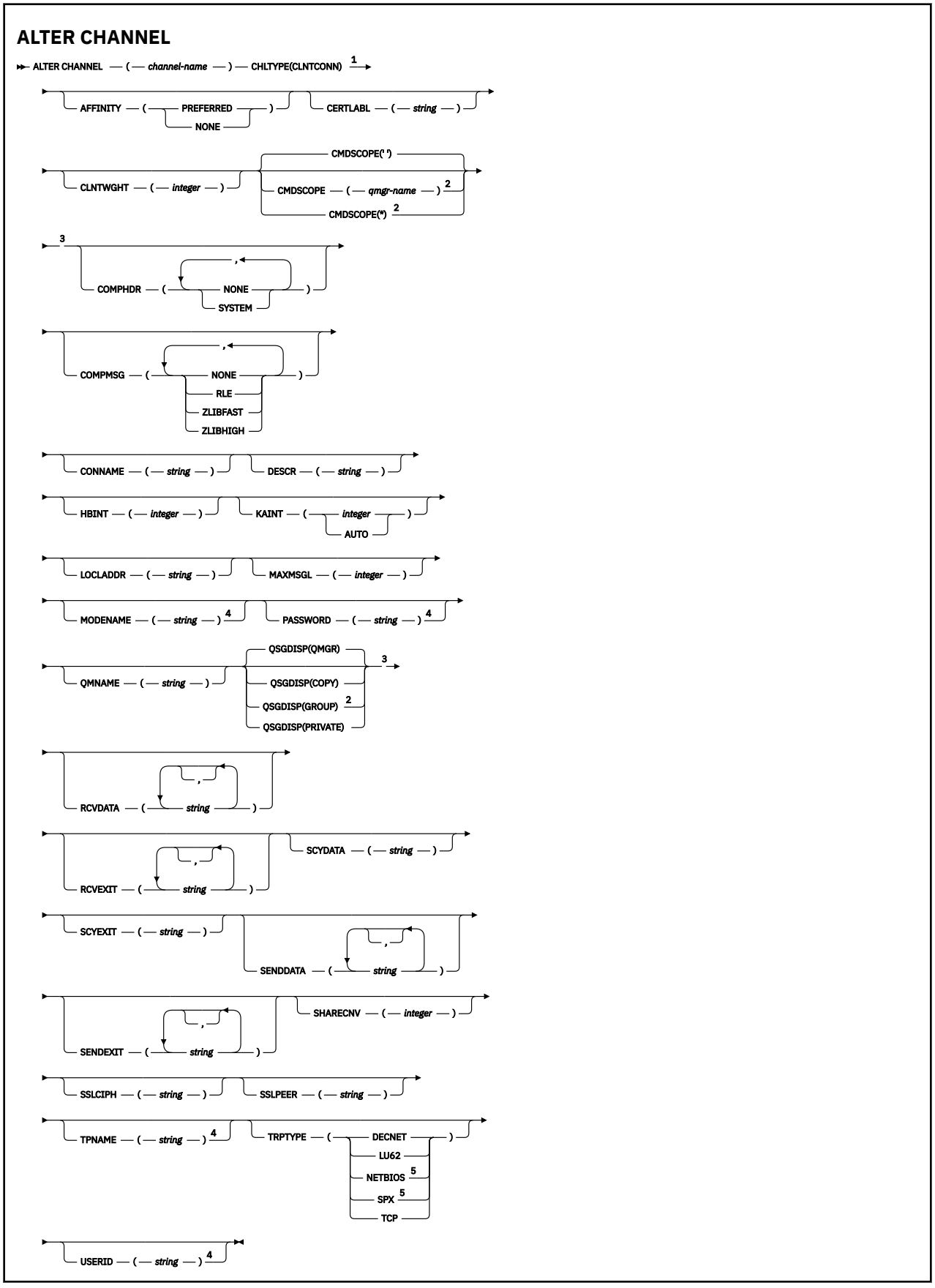
Notes:

[1] This parameter must follow immediately after the channel name except 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] Not valid on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Valid only on Windows.

The parameters are described in "ALTER CHANNEL" on page 653.

Client-connection channel

Syntax diagram for a client-connection channel when using the ALTER CHANNEL command.



Notes:

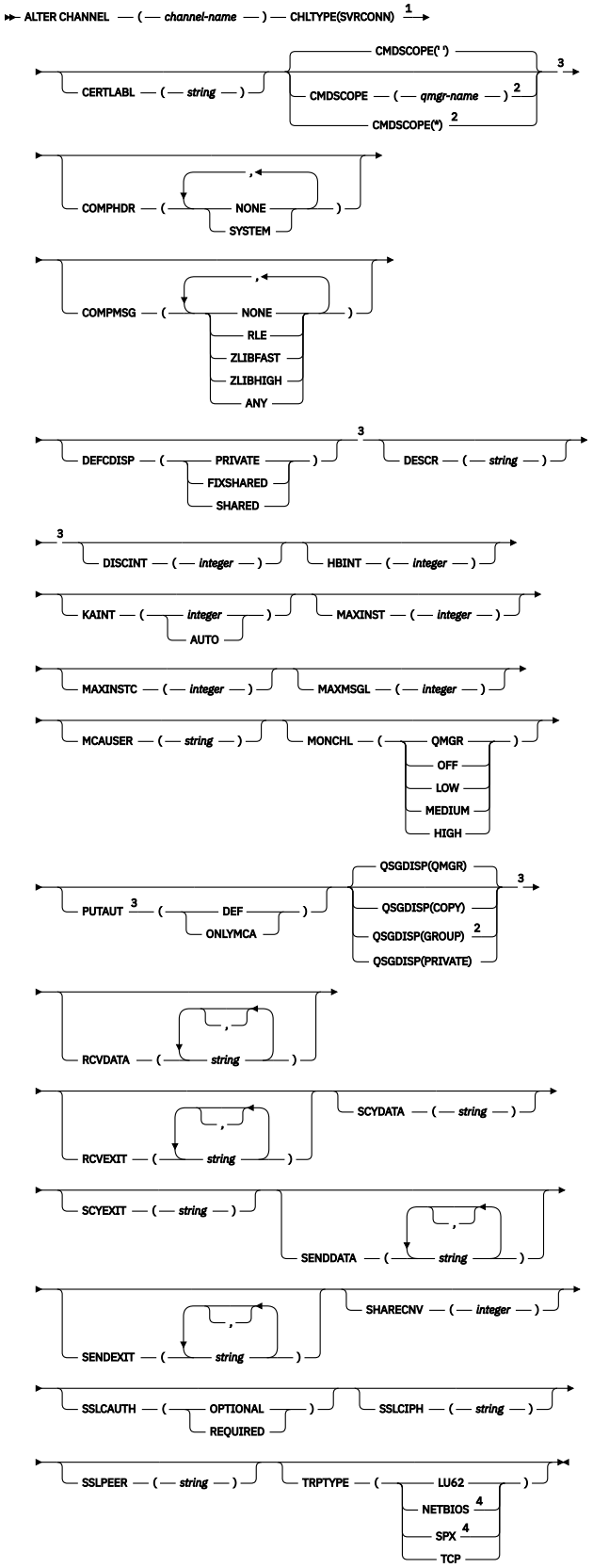
[1] This parameter must follow immediately after the channel name except on z/OS.
[2] Valid only on z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Valid only if TRPTYPE is LU62.
[5] Valid only for clients to be run on DOS and Windows.

The parameters are described in “ALTER CHANNEL” on page 653.

Server-connection channel

Syntax diagram for a server-connection channel when using the ALTER CHANNEL command.

ALTER CHANNEL



Notes:

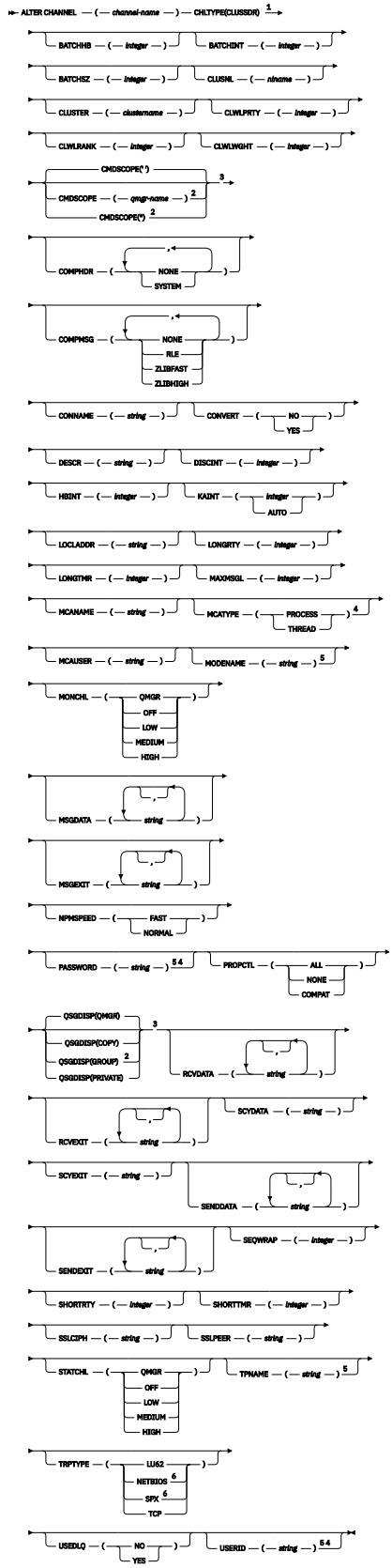
[1] This parameter must follow immediately after the channel name except on z/OS.
[2] Valid only on z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Valid only for clients to be run on Windows.

The parameters are described in "ALTER CHANNEL" on page 653.

Cluster-sender channel

Syntax diagram for a cluster-sender channel when using the ALTER CHANNEL command.

ALTER CHANNEL



Notes:

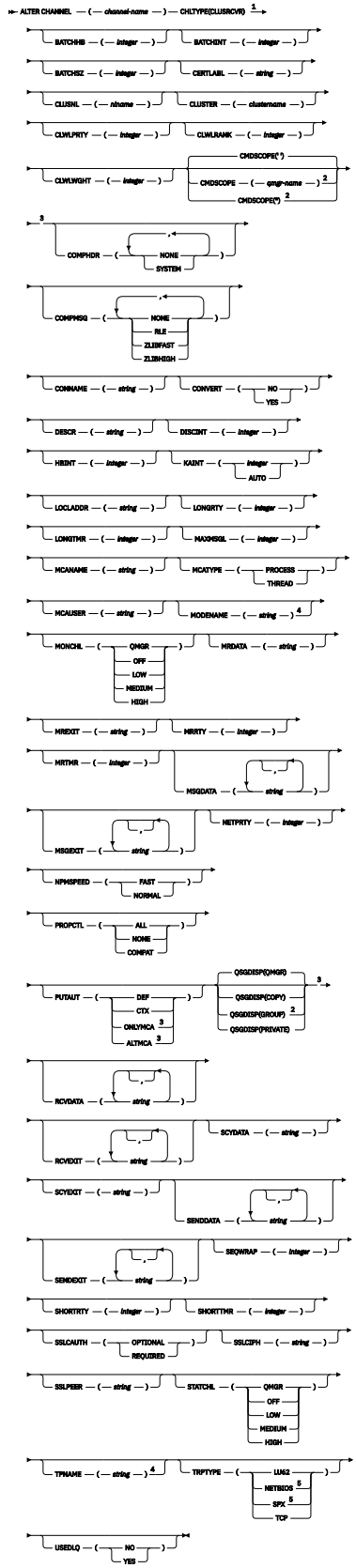
[1] This parameter must follow immediately after the channel name except on z/OS.
[2] Valid only on z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Not valid on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Valid only Windows.

The parameters are described in "ALTER CHANNEL" on page 653.

Cluster-receiver channel

Syntax diagram for a cluster-receiver channel when using the ALTER CHANNEL command.

ALTER CHANNEL



Notes:

[1] This parameter must follow immediately after the channel name except 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 if TRPTYPE is LU62.
[5] Valid only on Windows.

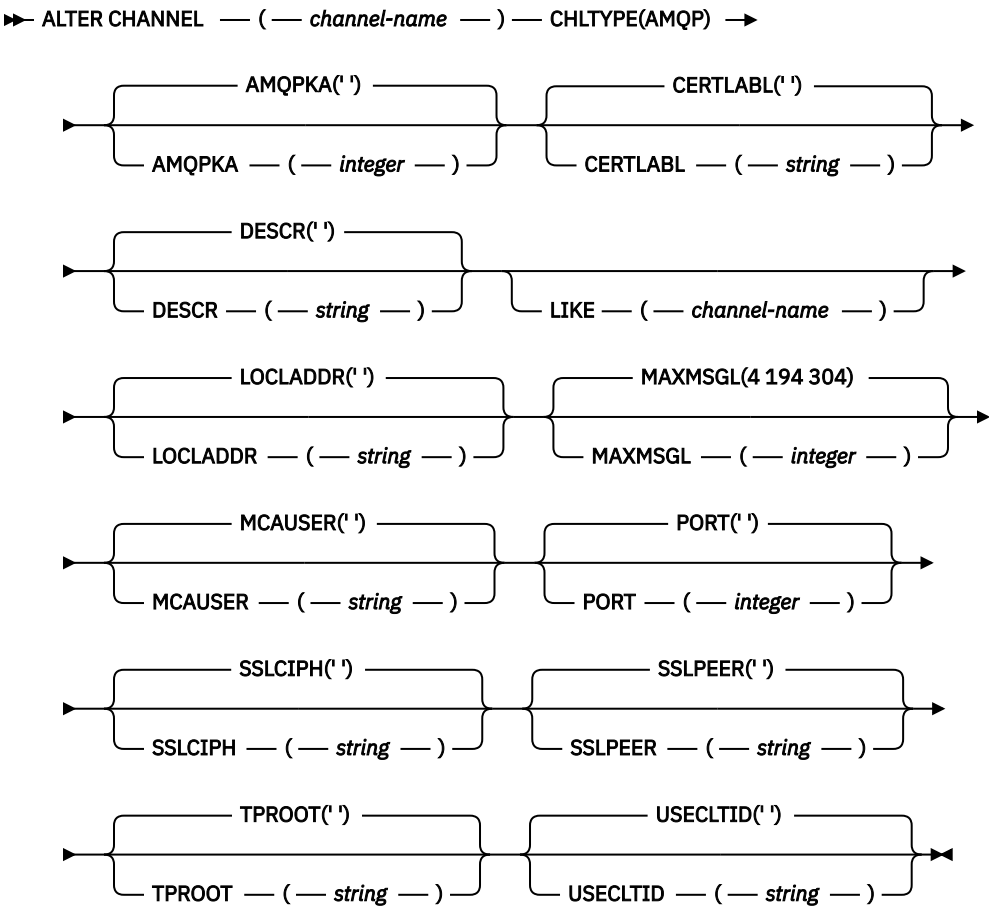
The parameters are described in "ALTER CHANNEL" on page 653.

distributed *AMQP channel*

Syntax diagram for an AMQP channel when using the ALTER CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



The parameters are described in "ALTER CHANNEL" on page 653.

ALTER CHANNEL (MQTT)

Syntax diagram for a telemetry channel when using the **ALTER CHANNEL** command.

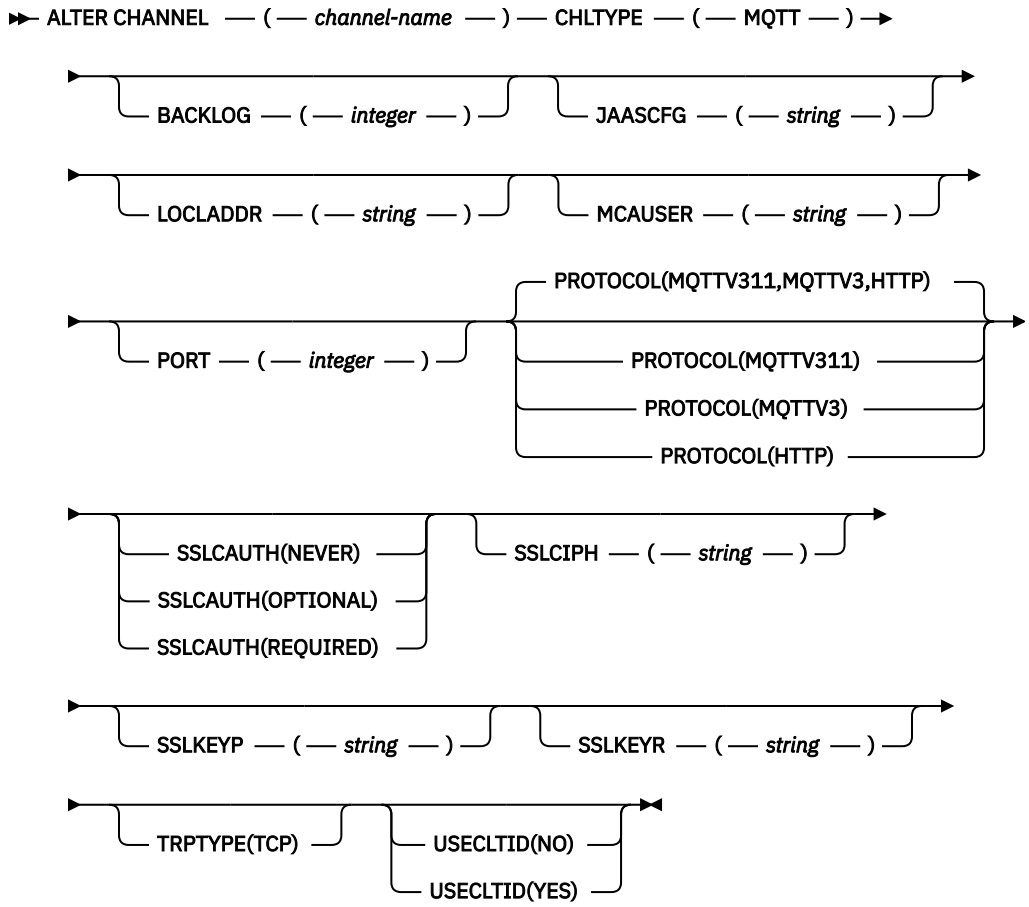
Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	AIX and Linux	Windows	z/OS
	✓	✓	

Note: For the telemetry server, AIX is the only supported UNIX platform.

ALTER CHANNEL (MQTT)



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:

V 8.0.0.3 **MQTTV311**
The channel accepts connections from clients using the protocol defined by the MQTT Version 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 IBM 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.

V 8.0.0.3 If you are using IBM MQ Version 8.0.0, Fix Pack 3 or later, and your configuration includes an MQTT channel that was last modified in an earlier version of IBM MQ, 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 SSL client. The initiating end of the channel acts as the SSL client, so this parameter applies to the end of the channel that receives the initiation flow, which acts as the SSL server.

NEVER
IBM MQ never requests a certificate from the SSL client.

REQUIRED
IBM MQ requires and validates a certificate from the SSL client.

OPTIONAL
IBM MQ lets the SSL 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 "SSL Cipher Suite". The SSL cipher suite is the one supported by the JVM that is running the telemetry (MQXR) service. If the `SSLCIPH` parameter is blank, no attempt is made to use SSL 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 Secure Sockets Layer (SSL) key repository.

SSLKEYR ***(string)***

The full path name of the Secure Sockets Layer (SSL) key repository file, the store for digital certificates and their associated private keys. If you do not specify a key file, SSL is not used.

The maximum length of the string is 256 characters. On UNIX and Linux, the name is of the form *pathname/keyfile* and on Windows *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 SSL
Telemetry channel configuration for channel authentication using SSL
CipherSpecs and CipherSuites
System requirements for using SHA-2 cipher suites with MQTT channels
Related reference
“DEFINE CHANNEL (MQTT)” on page 890
Syntax diagram for a telemetry channel when using the **DEFINE CHANNEL** command.

ALTER COMMINFO

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓

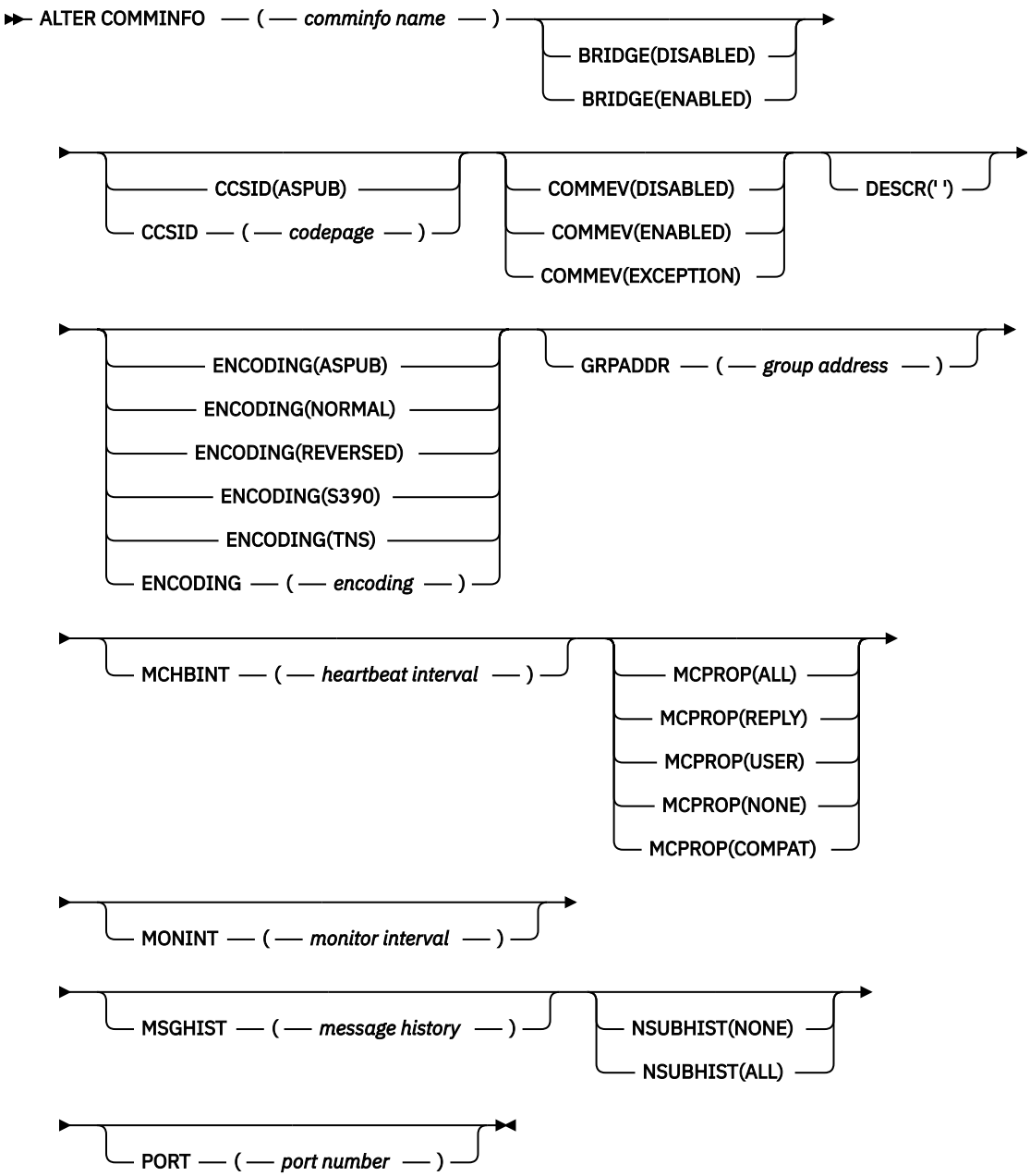
Parameters not specified in the ALTER COMMINFO command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for ALTER COMMINFO” on page 709

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 will only be 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” on page 1079).

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 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.

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.

ALTER LISTENER

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 Performing local administration tasks using MQSC commands.

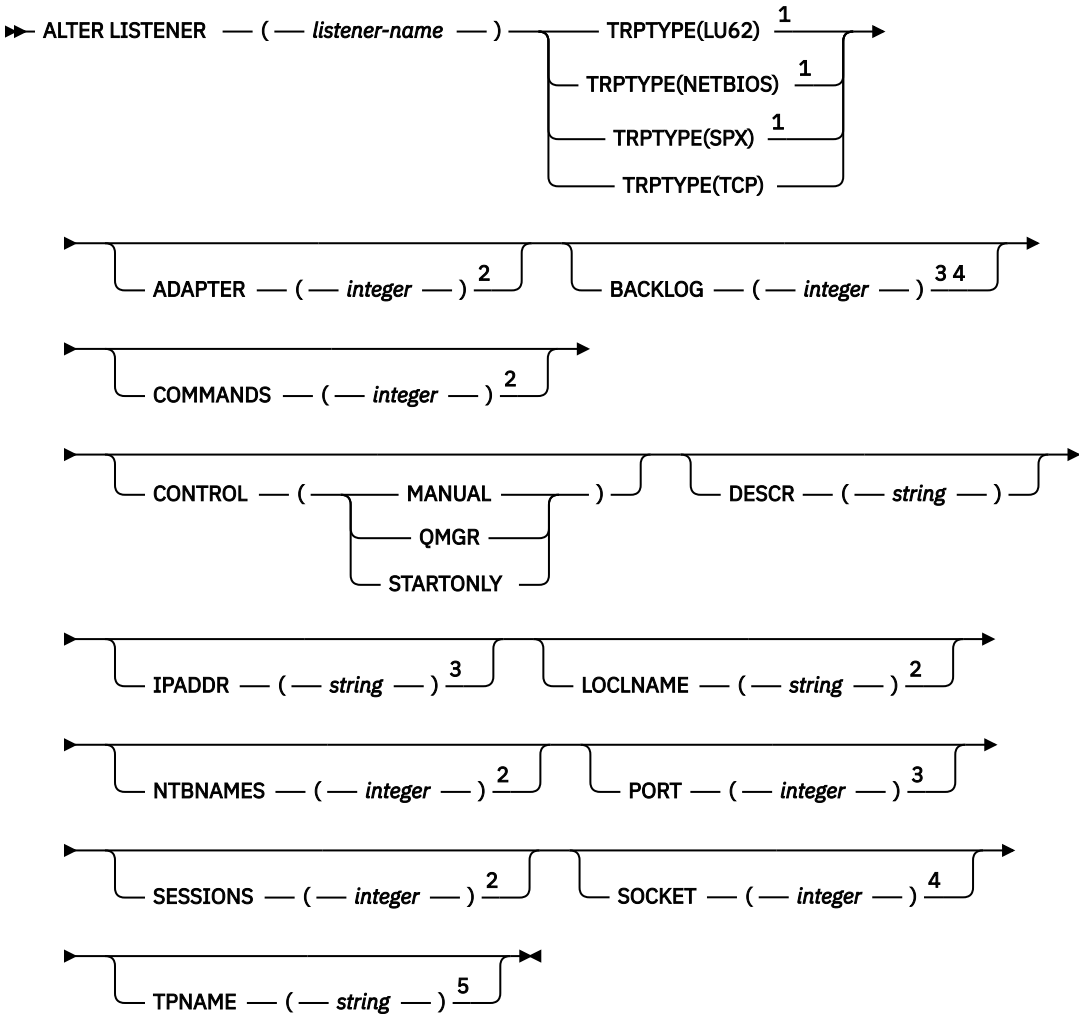
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

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 713

Synonym: ALT LSTR

ALTER LISTENER



Notes:

[1] Valid only on Windows.
[2] Valid only on Windows when TRPTYPE is NETBIOS.
[3] Valid when TRPTYPE is TCP.
[4] Valid on Windows when TRPTYPE is SPX.
[5] 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).

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.

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” on page 1099).

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.

LOCLNAME(*string*)
The NetBIOS local name that the listener uses. This parameter is valid only on Windows when TRPTYPE is NETBIOS.

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.

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.

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:

LU62
SNA LU 6.2. This is valid only on Windows.

NETBIOS
NetBIOS. This is valid only on Windows.

SPX
Sequenced packet exchange. This is valid only on Windows.

TCP
TCP/IP.

ALTER NAMELIST

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

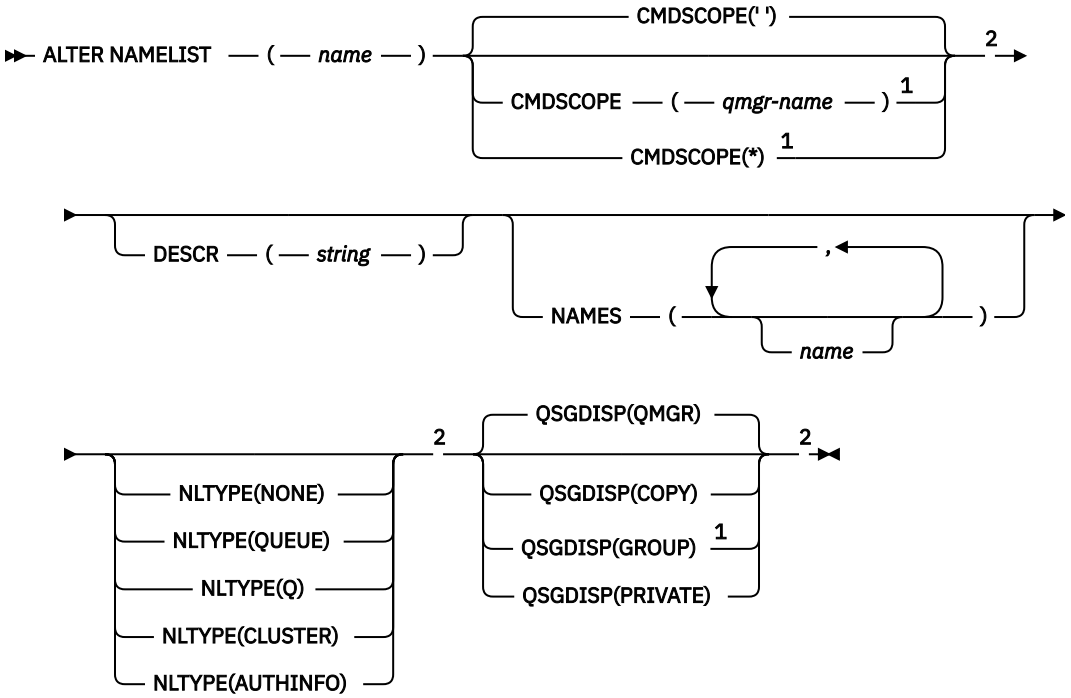
Parameters not specified in the ALTER NAMELIST command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 716
- "Parameter descriptions for ALTER NAMELIST" on page 716

Synonym: ALT NL

ALTER NAMELIST



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.

Usage notes

- On UNIX systems, the command is valid only on AIX, HP-UX, Linux and Solaris.
- 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 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 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" on page 1108).

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.

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 SSL and contains a list of authentication information object names.

Namelists used for clustering must have NLTYPE(CLUSTER) or NLTYPE(NONE).

Namelists used for SSL 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 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.

QSGDISP	ALTER

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 attempt to refresh local copies on page set zero:
	`DEFINE NAMELIST(name)`
	`REPLACE QSGDISP(COPY)`
	The ALTER for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with QSGDISP(QMGR) or QSGDISP(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 QSGDISP(QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

ALTER PROCESS

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

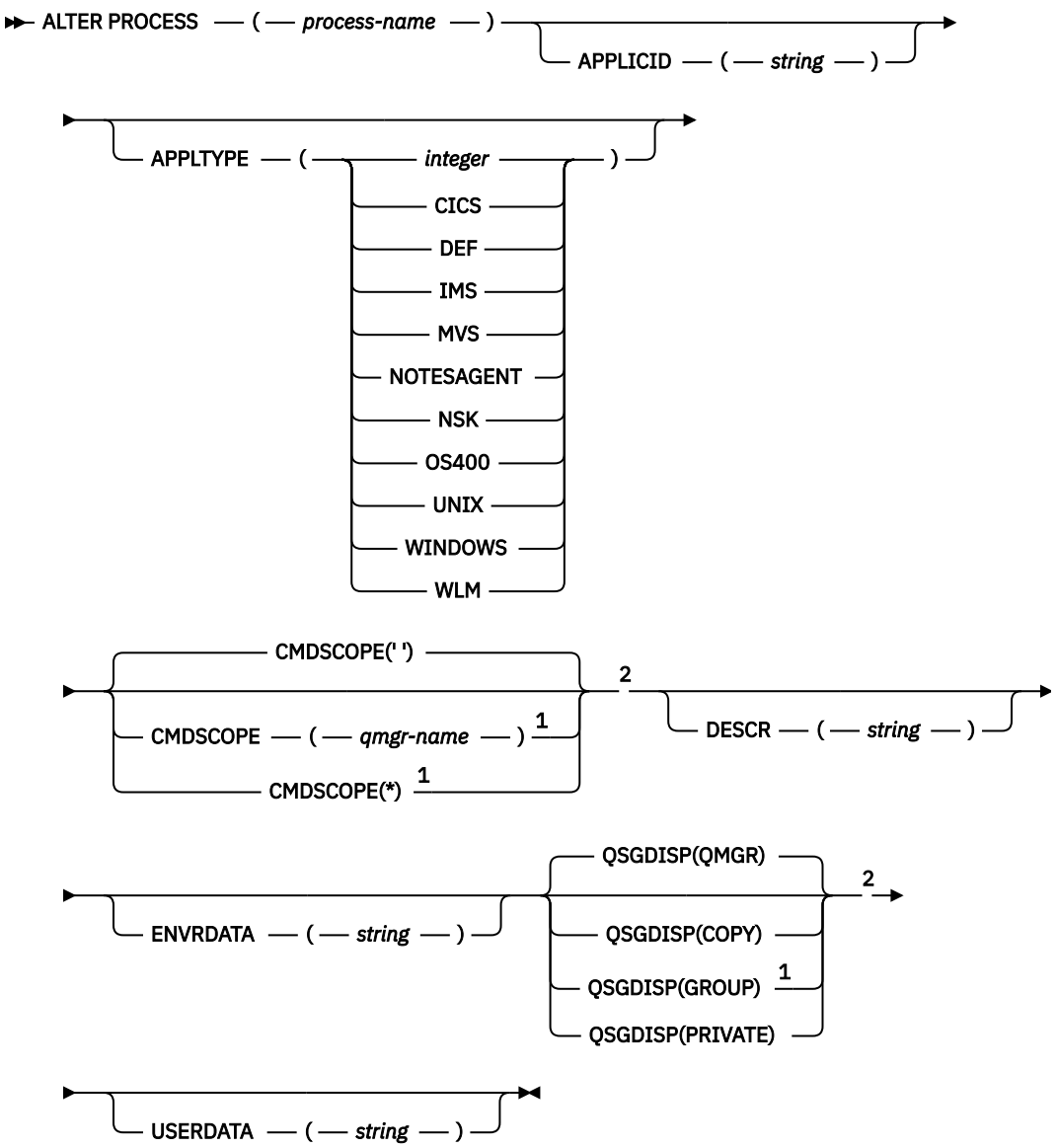
Parameters not specified in the ALTER PROCESS command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for ALTER PROCESS” on page 719

Synonym: ALT PRO

ALTER PROCESS



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.

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.

NSK
Represents an HP Integrity NonStop Server application.

OS400
Represents an IBM i application.

UNIX
Represents a UNIX 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 executed:

- 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
- On UNIX: `UNIX`, `WINDOWS`, `CICS`, and `DEF` are supported
- On Windows: `UNIX`, `WINDOWS`, `CICS`, and `DEF` are supported

z/OS **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.

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 is executed 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:

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 UNIX systems, 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).

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 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). 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.
	`DEFINE PROCESS(process-name)`
	`REPLACE QSGDISP(COPY)`
	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 ALTER for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with QSGDISP(QMGR) or QSGDISP(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 QSGDISP(QMGR). 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.

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

Use the MQSC command ALTER PSID to change the expansion method for a page set.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			CR

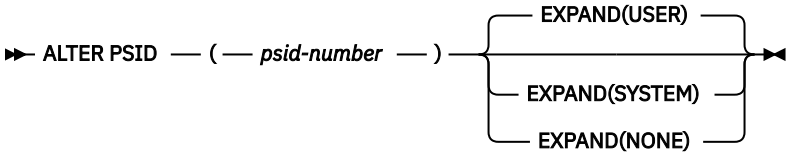
Parameters not specified in the ALTER PSID command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for ALTER PSID” on page 723

Synonym: ALT PSID

ALTER 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 pageset from being expanded; for example, once 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 pageset 3 configured with EXPAND(SYSTEM), you issue the command ALTER PSID(3) EXPAND(SYSTEM) to allow IBM MQ to retry pageset expansion.

ALTER QMGR

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

Parameters not specified in the **ALTER QMGR** command result in the existing values for those parameters being left unchanged.

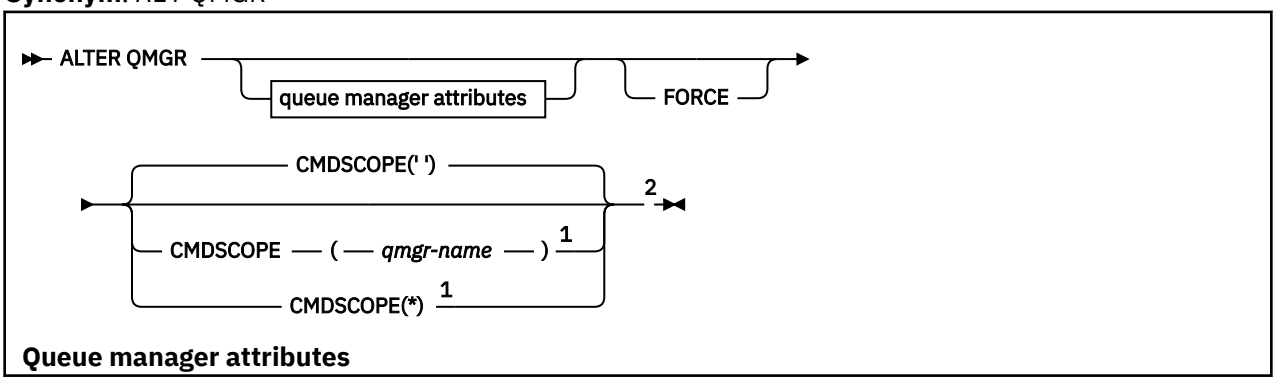
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

This information is divided into three sections:

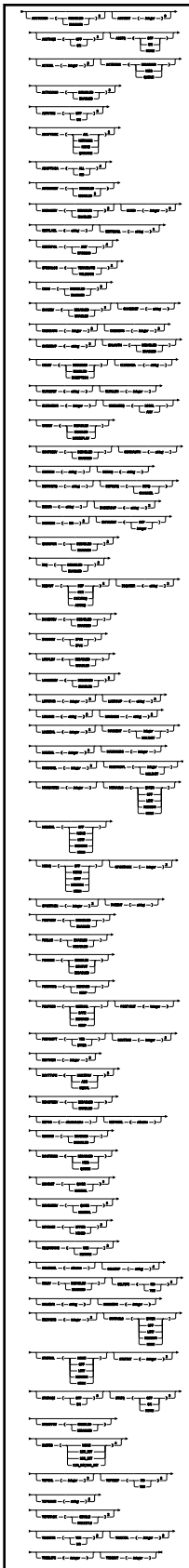
- “ALTER QMGR” on page 724
- “Parameter descriptions for ALTER QMGR” on page 726
- “Queue manager parameters” on page 726

ALTER QMGR

Synonym: ALT QMGR



Queue manager 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] Valid only on UNIX, Linux, and Windows.
[5] 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 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:

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 IBM i, UNIX, Linux, and Windows systems.

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 IBM i, UNIX, Linux, and Windows systems.

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 IBM i, UNIX, Linux, and Windows systems.

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 on z/OS only.

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.

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 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 IBM i, UNIX, Linux, and Windows systems.

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 IBM i, UNIX, Linux, and Windows systems.

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.

This parameter is valid on z/OS only.

Changes to this parameter take effect the next time that a channel attempts to adopt an MCA.

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.

This parameter is valid on z/OS only

Changes to this parameter take effect the next time that a channel attempts to adopt an MCA.

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 on z/OS only.

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.

z/OS This parameter is not valid on z/OS. To carry out the equivalent tasks on z/OS, use CSQ6SYSP to set your system parameters.

See Code page conversion for details of the supported CCSIDs for each platform.

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:

- z/OS On z/OS: *ibmWebSphereMQXXXX* where *XXXX* is the queue manager name.

  See z/OS systems for more information.
- 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 UNIX and Linux and Windows, *ibmwebspheremqxxxx* where *xxxx* is the queue manager name folded to lowercase.

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 **CERTQSGL**

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 QSG name.

This parameter is valid on z/OS only.

See z/OS systems for more information.

CERTVPOL

Specifies which SSL/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.

This parameter is valid on only UNIX, Linux, and Windows. Changes to the parameter take effect only after a **REFRESH SECURITY TYPE(SSL)** command is issued.

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 on z/OS only.

All queue managers in the queue-sharing group must be at command level 710 or greater and OPMODE set to NEWFUNC for **TOLERATE** to be selected.

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.

z/OS This parameter is not valid on z/OS.

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.

z/OS This parameter is not valid on z/OS.

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:

- On Windows, it is of the form *dllname(functionname)* where *dllname* is specified without the suffix `.DLL`. The maximum length is 128 characters.
- On IBM i, it is of the form:

  ```
  progname 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.
- On UNIX, and Linux, it is of the form *libraryname(functionname)* . The maximum length is 128 characters.
- z/OS On z/OS, it is a load module name, the maximum length is eight characters.

z/OS On z/OS, this 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:

- Test system: 8
- Production system: 30

This parameter is valid on z/OS only.

Changes to this parameter take effect when the channel initiator is restarted.

For more information about the relationship between `CHIADAPS`, `CHIDISPS` and `MAXCHL`, see Task 18: Tailor the channel initiator parameters.

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:

- Test system: 5
- Production system: 20

This parameter is valid on z/OS only.

Changes to this parameter take effect when the channel initiator is restarted.

z/OS For more information about the relationship between CHIADAPS, CHIDISPS and MAXCHL, see Task 18: Tailor the channel initiator parameters.

z/OS **CHISERVP**
This parameter is reserved for IBM use only; it is not for general use.

This parameter is valid on z/OS only.

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:

- On UNIX and Linux systems, it is of the form *libraryname(functionname)*. The maximum length is 128 characters.
- 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.

- On IBM i, it is of the form:

```
progname 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.

This parameter is valid only on IBM i, z/OS, UNIX, Linux, and Windows.

CLWLLEN (*integer*)
The maximum number of bytes of message data that is passed to the cluster workload exit.

Specify a value in the range:

- z/OS 0 - 100 MB on IBM MQ for z/OS systems
- 0 - 999,999,999 on distributed platforms

This parameter is valid only on IBM i, z/OS, UNIX, Linux, and Windows.

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**
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.

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.

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 the OAM (on UNIX, Linux, and Windows platforms) z/OS or the security component (on z/OS platforms) reads the configuration. 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 that has either REQDADM z/OS (on platforms other than z/OS) or REQUIRED set in the CHCKCLNT field, the connection fails.

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 and 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. This behavior is implicit in versions of IBM WebSphere MQ, earlier than Version 7.5. In earlier versions, the queue manager attribute 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.

z/OS **DNSGROUP (*string*)**
This parameter is no longer used. Refer to WLM/DNS no longer supported. This parameter is valid on z/OS only.

z/OS **DNSWLM**
This parameter is no longer used. Refer to WLM/DNS no longer supported.

NO
This value is the only value accepted.

This parameter is valid on z/OS only.

z/OS **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.

This parameter is valid only on z/OS. 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.

z/OS **IGQ**

Specifies whether intra-group queuing is used.

This 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, issue ALTER QMGR IGQ (ENABLED) to restart it.

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.

This 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.

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.

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.

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 CONNAME and LOCLADDR parameters of the channel is a host name that resolves to both an IPv4 and IPv6 address.

LOCALEV
Specifies whether local error events are generated:

ENABLED
Local error events are generated.

DISABLED
Local error events are not generated.

This is the queue manager's initial default value.

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 IBM i, UNIX, Linux, and Windows.

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.

This parameter is valid on z/OS only.

Changes to this parameter take effect for listeners that are later started. Listeners that are currently started are unaffected by changes to this parameter.

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.

This parameter is valid on z/OS only.

Changes to this parameter take effect for listeners that are later started. Listeners that are currently started are unaffected by changes to this parameter.

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

This parameter is valid on z/OS only.

Changes to this parameter take effect when the channel initiator is restarted.

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.

This parameter is valid on z/OS only.

Changes to this parameter take effect when the channel initiator is restarted.

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 on z/OS only.

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 mark-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.

z/OS **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:

- 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.

This parameter is valid on z/OS only.

z/OS For more information about the relationship between CHIADAPS, CHIDISPS, and MAXCHL, see Task 18: Tailor the channel initiator parameters.

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, IBM MQ 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.

z/OS 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 IBM MQ Telemetry. IBM 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 switched off.

LOW
Unless MONCHL is NONE, monitoring is switched on 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 switched on with a moderate rate of data collection with limited effect on system performance.

HIGH
Unless MONCHL is NONE, monitoring is switched on 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.

z/OS **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.

This parameter is valid on z/OS only.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

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.

This parameter is valid on z/OS only.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

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.

z/OS 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 DISPLAY PUBSUB, or on IBM i **DSPMQM**, to determine the current state of the publish/subscribe engine and 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. 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 WebSphere Message Broker (now known as IBM Integration Bus) 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 PSSYNCPT, 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 IBM i, UNIX, Linux, and Windows systems, 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.

PSRTYCNT
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.

z/OS **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 **RCVTTYPE** 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 **RCVTTYPE** 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 **RCVTTYPE** 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.

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 **RCVTTYPE** 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 **RCVTTYPE** 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.

z/OS **RCVTTYPE**

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.

This parameter is valid on z/OS only.

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.

REMOTEEV

Specifies whether remote error events are generated:

DISABLED
Remote error events are not generated.

This is the queue manager's initial default value.

ENABLED
Remote error events are generated.

z/OS If you are using the reduced function form of IBM MQ for z/OS supplied with WebSphere Application Server, only DISABLED is valid.

REPOS (*clustername*)
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.

This parameter is valid on IBM i, z/OS z/OS, UNIX, Linux, and Windows.

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**.

This parameter is valid on IBM i, z/OS z/OS, UNIX, Linux, and Windows.

REVDNS
Controls whether reverse lookup of the hostname 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.

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 IBM i, UNIX, Linux, and Windows.

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 IBM i, UNIX, Linux, and Windows systems.

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. MQCMDS 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:

```
REFFRESH 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/SSL certificate checking.

If SSLCRLNL is blank, certificate revocation checking is not invoked unless one of the SSL 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 either:

- On IBM i, UNIX, Linux, and Windows systems when a new channel process is started.
- For channels that run as threads of the channel initiator on IBM i, UNIX, Linux, and Windows systems, when the channel initiator is restarted.
- For channels that run as threads of the listener on IBM i, UNIX, Linux, and Windows systems, 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.
- 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 LDAPCRL 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 IBM Global Security Kit (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_OF F.

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 SSL 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 SSL events are generated.

DISABLED
SSL events are not generated.

This is the queue manager's initial default value.

ENABLED
All SSL events are generated.

SSLFIPS

This parameter is valid only on z/OS z/OS, UNIX, Linux, and Windows systems.

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 SSL 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 UNIX, Linux, and Windows systems, when a new channel process is started.
- For channels that run as threads of the channel initiator on UNIX, Linux, and Windows systems, when the channel initiator is restarted.
- For channels that run as threads of the listener on UNIX, Linux, and Windows systems, 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 an SSL channel. If the process pooling process has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command REFRESH SECURITY TYPE(SSL). The process pooling process is **amqrmppa** on UNIX, Linux, and Windows systems.
- z/OS On z/OS, when the channel initiator is restarted.
- When a REFRESH SECURITY TYPE(SSL) command is issued z/OS, except on z/OS.

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.

On IBM i, UNIX, Linux, and Windows, the name is in stem format, which means that it includes the full path and the file name without an extension:

- IBM i On IBM i, the name is of the form *pathname/keyfile*, where *keyfile* is specified without the suffix .kdb, and identifies a GSKit key database file.

  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 UNIX On UNIX and Linux, the name is of the form *pathname/keyfile* where *keyfile* is specified without the suffix .kdb and identifies a GSKit CMS key database file.
- Windows On Windows, the name is of the form *pathname\keyfile* where *keyfile* is specified without the suffix .kdb and identifies a GSKit CMS key database file.

On IBM i, UNIX, Linux, and Windows, the syntax of this parameter is validated to ensure that it contains a valid, absolute, directory path.

If SSLKEYR is blank, channels using SSL fail to start. If SSLKEYR is set to a value that does not correspond to a key ring or key database file, channels using SSL also fail to start.

Changes to SSLKEYR become effective as follows:

- On IBM i, UNIX, Linux, and Windows systems, when a new channel process is started.
- For channels that run as threads of the channel initiator on IBM i, UNIX, Linux, and Windows systems, when the channel initiator is restarted.
- For channels that run as threads of the listener on IBM i, UNIX, Linux, and Windows systems, 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 an SSL channel. If the process pooling process has already run an SSL 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.
- When a REFRESH SECURITY TYPE(SSL) command is issued.

SSLRKEYC (*integer*)

The number of bytes to be sent and received within an SSL conversation before the secret key is renegotiated. The number of bytes includes control information.

SSLRKEYC is used only by SSL 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 an SSL/TLS secret key reset count in the range 1 - 32767 bytes (32 KB), SSL/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 SSL/TLS secret key reset values.

Attention: 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 SSL calls. To use SSL channels, you must have at least two of these tasks running.

This parameter is valid only on z/OS.

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.

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 switched off.

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.

On z/OS, 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.

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.

On z/OS, 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.

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.

This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

Changes to this parameter take immediate effect on the collection of monitoring and statistics data.

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.

This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

Changes to this parameter take immediate effect on the collection of monitoring and statistics data.

STATQ
Specifies whether statistics data is to be collected for queues:

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.

This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

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 CHIDISPS. The z/OS UNIX System Services MAXFILEPROC parameter (specified in the BPXPRM *xx* 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 CHIDISPS.

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 on z/OS only.

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.

This parameter is valid on z/OS only.

Changes to this parameter take effect for channels that are later started. Channels that are currently started are unaffected by changes to this parameter.

Using the TCPKEEP parameter is no longer required for 'modern' queue managers. The replacement is a combination of:

- using 'modern' client channels (SHARECNV <> 0); and
- 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 the following address: https://www.ibm.com/support/docview.wss?uid=swg21216834

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 BPXPRM *xx* 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.

This parameter is valid on z/OS only.

Changes to this parameter take effect when the channel initiator is restarted.

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.

This parameter is valid on z/OS only.

Changes to this parameter take effect when the channel initiator is restarted.

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.

This parameter is valid on z/OS only.

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.

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.

This parameter is valid on z/OS only.

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.

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" on page 938 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

z/OS Working with SSL or TLS on z/OS

ALTER queues

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 Performing local administration tasks using MQSC commands.

This section contains the following commands:

- "ALTER QALIAS" on page 777
- "ALTER QLOCAL" on page 779
- "ALTER QMODEL" on page 782
- "ALTER QREMOTE" on page 784

These commands are supported on the following platforms:

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

Parameters not specified in the **ALTER** queue commands result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on 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 60 on page 756. Each parameter is described after the table.

Table 60. DEFINE and ALTER QUEUE parameters.

Cross-tabulation of queue parameters and queue types. If the parameter applies to the queue type, the cell contains a check mark.

Parameter	Local queue	Model queue	Alias queue	Remote queue
ACCTQ	✓	✓		
BOQNAME	✓	✓		
BOTHRESH	✓	✓		

Table 60. DEFINE and ALTER QUEUE parameters.

Cross-tabulation of queue parameters and queue types. If the parameter applies to the queue type, the cell contains a check mark.

(continued)

Parameter	Local queue	Model queue	Alias queue	Remote queue
z/OS z/OS **CFSTRUCT**	✓	✓		
CLCHNAME	✓			
CLUSNL	✓		✓	✓
CLUSTER	✓		✓	✓
CLWLPRTY	✓		✓	✓
CLWLRANK	✓		✓	✓
CLWLUSEQ	✓			
z/OS z/OS **CMDSCOPE**	✓	✓	✓	✓
CUSTOM	✓	✓	✓	✓
DEFBIND	✓		✓	✓
DEFPRESP	✓	✓	✓	✓
DEFPRTY	✓	✓	✓	✓
DEFPSIST	✓	✓	✓	✓
DEFREADA	✓	✓	✓	
DEFSOPT	✓	✓		
DEFTYPE	✓	✓		
DESCR	✓	✓	✓	✓
DISTL	✓	✓		
FORCE	✓		✓	✓
GET	✓	✓	✓	
HARDENBO or **NOHARDENBO**	✓	✓		
INDXTYPE	✓	✓		
INITQ	✓	✓		

Table 60. DEFINE and ALTER QUEUE parameters.

Cross-tabulation of queue parameters and queue types. If the parameter applies to the queue type, the cell contains a check mark.

(continued)

Parameter	Local queue	Model queue	Alias queue	Remote queue
LIKE	✓	✓	✓	✓
MAXDEPTH	✓	✓		
MAXMSGL	✓	✓		
MONQ	✓	✓		
MSGDLVSQ	✓	✓		
NPMCLASS	✓	✓		
PROCESS	✓	✓		
PROPCTL	✓	✓	✓	
PUT	✓	✓	✓	✓
queue-name	✓	✓	✓	✓
QDEPTHHI	✓	✓		
QDEPTHLO	✓	✓		
QDPHIEV	✓	✓		
QDPLOEV	✓	✓		
QDPMAXEV	✓	✓		
z/OS z/OS **QSGDISP**	✓	✓	✓	✓
QSVCIEV	✓	✓		
QSVCINT	✓	✓		
RETINTVL	✓	✓		
RNAME				✓
RQMNAME				✓
SCOPE	✓		✓	✓
SHARE or **NOSHARE**	✓	✓		
STATQ	✓	✓		

Table 60. DEFINE and ALTER QUEUE parameters.

Cross-tabulation of queue parameters and queue types. If the parameter applies to the queue type, the cell contains a check mark.

(continued)

Parameter	Local queue	Model queue	Alias queue	Remote queue
z/OS z/OS **STGCLASS**	✓	✓		
TARGET			✓	
TARGQ			✓	
TARGTYPE			✓	
TRIGDATA	✓	✓		
TRIGDPTH	✓	✓		
TRIGGER or **NOTRIGGER**	✓	✓		
TRIGMPRI	✓	✓		
TRIGTYPE	✓	✓		
USAGE	✓	✓		
XMITQ				✓

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 MQCONNX 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 switch on 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.

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.

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 nonblank; you cannot specify a value for both.

On local queues, this parameter cannot be set for transmission, SYSTEM.CHANNEL. *xx* , SYSTEM.CLUSTER. *xx* , or SYSTEM.COMMAND. *xx* queues z/OS, and on z/OS only, for SYSTEM.QSG. *xx* queues.

This parameter is valid only on AIX, HP-UX, Linux, Solaris, and Windows z/OS, and z/OS.

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 nonblank; you cannot specify a value for both.

On local queues, this parameter cannot be set for transmission, SYSTEM.CHANNEL. *xx* , SYSTEM.CLUSTER. *xx* , or SYSTEM.COMMAND. *xx* queues z/OS, and on z/OS only, for SYSTEM.QSG. *xx* queues.

This parameter is valid only on AIX, HP-UX, Linux, Solaris, and Windows z/OS, and 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 is run on the queue manager on which it was entered.

QmgrName
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. You can specify another name, 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 * is the same as entering the command on every queue manager in the queue-sharing group.

CUSTOM(*string*)
The custom attribute for new features.

V 8.0.0.4 This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name and value pairs have the form NAME(VALUE). Single quotation marks must be escaped with another single quotation mark.

CAPEXPRY(*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.

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 CAPEXPRY 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 **CAPEXPRY**, 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 **CAPEXPRY** have the new expiry time.

The maximum length is defined by the constant MQ_CUSTOM_LENGTH and is currently set to 128 on all platforms.

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.

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, MQOO_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 only on AIX, HP-UX, Linux, IBM i, Solaris, and Windows z/OS, and z/OS.

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.

z/OS 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 in 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

SHARED
The open request is for shared input from the queue

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.

Do not specify this value for a model queue definition with a **DEFPSIST** parameter of YES.

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.

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, HP-UX, Linux, Solaris, 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 are true:

- The **TARGET** parameter specifies a queue
- An application has this alias queue open

For a local queue, if both of the following 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 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 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 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 and 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 manger.

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: z/OS This parameter affects only IBM MQ for 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 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:

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 INDXTYPE 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	-	-

Queue type		NON-SHARED			SHARED	
MSGTOKEN	MSGID	No	Yes	Yes	-	-
MSGTOKEN	CORRELID	No	Yes	Yes	-	-
MSGTOKEN	GROUPID	No	No	Yes	-	-
GROUPID	NONE	No	No	Yes	No	Yes
GROUPID	MSGID	No	No	Yes	No	Yes
GROUPID	CORRELID	No	No	Yes	No	Yes
GROUPID	MSGTOKEN	No	No	Yes	-	-

This parameter is supported only on z/OS. On other platforms, all queues are automatically indexed.

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:

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.

MAXDEPTH(*integer*)
The maximum number of messages allowed on the queue.

This parameter is supported only on local and model queues.

On AIX, HP-UX, Linux, Solaris, and Windows z/OS, and z/OS , specify a value in the range zero through 999999999.

This parameter is valid only on AIX, HP-UX, Linux, Solaris, and Windows z/OS, and 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.

MAXMSGL(*integer*)
The maximum length (in bytes) of messages on this queue.

This parameter is supported only on local and model queues.

On AIX, HP-UX, Linux, Solaris, 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.

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, IBM MQ 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: z/OS 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.

z/OS 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 AIX, HP-UX, IBM i, Linux, Solaris, and Windows z/OS, and 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. z/OS For more information about the effect that shared queues on z/OS have on this event; see Shared queues and queue depth events (IBM MQ for 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. z/OS For more information about the effect that shared queues on z/OS have on this event; see Shared queues and queue depth events (IBM MQ for 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.

Note: The value of this parameter can change implicitly z/OS, and shared queues on z/OS affect the event. See the description of Queue Depth High.

ENABLED
Queue Depth High events are generated

DISABLED
Queue Depth High events are not generated

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.

Note: The value of this parameter can change implicitly z/OS, and shared queues on z/OS affect the event. For more information about this event, see Queue Depth Low.

ENABLED
Queue Depth Low events are generated

DISABLED
Queue Depth Low events are not generated

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.

Note: The value of this parameter can change implicitly z/OS, and shared queues on z/OS affect the event. For more information about this event, see Queue Full.

ENABLED
Queue Full events are generated

DISABLED
Queue Full events are not generated

z/OS **QSGDISP**
This parameter applies to z/OS only.

Specifies the disposition of the object within the group.

Action of ALTER depending on different values of **QSGDISP**.

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 **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), or any object defined using a command that had the parameters **QSGDISP** (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:
	`DEFINE QUEUE(QNAME)`
	`REPLACE QSGDISP(COPY)`
	The ALTER for the group object takes effect regardless of whether the generated command with **QSGDISP** (COPY) fails.

PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with **QSGDISP** (QMGR) or **QSGDISP** (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 **QSGDISP** (QMGR). Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.
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 **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. 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.

QSVCIEV
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 **QSVCINT** 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 **QSVCINT** 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

QSVCINT(*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 **QSVCIEV** 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.

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 UNIX and Linux systems, and Windows.

SHAREand 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.

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.

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.

This parameter is valid only on IBM i, UNIX and Linux systems, and Windows.

z/OS **STGCLASS(*string*)**
The name of the storage class.

This parameter is supported only on local and model queues.

This parameter is an installation-defined name.

This parameter is valid on z/OS only z/OS; see Storage classes .

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.

Note: You can change this parameter only if the queue is empty and closed.

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 **TARGTYPE**. 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**.

TARGTYPE(*string*)
The type of object to which the alias resolves.

QUEUE
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 on AIX, HP-UX, IBM i, Linux, Solaris, and Windows z/OS, and z/OS, 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.

This parameter is supported only on local and model queues.

This parameter can also be changed using the MQSET API call.

TRIGGER and 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 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" on page 1122 for details.

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.

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.

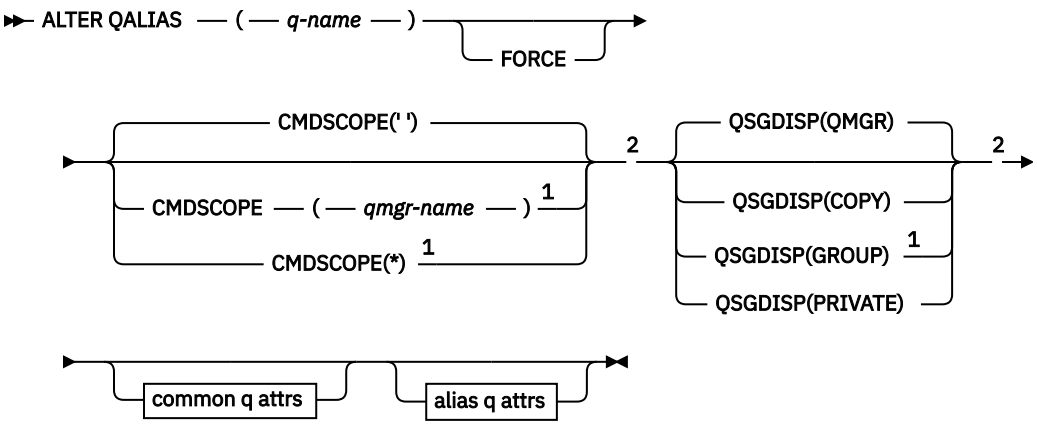
This parameter is ignored if the definition is being used as a queue manager alias and **RQMNAME** is the name of the local queue manager.

It is also ignored if the definition is used as a reply-to queue alias definition.

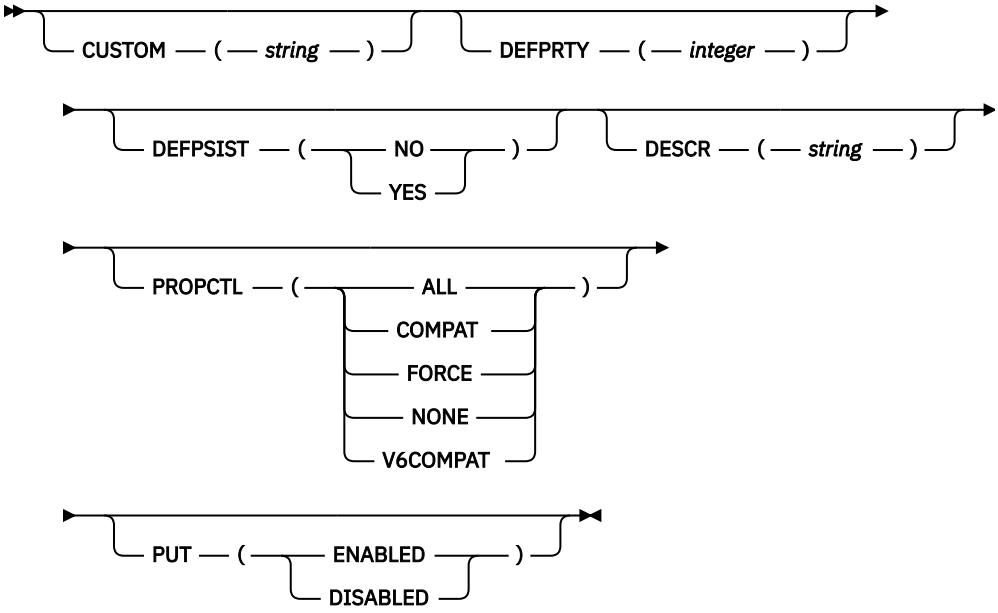
ALTER QALIAS
Use the MQSC command ALTER QALIAS to alter the parameters of an alias queue.

Synonym: ALT QA

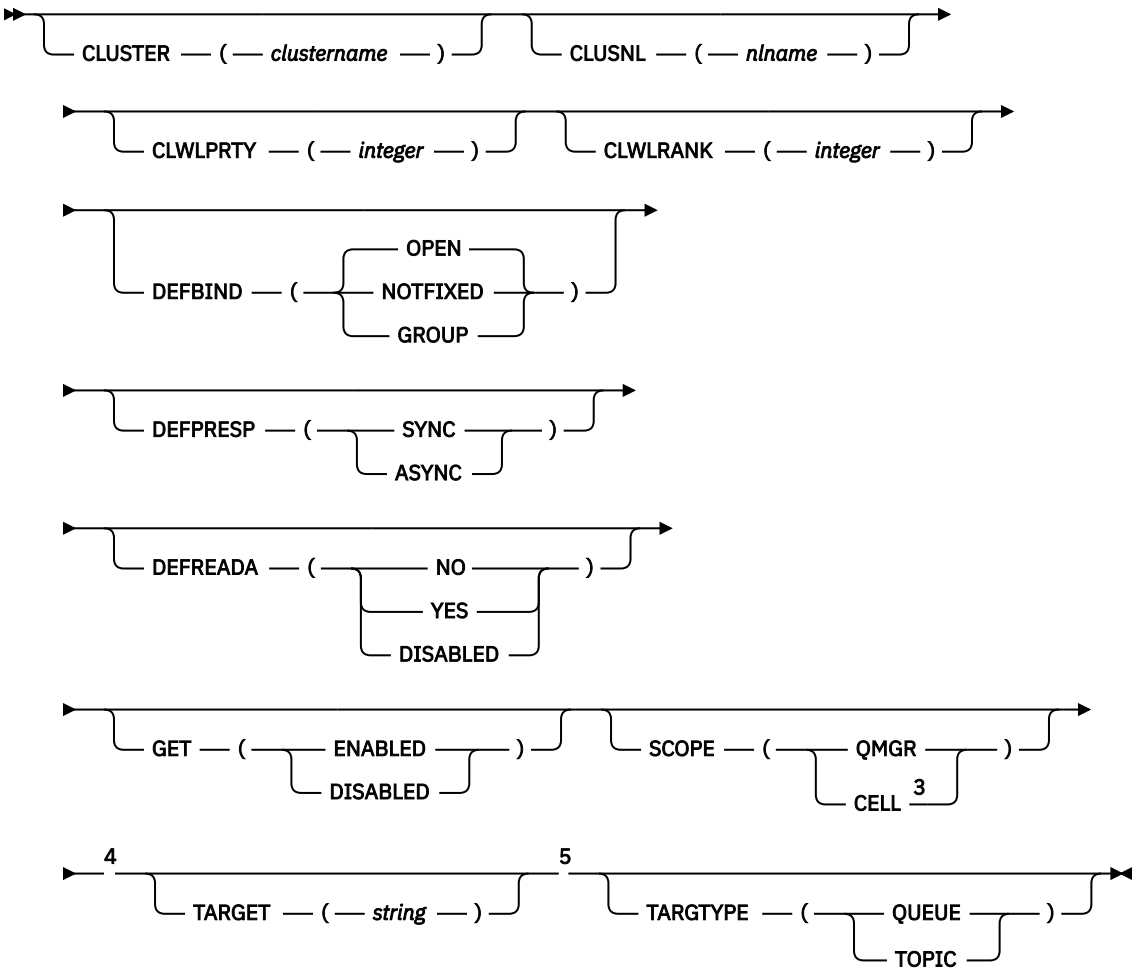
ALTER QALIAS



Common q attrs



Alias q attrs



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] Valid only on Windows, UNIX, and Linux systems.
[4] Not valid on z/OS.
[5] 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.

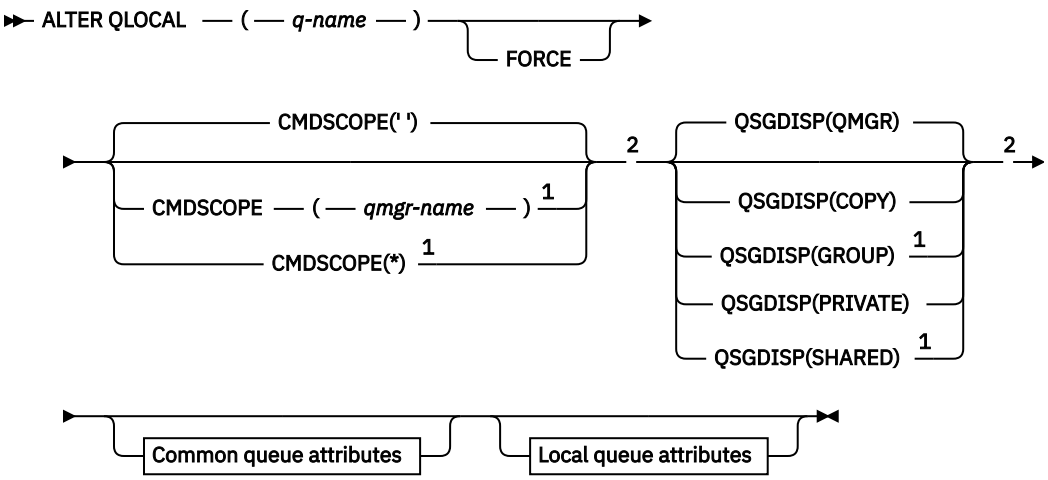
The parameters are described in "ALTER queues" on page 756.

ALTER QLOCAL

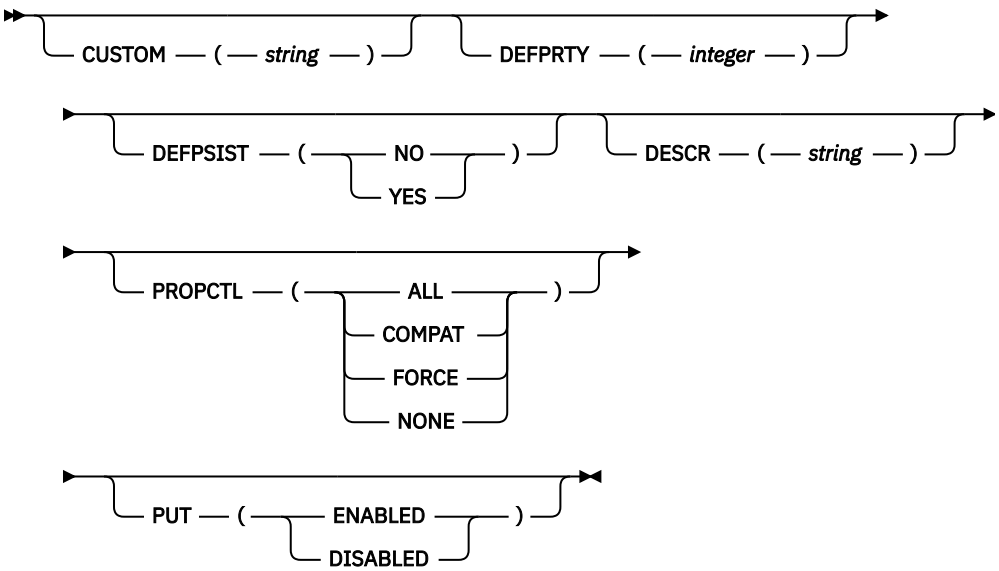
Use the MQSC command **ALTER QLOCAL** to alter the parameters of a local queue.

Synonym: ALT QL

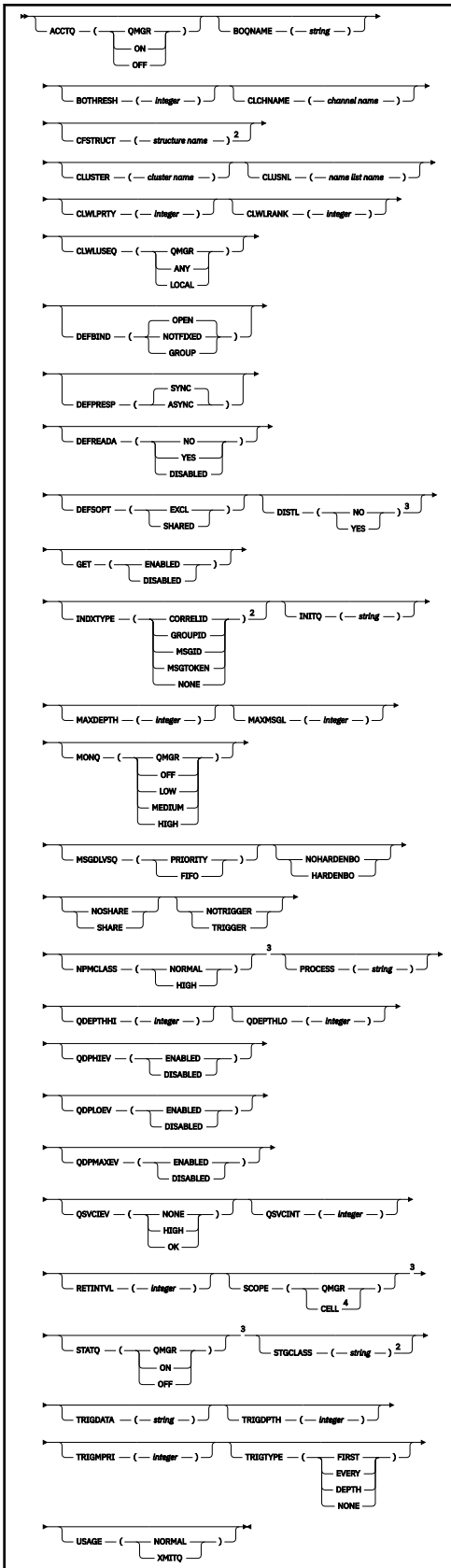
ALTER QLOCAL



Common queue attributes



Local queue 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] Valid on UNIX, Linux, and Windows systems.

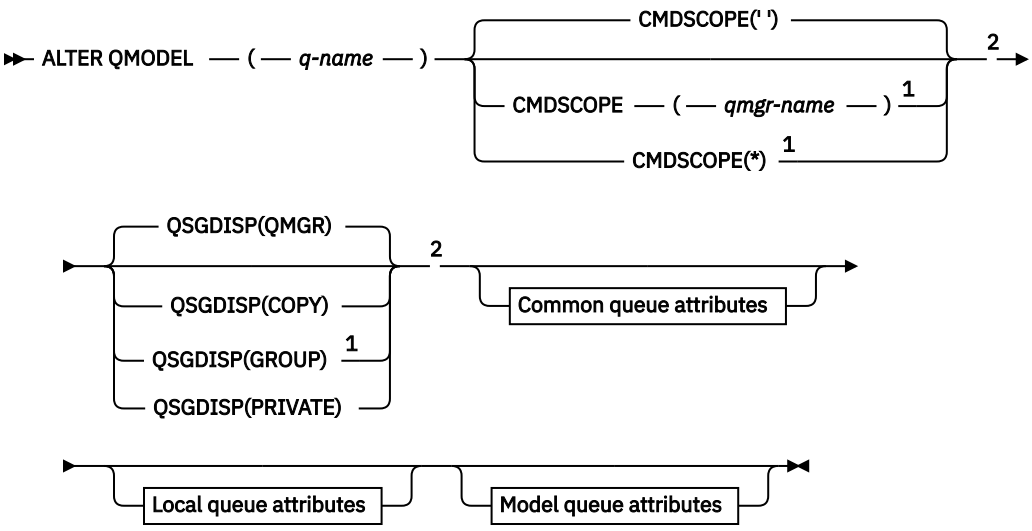
The parameters are described in "ALTER queues" on page 756.

ALTER QMODEL

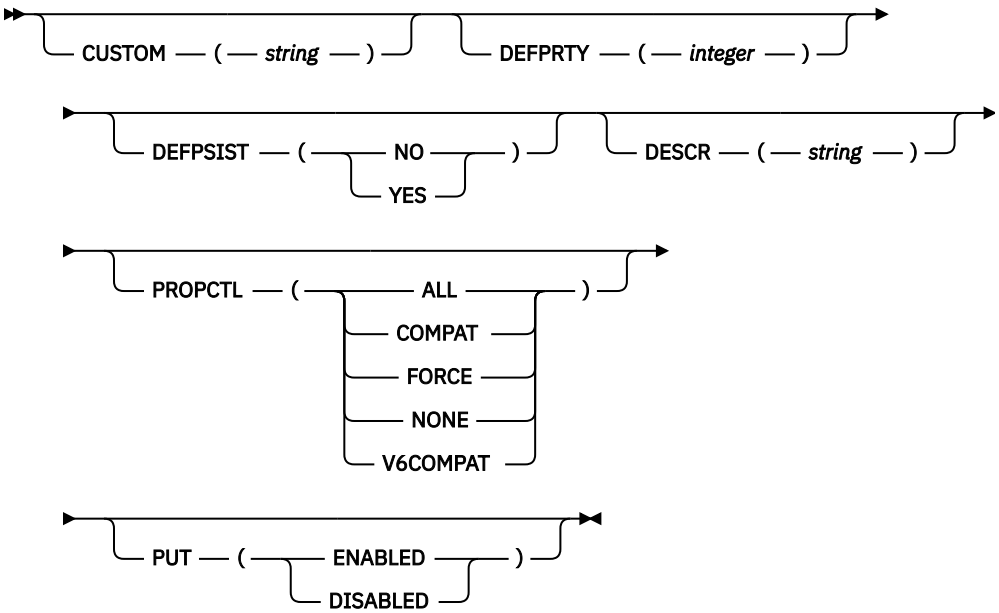
Use the MQSC command ALTER QMODEL to alter the parameters of a model queue.

Synonym: ALT QM

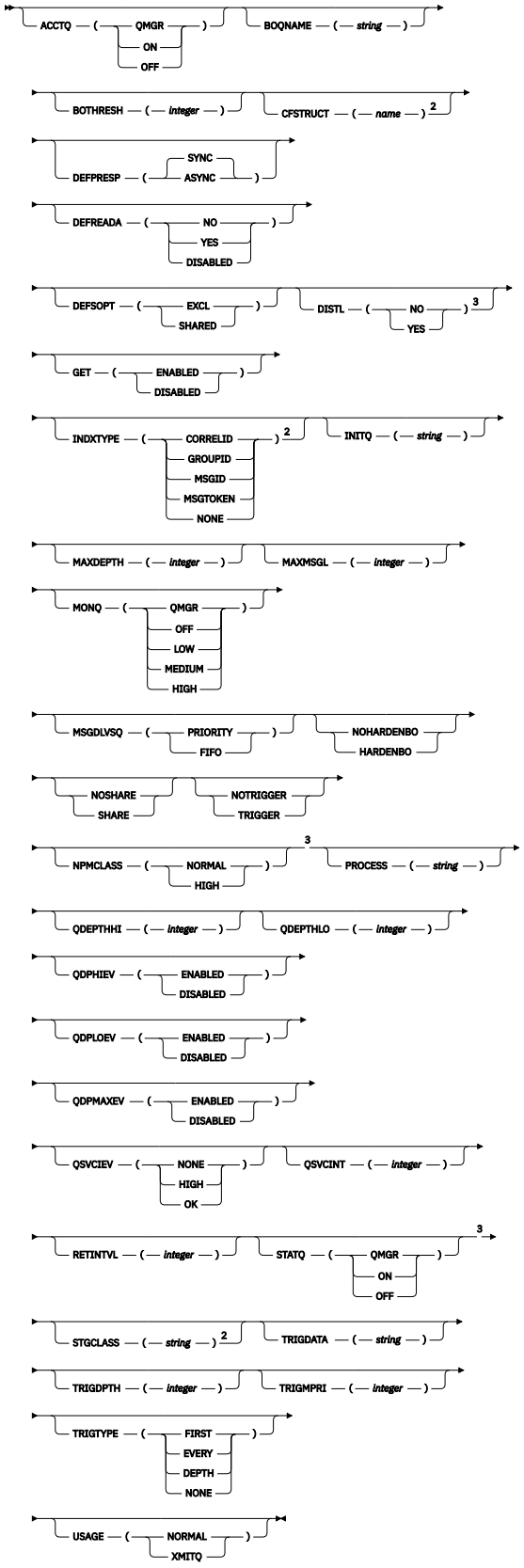
ALTER QMODEL



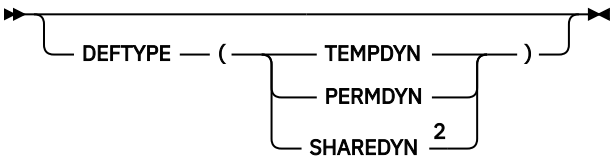
Common queue attributes



Local queue attributes



Model queue 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.

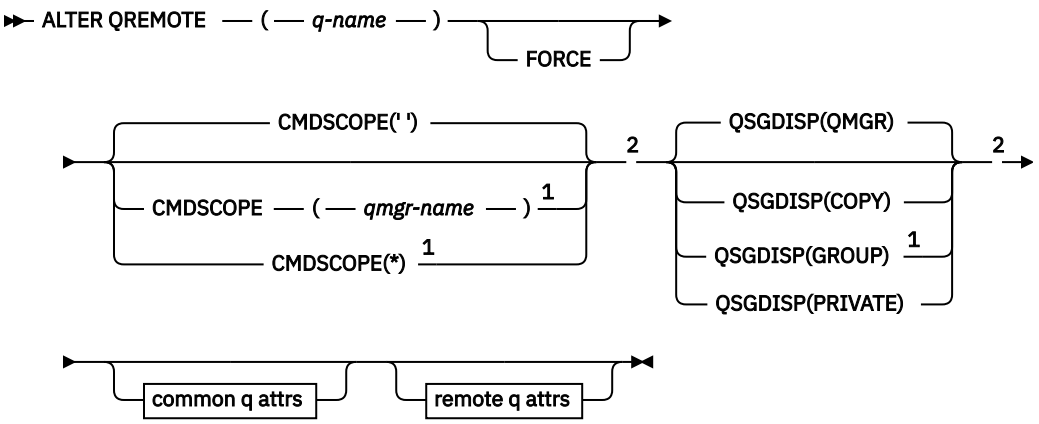
The parameters are described in "ALTER queues" on page 756.

ALTER QREMOTE

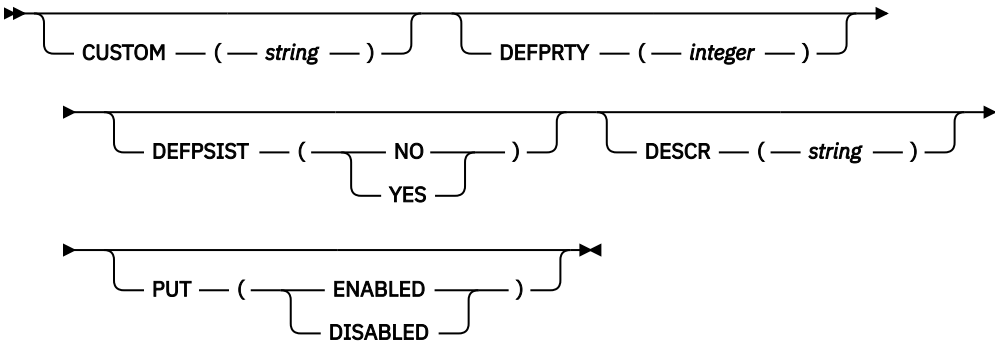
Use the MQSC command ALTER QREMOTE to alter the parameters of a local definition of a remote queue, a queue-manager alias, or a reply-to queue alias.

Synonym: ALT QR

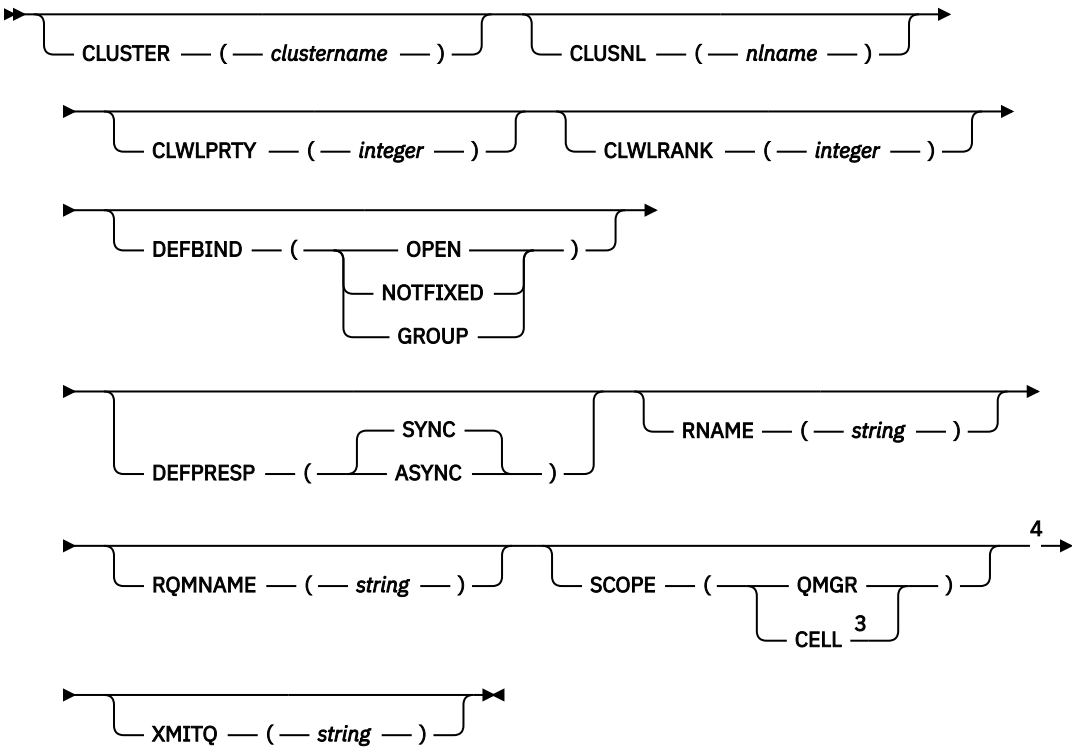
ALTER QREMOTE



Common q attrs



Remote q attrs



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] Valid only on Windows, UNIX, and Linux systems.
[4] Not valid on z/OS.

The parameters are described in "ALTER queues" on page 756.

z/OS ALTER SECURITY

Use the MQSC command ALTER SECURITY to define system-wide security options.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			12CR

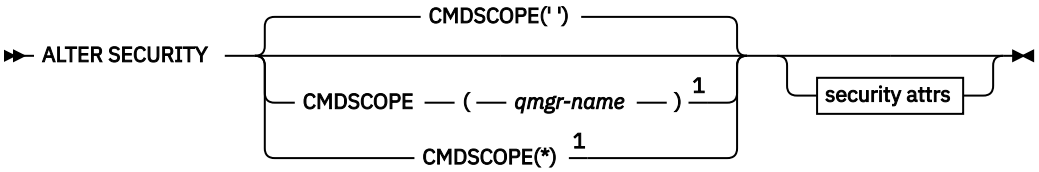
Parameters not specified in the ALTER SECURITY command result in the existing values for those parameters being left unchanged.

z/OS For an explanation of the symbols in the z/OS column, see "Using commands on z/OS" on page 629.

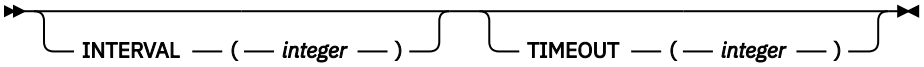
- Syntax diagram
- "Parameter descriptions for ALTER SECURITY" on page 786

Synonym: ALT SEC

ALTER SECURITY



Security attrs



Notes:

1 Valid only on z/OS when the queue manager is a member of a queue-sharing group.

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 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 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 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 * 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

ALTER SERVICE

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 Performing local administration tasks using MQSC commands.

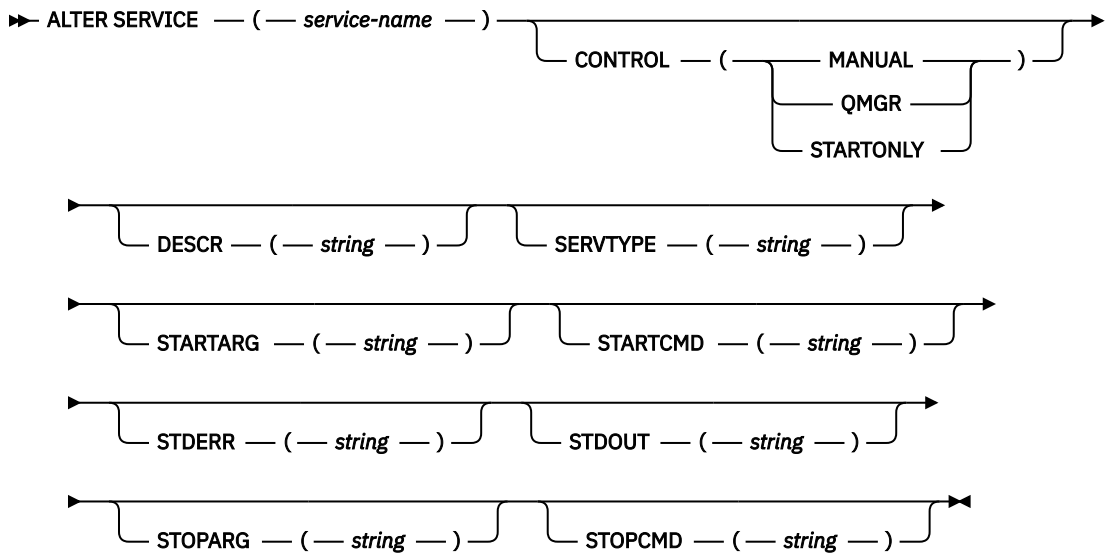
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

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 787

Synonym:

ALTER SERVICE



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" on page 1170).

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

z/OS ALTER SMDS

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

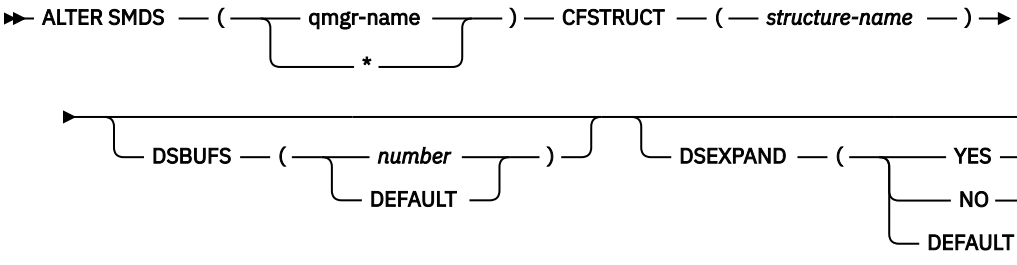
Parameters not specified in the ALTER SMDS command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for ALTER SMDS” on page 790

Synonym:

ALTER SMDS



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

Use the MQSC command ALTER STGCLASS to alter the characteristics of a storage class.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

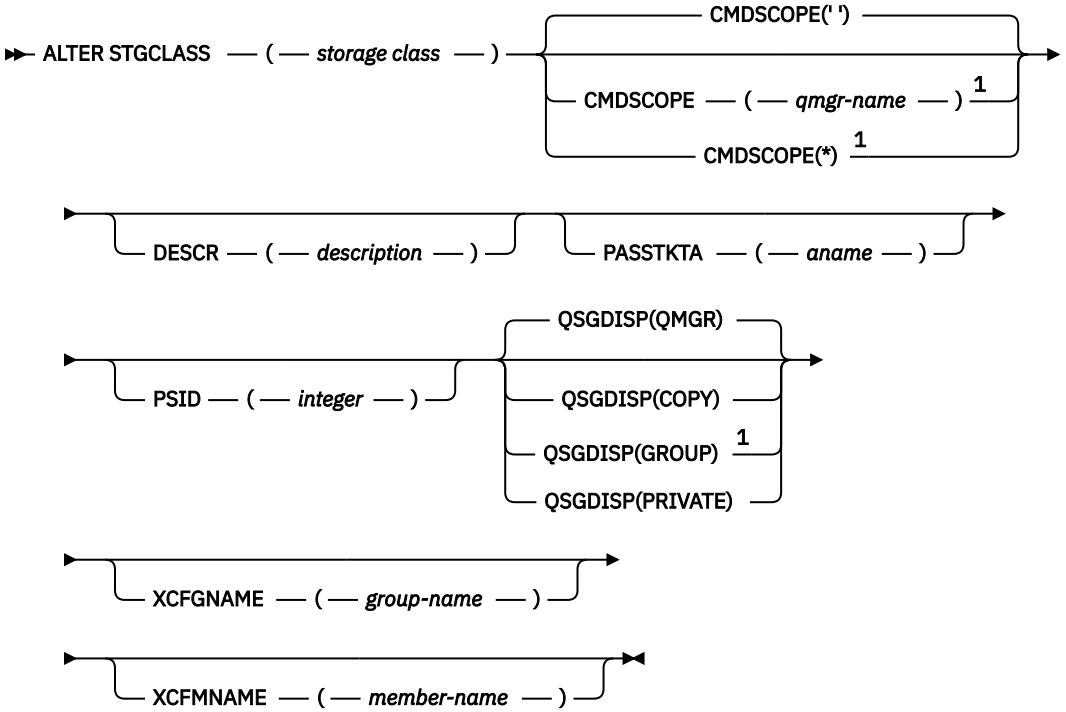
Parameters not specified in the ALTER STGCLASS command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for ALTER STGCLASS” on page 791

Synonym: ALT STC

ALTER STGCLASS



Notes:

1 Valid only on z/OS when the queue manager is a member of a queue-sharing group.

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 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.

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” on page 911.

QSGDISP
Specifies the disposition of the object in the group.

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 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 attempt to refresh local copies on page set zero:
	`DEFINE STGCLASS(storage-class)`
	`REPLACE QSGDISP(COPY)`
	The ALTER for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with QSGDISP(QMGR) or QSGDISP(COPY). Any object residing in the shared repository is unaffected.

QSGDISP	ALTER
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.

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

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

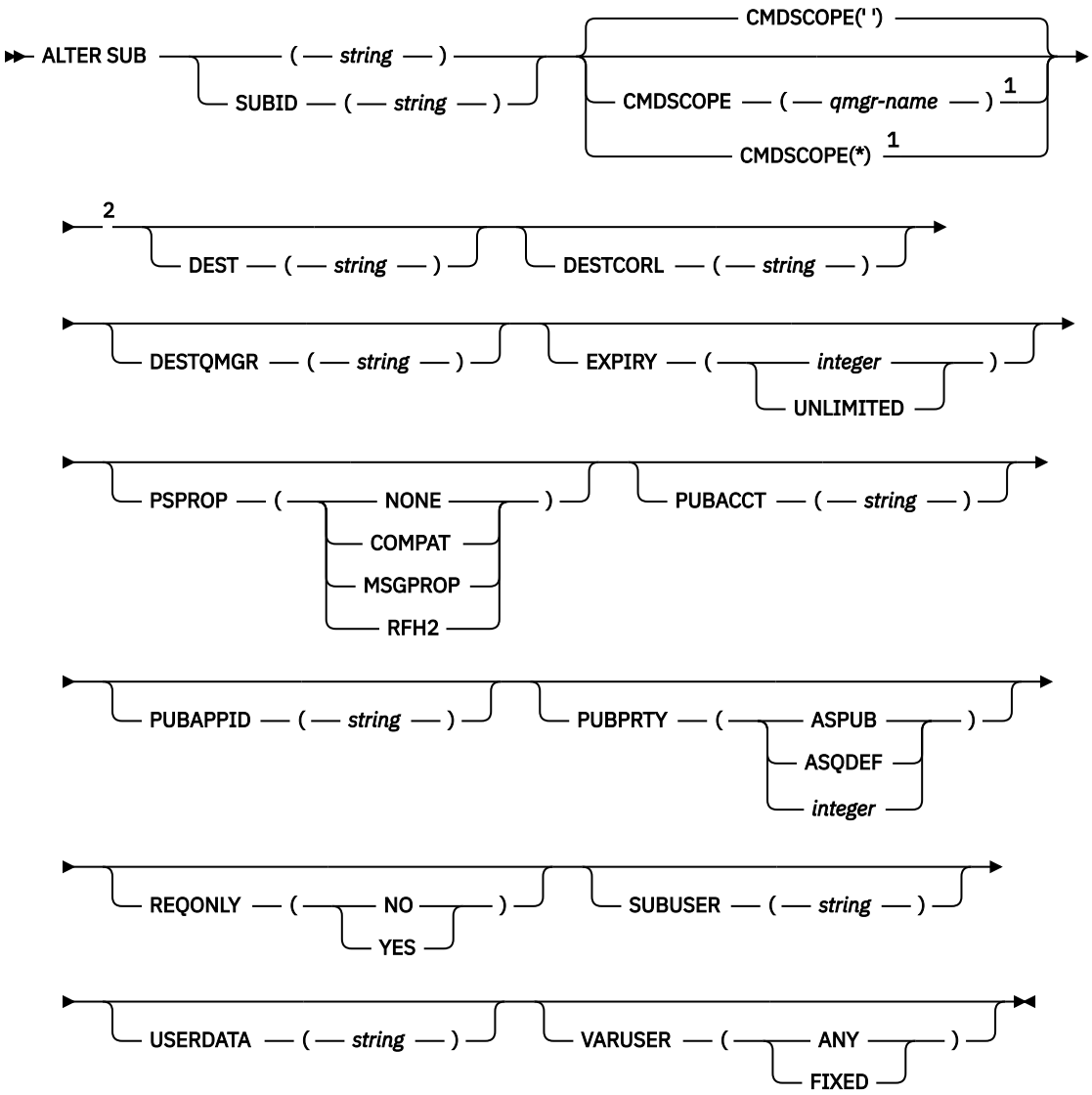
Parameters not specified in the ALTER SUB command result in the existing values for those parameters being left unchanged.

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for ALTER SUB” on page 794
- “Parameter descriptions for ALTER SUB” on page 795

Synonym: ALT SUB

ALTER SUB



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.

Usage notes for ALTER SUB

1. The following are valid forms of the command:

```
ALT SUB(xyz)
ALT SUB SUBID(123)
ALT SUB(xyz) SUBID(123)
```

2. Although permitted on the DEFINE command, you cannot alter the following fields using DEFINE SUB (REPLACE):
   - TOPICOBJ
   - TOPICSTR
   - WSCHEMA

- SELECTOR
- SUBSCOPE
- DESTCLAS

3. At the time the ALT 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.
4. V 8.0.0.6 Subscriptions with a SUBTYPE of PROXY cannot be modified. Attempts to modify a proxy subscription by using the PCF interface return MQRCCF_SUBSCRIPTION_IN_USE. MQSC reports the following message:

   ```
   AMQ8469: WebSphere MQ subscription SYSTEM.PROXY in use.
   ```

Parameter descriptions for ALTER SUB

(string)
A mandatory parameter. Specifies the unique name for this subscription, see **SUBNAME** property.

z/OS **CMDSCOPE**
This parameter applies to z/OS only and specifies how the command is processed when the queue manager is a member of a queue-sharing group.

' '
The command is processed on the queue manager on which it was entered. This is the default value.

qmgr-name
The command is processed 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 processed 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.

DEST(*string*)
The destination for messages published to this subscription; this parameter is the name of a queue.

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 ' `00000000000000000000000000000000000000000000000000000000` ' (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.

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. You must define the channels to the remote queue manager, for example, the XMITQ, and a sender channel. If you do not, messages do not arrive at the destination.

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.

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.

SUBLEVEL(*integer*)
The level within the subscription hierarchy at which this subscription is made. The range is zero through 9.

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.

V 8.0.0.7 From IBM MQ 8.0.0, Fix Pack 7, 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` in the JmsConstants interface with the method `javax.jms.Message.getStringProperty(java.lang.String)`. 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.

ALTER TOPIC

Use ALTER TOPIC to alter the parameters of an existing IBM MQ topic object.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

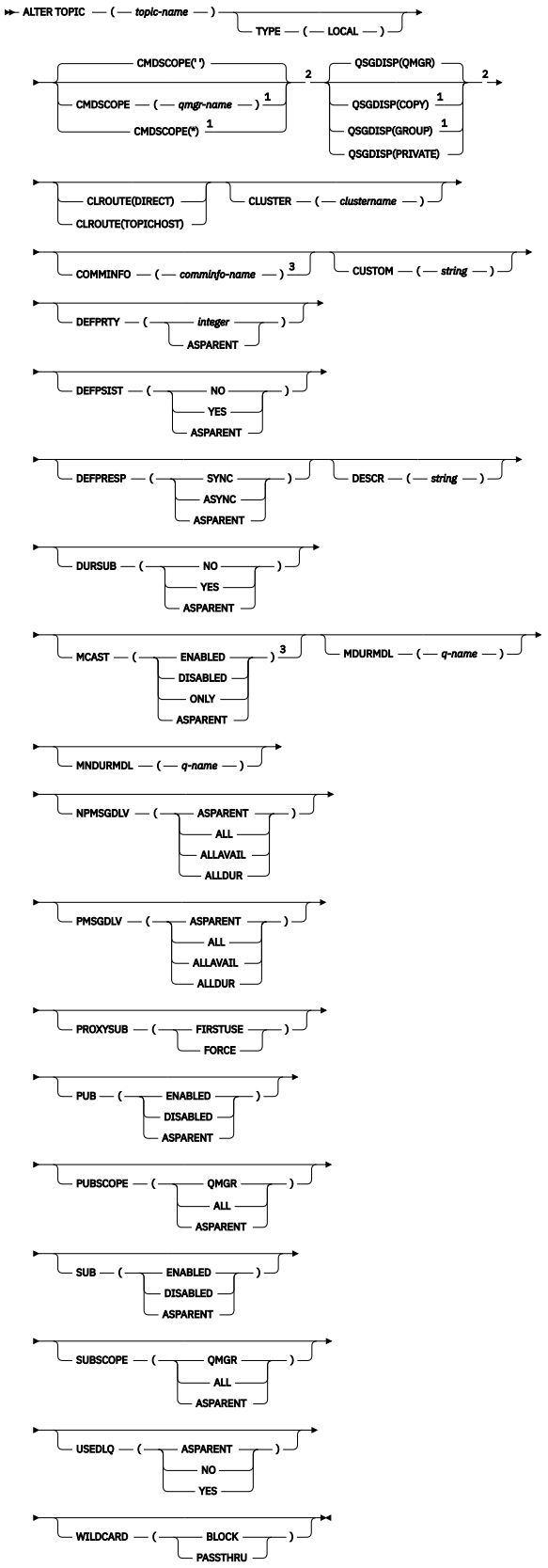
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

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 799
- “Parameter descriptions for ALTER TOPIC” on page 799

Synonym: ALT TOPIC

ALTER TOPIC



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.

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).

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 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.

COMMINFO(*comminfo-name*)
The name of the communication information object associated with this topic object.

CUSTOM (*string*)
The custom attribute for new features.

V 8.0.0.4 This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name and value pairs have the form `NAME(VALUE)`. Single quotation marks must be escaped with another single quotation mark.

CAPEXPRY (*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.

Specifying a value for CAPEXPRY 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.

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. 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.

z/OS **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object within the group.

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 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 attempt to refresh local copies on page set zero:
	`DEFINE TOPIC(name)`
	`REPLACE QSGDISP(COPY)`
	The ALTER for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with QSGDISP(QMGR) or QSGDISP(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 QSGDISP(QMGR). 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 NPMSGDLV 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.

z/OS ALTER TRACE

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			12CR

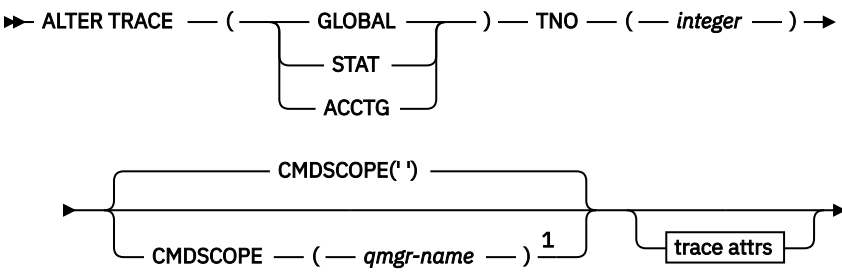
Parameters not specified in the ALTER TRACE command result in the existing values for those parameters being left unchanged.

z/OS For an explanation of the symbols in the z/OS column, see "Using commands on z/OS" on page 629.

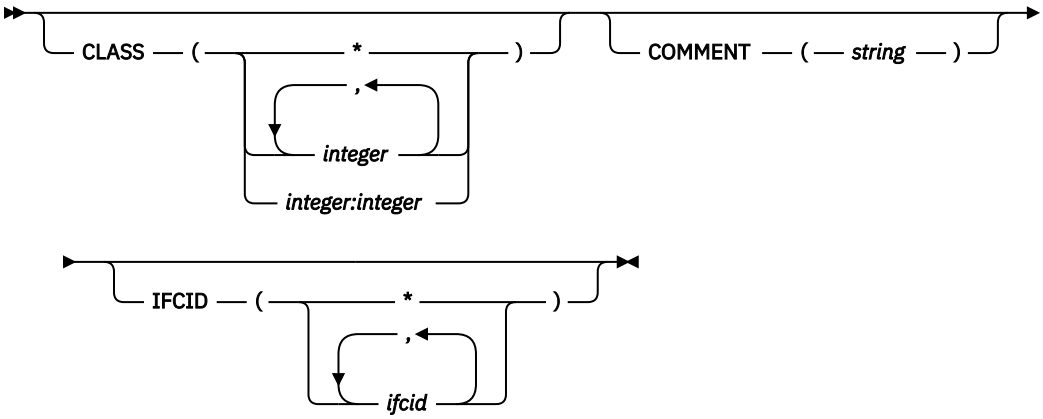
- Syntax diagram
- "Usage notes" on page 807
- "Parameter descriptions for ALTER TRACE" on page 807
- "Trace parameters" on page 807

Synonym: ALT TRACE

ALTER TRACE



Trace attrs



Notes:

1 Valid only on z/OS when the queue manager is a member of a queue-sharing group.

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 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 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 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" on page 1301 for a list of allowed classes. A range of classes can be specified as *m:n* (for example, CLASS(01:03)). CLASS(*) activates all classes.

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.

z/OS ARCHIVE LOG

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

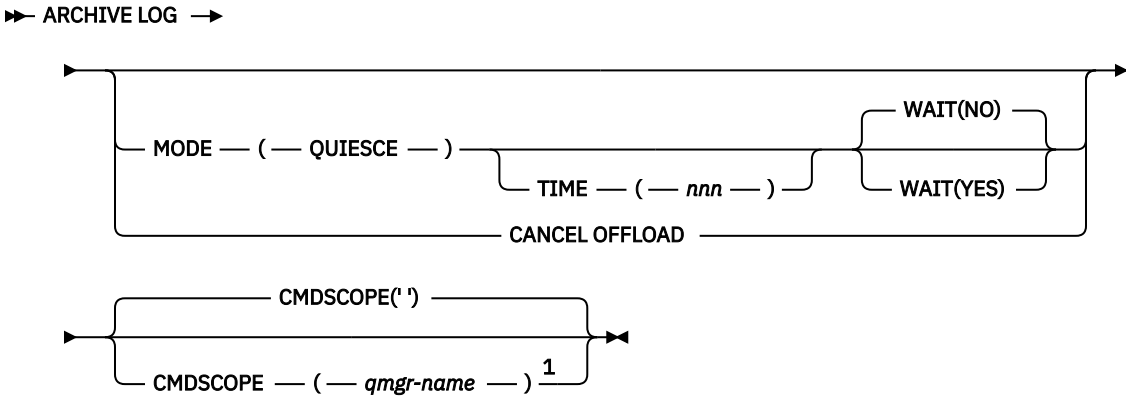
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for ARCHIVE LOG" on page 808
- "Parameter descriptions for ARCHIVE LOG" on page 809

Synonym: ARC LOG

ARCHIVE LOG



Notes:

1 Valid only on z/OS when the queue manager is a member of a queue-sharing group.

Usage notes for ARCHIVE LOG

In detail, ARCHIVE LOG:

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 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 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.

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.

z/OS BACKUP CFSTRUCT

Use the MQSC command BACKUP CFSTRUCT to initiate a CF application structure backup.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

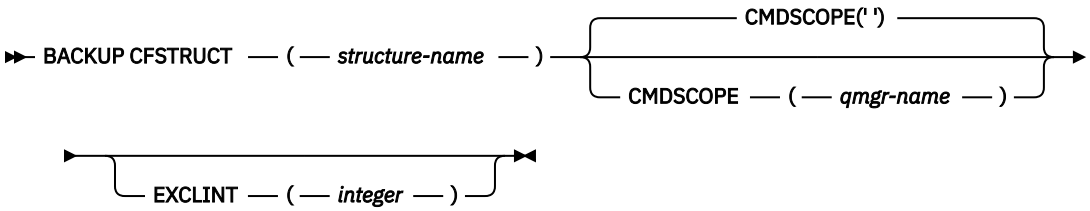
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

IBM i	UNIX and Linux	Windows	z/OS
			CR

- Syntax diagram
- "Usage notes for BACKUP CFSTRUCT" on page 810
- "Keyword and parameter descriptions for BACKUP CFSTRUCT" on page 811

Synonym: None

BACKUP CFSTRUCT



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 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 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

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 Performing local administration tasks using MQSC commands.

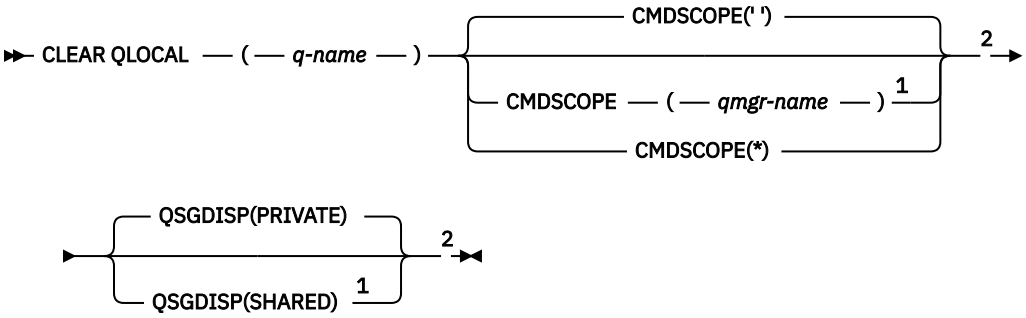
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for CLEAR QLOCAL” on page 812

Synonym: CLEAR QL

CLEAR QLOCAL



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.

Parameter descriptions for CLEAR QLOCAL

You must specify which local queue you want to clear.

The command fails if either:

- The queue has uncommitted messages that have been put on the queue under syncpoint
- The queue is currently open by an application (with any open options)

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 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 SHARED.

' '
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 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).

CLEAR TOPICSTR

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 Performing local administration tasks using MQSC commands.

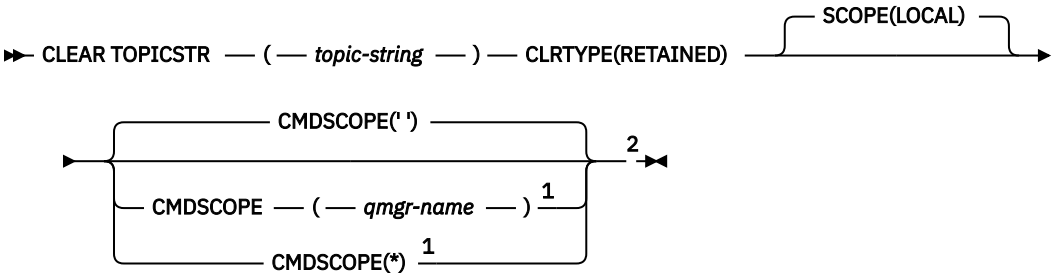
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

z/OS For an explanation of the symbols in the z/OS column, see "Using commands on z/OS" on page 629.

- Syntax diagram
- Usage notes for CLEAR TOPICSTR
- Parameter descriptions for CLEAR TOPICSTR

Synonym: None.

CLEAR TOPICSTR



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.

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.

z/OS **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 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 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.

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

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 enable user ID and password checking.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

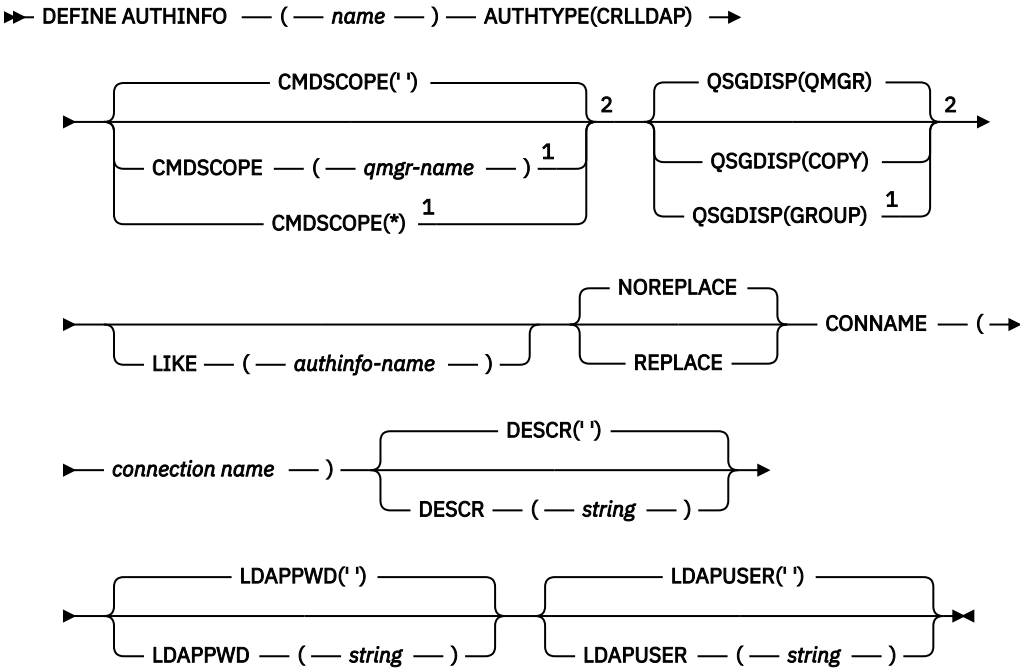
- "Usage notes for DEFINE AUTHINFO" on page 819
- "Parameter descriptions for DEFINE AUTHINFO" on page 819
- Syntax diagram for TYPE(CRLLDAP)
- Syntax diagram for TYPE(OCSP)
- Syntax diagram for TYPE(IDPWOS)
- Syntax diagram for TYPE(IDPWLDAP)

Synonym: DEF AUTHINFO

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

Syntax diagram for TYPE(CRLLDAP)

DEFINE AUTHINFO

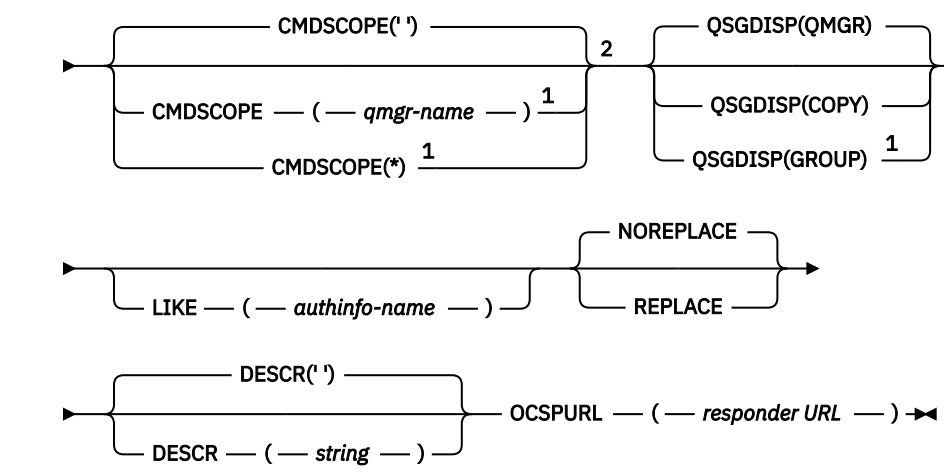


Notes:

[1] 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.
[2] Valid only on z/OS.

Syntax diagram for TYPE(OCSP)

DEFINE AUTHINFO



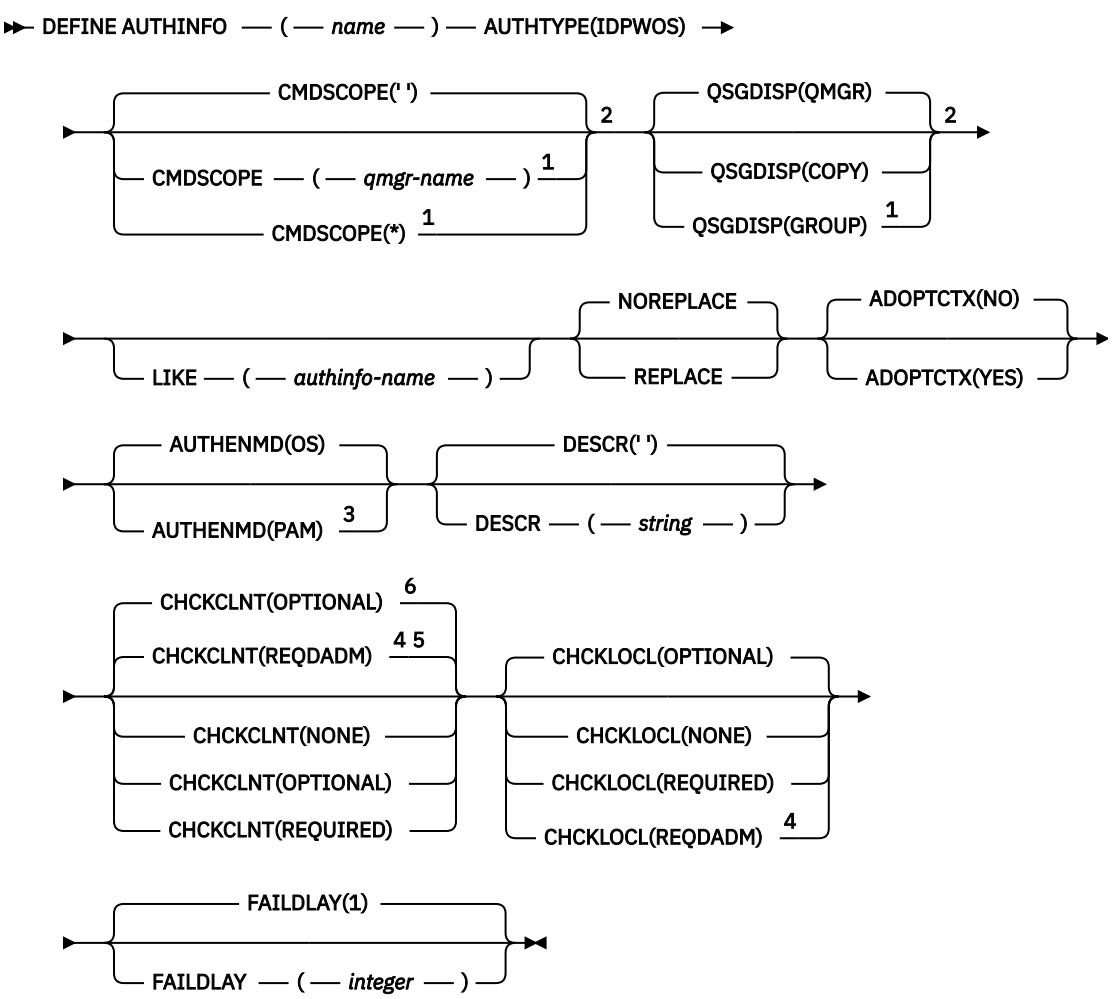
Notes:

[1] 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.

[2] Valid only on z/OS.

Syntax diagram for TYPE(IDPWOS)

DEFINE AUTHINFO

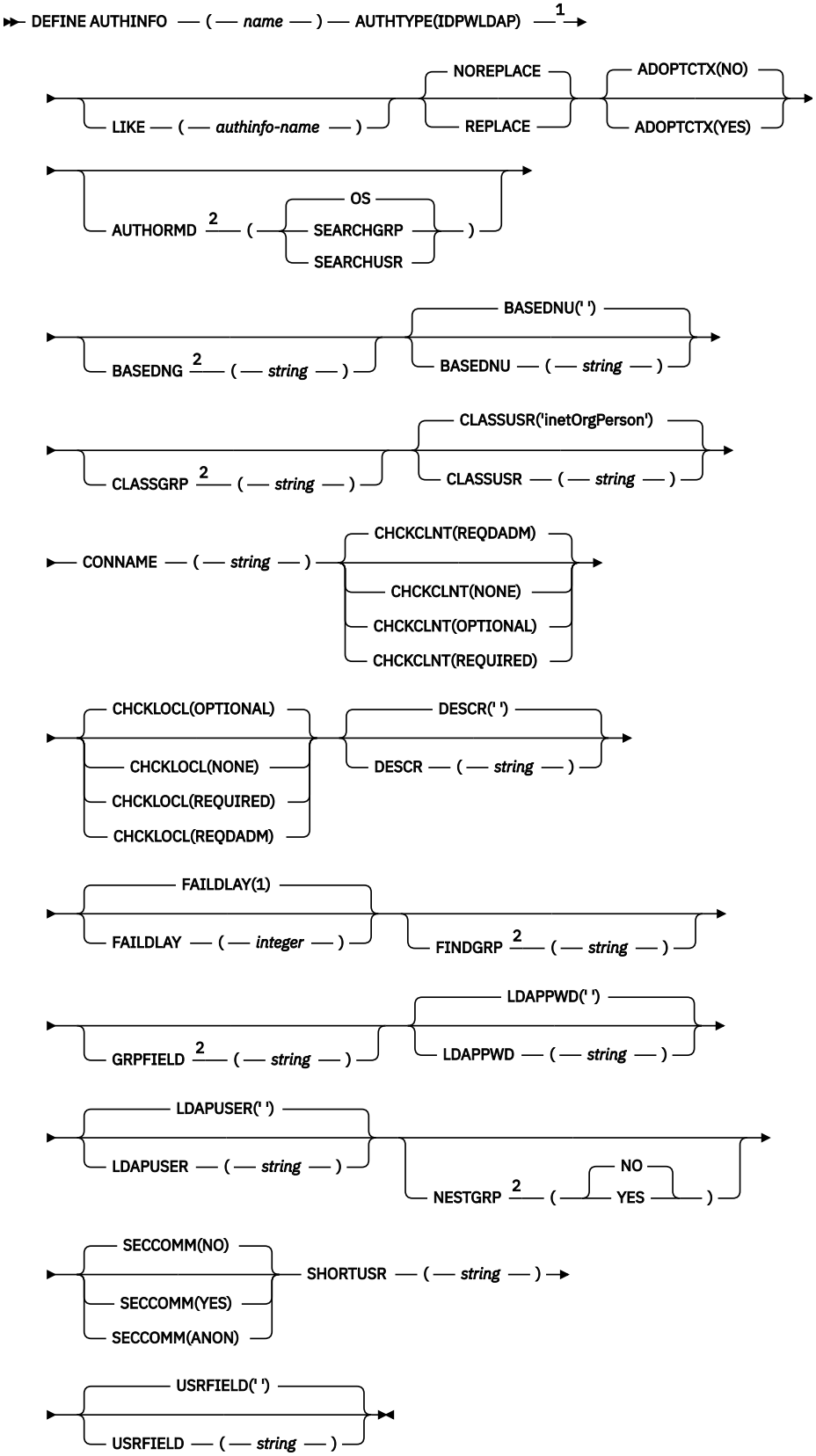


Notes:

[1] 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.
[2] Valid only on z/OS.
[3] Not valid on z/OS and PAM value can be set only on UNIX platforms.
[4] Not valid on IBM MQ for z/OS.
[5] Default for platforms other than z/OS.
[6] Default for z/OS.

Syntax diagram for TYPE(IDPWLDAP)

DEFINE AUTHINFO



Notes:

[1] Not valid on IBM MQ for z/OS.
[2] Not valid on Windows.

Usage notes for DEFINE AUTHINFO

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 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.

V 8.0.0.3 **AUTHENMD**

Authentication method. Whether to use the operating system or Pluggable Authentication Method (PAM) to authenticate user passwords.

OS

Use the traditional UNIX platforms password verification method.

PAM

Use the PAM to authenticate the user password.

You can set the PAM value only on UNIX and Linux platforms.

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*.

V 8.0.0.2 **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 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 all the groups to which the specified user belongs. The attribute to query is defined by the FINDGRP value, typically *memberOf*.

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* V 8.0.0.2, and is not applicable to Windows platforms.

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.

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.

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.

V 8.0.0.2 **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.

This attribute is valid only for an **AUTHTYPE** of *IDPWLDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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
No user ID and password checks are made. If any user ID or password is supplied by a client application, the credentials are ignored.

OPTIONAL
Client applications are not required to provide a user ID and password.

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 will only be allowed to continue if the user ID and password are valid.

This option might be useful during migration, for example.

REQUIRED
All client applications must provide a user ID and password in the MQCSP structure. This user ID and password is 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.

REQDADM
All client applications using a privileged user ID must provide a user ID and password in the MQCSP structure. Any locally bound applications using a non-privileged user ID are not required to provide a user ID and password and are treated as with the OPTIONAL setting.

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.

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.

A privileged user is one that has full administrative authorities for IBM MQ. See Privileged users for more information.

z/OS This setting is not valid 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 message AMQ9793 on platforms other than z/OS, and message 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* . The possible values are:

NONE
No user ID and password checks are made. If any user ID or password is supplied by a locally bound application, the credentials are ignored.

OPTIONAL
Locally bound applications are not required to provide a user ID and password.

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 will only be allowed to continue if the user ID and password are valid.

This option might be useful during migration, for example.

REQUIRED
All locally bound applications must provide a user ID and password in the MQCSP structure. This 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.

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 a user ID and password in the MQCSP structure. Any locally bound applications using a non-privileged user ID are not required to provide a user ID and password and are treated as with the `OPTIONAL` setting.

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.

A privileged user is one that has full administrative authorities for IBM MQ. See Privileged users for more information.

z/OS This setting is not allowed on z/OS systems.

V 8.0.0.2 **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 an **AUTHTYPE** of *IDPWLDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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 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.

CONNAME(*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.

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 is 264 characters on IBM i, UNIX systems, and Windows, and 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” on page 993).

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 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* .

V 8.0.0.2 **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 the **FINDGRP** attribute is left blank:

- If AUTHORMD =*SEARCHGRP*, the **FINDGRP** attribute defaults to *memberOf*.

- If AUTHORMD =*SEARCHUSR*, the **FINDGRP** attribute defaults to *member*.

V 8.0.0.2 **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.

This attribute is valid only for an **AUTHTYPE** of *IDPWLDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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* .

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 1024 characters on IBM i, UNIX systems, and Windows, and 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.

On IBM i, UNIX systems, and Windows, 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.

V 8.0.0.2 **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.

This attribute is valid only for an **AUTHTYPE** of *IDPWLDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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).

QSGDISP	DEFINE
COPY	The object is defined on the page set of the queue manager that executes the command using the QSGDISP(GROUP) object of the same name as the 'LIKE' object.

GROUP	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 and sent to all active queue managers in the queue-sharing group to make or refresh local copies on page set zero:
	`DEFINE AUTHINFO(name)`
	`REPLACE QSGDISP(COPY)`
	The DEFINE for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

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 SSL/TLS

YES

Connectivity to the LDAP server is made securely using SSL/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 SSL/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 SSL/TLS.

This attribute is valid only for an **AUTHTYPE** of *IDPWLDAP*.

SHORTUSR(*LDAP field 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.

This 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 ' = ' 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

Use the MQSC command DEFINE BUFFPOOL to define a buffer pool that is used for holding messages in main storage.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

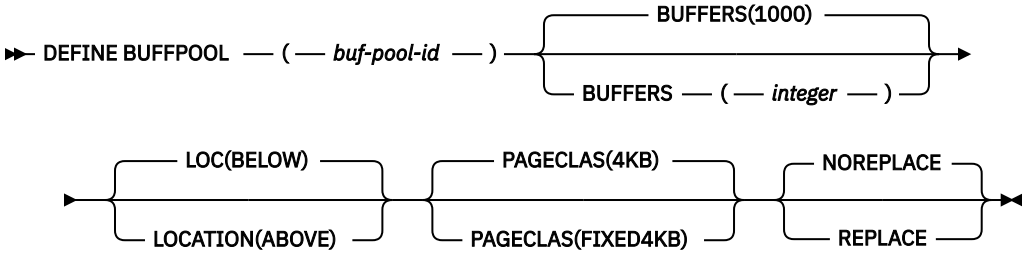
IBM i	UNIX and Linux	Windows	z/OS
			1

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes” on page 827
- “Parameter descriptions for DEFINE BUFFPOOL” on page 827

Synonym: DEF BP

DEFINE BUFFPOOL



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” on page 1220).
3. Use the ALTER BUFFPOOL command to dynamically change the settings of a predefined buffer pool (see “ALTER BUFFPOOL” on page 644).

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 15. If **OPMODE** is set to `OPMODE=(NEWFUNC,800)`, 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 `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.

LOCATION or LOC(BELOW or ABOVE)

The **LOCATION** or **LOC** parameter specifies where the memory used by the specified buffer pool is located.

LOCATION and **LOC** are synonyms and either, but not both, can be used.

This memory location can be either ABOVE (64 bit) or BELOW (31 bit) the bar, with BELOW being the default.

ABOVE can only be specified if OPMODE(NEWFUNC, 800) is in effect. BELOW can be specified regardless of OPMODE(NEWFUNC, 800) being used and has the same effect as not specifying the **LOCATION** parameter.

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 utilized.

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.

FIXED4KB can only be specified if OPMODE(NEWFUNC, 800) is in effect whereas 4KB can be specified regardless of the setting of OPMODE(NEWFUNC, 800).

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 up, 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.

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.

Switching from OPMODE=(NEWFUNC,800) to OPMODE=(COMPAT,800)

When you switch from OPMODE=(NEWFUNC,800) to OPMODE=(COMPAT,800) the following occurs:

1. Any buffer pools with an id greater than 15 are marked as suspended. This means that these buffer pools cannot be used, deleted, or altered until OPMODE=(NEWFUNC,800) is specified again. Information about the buffer pools is kept in check point log records until OPMODE=(NEWFUNC,800) is specified again.

   Any page set that uses a suspended buffer pool is also suspended. Information about the suspended page set is also kept in check point records.

   While suspended, any object definitions or messages in the page set are unavailable. An attempt to use a queue or topic which uses the suspended page set results in an MQRC_PAGESET_ERROR message

   While suspended, a page set can be associated with a different buffer pool by using the FORMAT function of the utility program CSQUTIL, specifying TYPE(REPLACE). You can then issue a **DEFINE PSID** command to bring the page set back into use with a different buffer pool.

   Note: All units of recovery that involved the suspended page set, except units that are indoubt, will have been backed out by the queue manager when the page set was last used. Indoubt units of recovery can be resolved when the page set is again in use by the queue manager.
2. Any buffer pools with an id of 15 or less that have their LOCATION attribute set to ABOVE, will have the LOCATION attribute switched to BELOW, and their PAGECLAS attribute set to 4KB.

z/OS DEFINE CFSTRUCT

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

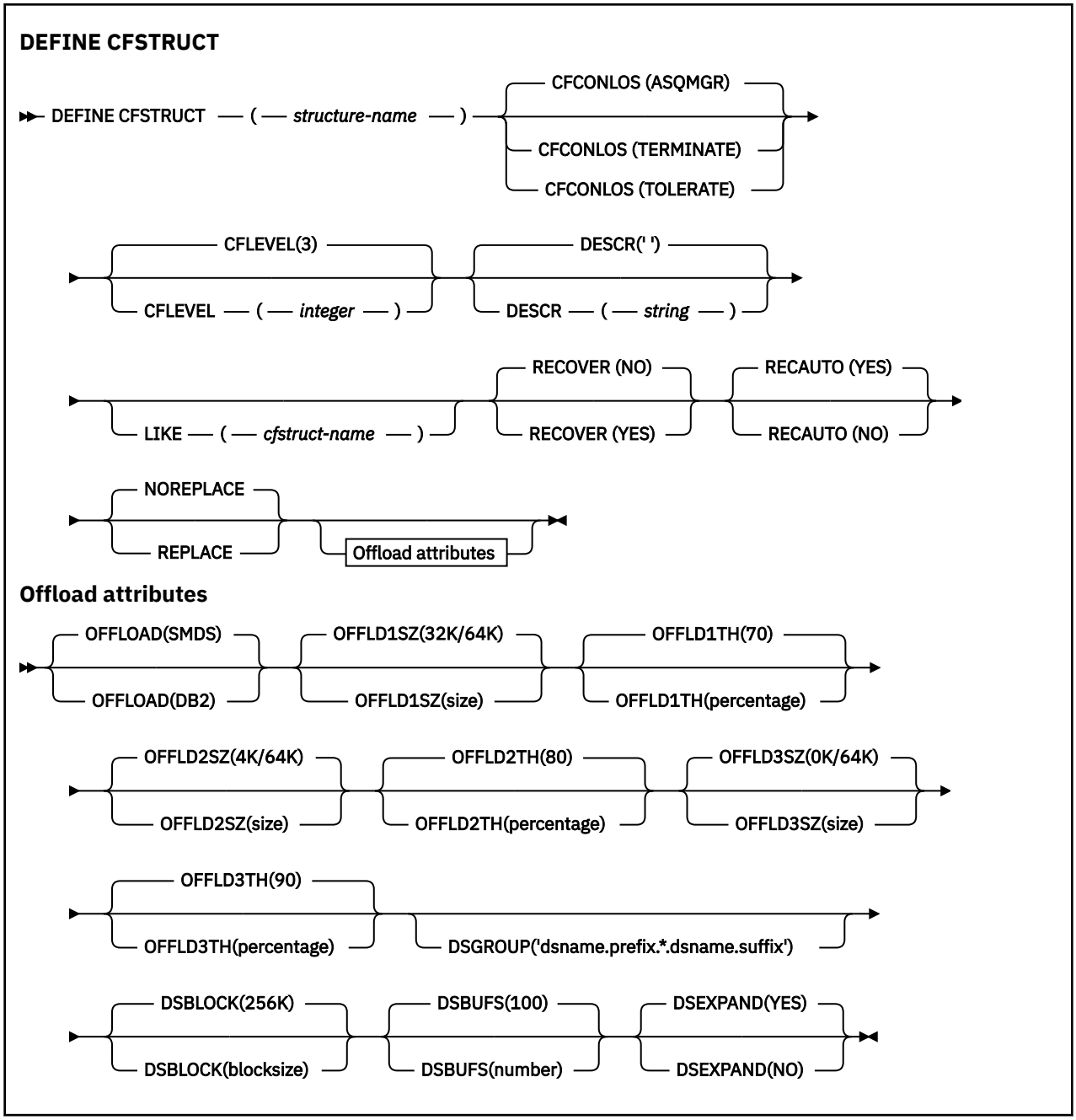
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for DEFINE CFSTRUCT” on page 830
- “Parameter descriptions for DEFINE CFSTRUCT” on page 830

Synonym: DEF CFSTRUCT



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 only increase the value of CFLEVEL to 5 if all the queue managers in the queue-sharing group are at command level 710 or greater and have IBM WebSphere MQ 7.1.0 new functions enabled with OPMODE

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 of headers and descriptors require approximately 400 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 a 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

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

Synonym: DEF CHL

- "Usage notes" on page 837
- "Parameter descriptions for DEFINE CHANNEL" on page 837

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. There is a description of each parameter after the table. Parameters are optional unless the description states that they are required.

SDR
"Sender channel" on page 873

SVR
"Server channel" on page 875

RCVR
"Receiver channel" on page 877

RQSTR
"Requester channel" on page 879

CLNTCONN
"Client-connection channel" on page 881

SVRCONN
"Server-connection channel" on page 883

CLUSSDR
"Cluster-sender channel" on page 885

CLUSRCVR
"Cluster-receiver channel" on page 887

V 8.0.0.4 **AMQP**
"AMQP channel" on page 889

Table 61. DEFINE and ALTER CHANNEL parameters

Parameter	SDR	SVR	RCVR	RQSTR	CLNTCONN	SVRCONN	CLUSSDR	CLUSRCVR	V 8.0.0.4 AMQP
AFFINITY					✓				
V 8.0.0.4 V 8.0.0.4 AMQPKA									✓
BACKLOG									
BATCHHB	✓	✓					✓	✓	
BATCHINT	✓	✓					✓	✓	
BATCHLIM	✓	✓					✓	✓	
BATCHSZ	✓	✓	✓	✓			✓	✓	
CERTLABL	✓	✓	✓	✓	✓	✓		✓	✓
channel-name	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 61. DEFINE and ALTER CHANNEL parameters (continued)									
Parameter	**SDR**	**SVR**	**RCVR**	**RQSTR**	**CLNTCONN**	**SVRCONN**	**CLUSSDR**	**CLUSRCVR**	V 8.0.0.4 **AMQP**
CHLTYPE	✓	✓	✓	✓	✓	✓	✓	✓	✓
CLNTWGHT					✓				
CLUSNL							✓	✓	
CLUSTER							✓	✓	
CLWLPRTY							✓	✓	
CLWLRANK							✓	✓	
CLWLWGHT							✓	✓	
z/OS z/OS CMDSCOPE	✓	✓	✓	✓	✓	✓	✓	✓	
COMPHDR	✓	✓	✓	✓	✓	✓	✓	✓	
COMPMSG	✓	✓	✓	✓	✓	✓	✓	✓	
CONNAME	✓	✓		✓	✓		✓	✓	
CONVERT	✓	✓					✓	✓	
DEFCDISP	✓	✓	✓	✓		✓			
DEFRECON					✓				
DESCR	✓	✓	✓	✓	✓	✓	✓	✓	✓
DISCINT	✓	✓				✓	✓	✓	
HBINT	✓	✓	✓	✓	✓	✓	✓	✓	
JAASCFG									
KAINT	✓	✓	✓	✓	✓	✓	✓	✓	
LIKE	✓	✓	✓	✓	✓	✓	✓	✓	
LOCLADDR	✓	✓		✓	✓		✓	✓	✓
LONGRTY	✓	✓					✓	✓	
LONGTMR	✓	✓					✓	✓	
MAXINST						✓			✓
MAXINSTC						✓			
MAXMSGL	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 61. DEFINE and ALTER CHANNEL parameters (continued)

Parameter	SDR	SVR	RCVR	RQSTR	CLNTCONN	SVRCONN	CLUSSDR	CLUSRCVR	V 8.0.0.4 AMQP
MCANAME	✓	✓		✓			✓	✓	
MCATYPE	✓	✓		✓			✓	✓	
MCAUSER			✓	✓		✓		✓	✓
MODENAME	✓	✓		✓	✓		✓	✓	
MONCHL	✓	✓	✓	✓		✓	✓	✓	
MRDATA			✓	✓				✓	
MREXIT			✓	✓				✓	
MRRTY			✓	✓				✓	
MRTMR			✓	✓				✓	
MSGDATA	✓	✓	✓	✓			✓	✓	
MSGEXIT	✓	✓	✓	✓			✓	✓	
NETPRTY								✓	
NPMSPEED	✓	✓	✓	✓			✓	✓	
PASSWORD	✓	✓		✓	✓		✓	✓	
V 8.0.0.4 V 8.0.0.4 PORT									✓
PROPCTL	✓	✓					✓	✓	
PUTAUT			✓	✓		✓		✓	
QMNAME					✓				
z/OS z/OS QSGDISP	✓	✓	✓	✓	✓	✓	✓	✓	
RCVDATA	✓	✓	✓	✓	✓	✓	✓	✓	
RCVEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
REPLACE	✓	✓	✓	✓	✓	✓	✓	✓	
SCYDATA	✓	✓	✓	✓	✓	✓	✓	✓	
SCYEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
SENDDATA	✓	✓	✓	✓	✓	✓	✓	✓	

Table 61. DEFINE and ALTER CHANNEL parameters (continued)									
Parameter	**SDR**	**SVR**	**RCVR**	**RQSTR**	**CLNTCONN**	**SVRCONN**	**CLUSSDR**	**CLUSRCVR**	V 8.0.0.4 **AMQP**
SENDEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
SEQWRAP	✓	✓	✓	✓			✓	✓	
SHARECNV					✓	✓			
SHORTRTY	✓	✓					✓	✓	
SHORTTMR	✓	✓					✓	✓	
SSLCAUTH		✓	✓	✓		✓		✓	
SSLCIPH	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSLKEYP									
SSLPEER	✓	✓	✓	✓	✓	✓	✓	✓	✓
STATCHL	✓	✓	✓	✓			✓	✓	
TPNAME	✓	✓		✓	✓	✓	✓	✓	
V 8.0.0.4 V 8.0.0.4 TPROOT									✓
TRPTYPE	✓	✓	✓	✓	✓	✓	✓	✓	
V 8.0.0.4 V 8.0.0.4 USECLTID									✓
USEDLQ	✓	✓	✓	✓			✓	✓	
USERID	✓	✓		✓	✓		✓		
XMITQ	✓	✓							

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(PREFFERED), 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.

distributed V 8.0.0.4 **AMQPKA(*integer*)**
The keep alive time for an AMQP channel in seconds. 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. 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.
- z/OS 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 distributed platforms, the maximum number of uncommitted messages allowed at the sending queue manager (or one if this value is zero or less).
- z/OS 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 distributed platforms, 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** parameter.

Note that inbound channels (including receiver, 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.

In all other cases, the queue manager **CERTLABL** parameter determines the certificate sent. In particular, the following only ever receive the certificate configured by the **CERTLABL** parameter of the queue manager, regardless of the channel-specific label setting:

- All current Java and JMS clients.
- Versions of IBM MQ prior to Version 8.0.

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. On CLUSSDR channels, it 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. This facility applies to AIX, HP-UX, IBM i, Linux, Solaris, and Windows only; 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. 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.

CHLTYPE

Channel type. This parameter is required. It must follow immediately after the *(channel-name)* parameter z/OS 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
CLUSSDR channel.

CLUSRCVR
Cluster-receiver channel.

V 8.0.0.4 **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.

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.

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.

z/OS **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 either be left blank, or if QSGDISP is set to GROUP, the local queue manager name.

' '
The command is executed on the queue manager on which it was entered.

QmgrName
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. To do so, you must be using a shared queue environment, and the command server must be 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 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.

z/OS 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.

CONNAME (*string* <, *string* >)
Connection name.

For CLUSRCVR channels, CONNAME relates to the local queue manager, and for other channels it relates to the target queue manager.

z/OS On z/OS, CONNAME is mandatory for CLUSRCVR channels. In addition, whether you specify CONNAME, or the name is generated for you, the CONNAME 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.

z/OS On z/OS, the maximum length of the string is 48 characters.

On distributed platforms, 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.

Providing multiple connection names in a list was first supported in IBM WebSphere MQ 7.0.1. It changes the syntax of the CONNAME parameter. Earlier clients and queue managers connect using the first connection name in the list, and do not read the rest of the connection names in the list. In order for the earlier clients and queue managers to parse the new syntax, you must specify a port number on the first connection name in the list. Specifying a port number avoids problems when connecting to the channel from a client or queue manager that is running at a level earlier than IBM WebSphere MQ 7.0.1.

On AIX, HP-UX, IBM i, Linux, Solaris, and Windows, 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
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 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.

- On AIX, HP-UX, IBM i, Linux, Solaris, 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 z/OS; 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 AIX, HP-UX, Linux, IBM i, Solaris, Windows z/OS, and z/OS, 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
Unless overridden by MQCONNX, the client is not reconnected automatically.

YES
Unless overridden by MQCONNX, the client reconnects automatically.

QMGR
Unless overridden by MQCONNX, 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 MQCONNX MQI call.

Table 62. 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	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 server-connection and client-connection channels working in the channel mode before IBM WebSphere MQ 7.0, heartbeats flow only when a server MCA is waiting for an MQGET command with the WAIT option specified, which it has issued on behalf of a client application.

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.

On distributed platforms, 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 switched on within TCP/IP itself. Refer to your TCP/IP documentation for information about configuring keepalive. On AIX, use the **no** command. On HP-UX, use the **ndd** command. 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.

On distributed platforms, 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 Solaris, for example, flows to (and is implemented by) z/OS queue managers that are in, or join, the cluster.

On distributed platforms, 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:

```
LIKE(SYSTEM.DEF.SENDER)
```

for a SDR channel, and similarly for other channel types.

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 the 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:

Table 63. Examples of how the LOCLADDR parameter can be used

LOCLADDR	Meaning
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

Table 63. Examples of how the LOCLADDR parameter can be used (continued)	
LOCLADDR	**Meaning**
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 64. How the IP stack to be used for communication is determined

Protocols supported	CONNAME	LOCLADDR	Action of channel
IPv4 only	IPv4 address [1]		Channel binds to IPv4 stack
	IPv6 address [2]		Channel fails to resolve CONNAME
	IPv4 and 6 host name [3]		Channel binds to IPv4 stack
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve CONNAME
	IPv4 and 6 host name	IPv4 address	Channel binds to IPv4 stack
	Any address [4]	IPv6 address	Channel fails to resolve LOCLADDR
	IPv4 address	IPv4 and 6 host name	Channel binds to IPv4 stack
	IPv6 address	IPv4 and 6 host name	Channel fails to resolve CONNAME
	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 IPADDRV
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve CONNAME
	IPv4 and 6 host name	IPv4 address	Channel binds to IPv4 stack
	IPv4 address	IPv6 address	Channel maps CONNAME to IPv6 [5]
	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 IPADDRV

Table 64. 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 CONNAME to IPv6 [5]
	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 LOCLADDR
	IPv4 address	IPv6 address	Channel maps CONNAME to IPv6 [5]
	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 CONNAME to IPv6 [5]
	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 of 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.

On AIX, HP-UX, IBM i, Linux, Solaris, and Windows, 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, IBM MQ 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.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, RQSTR, CLUSSDR, or CLUSRCVR. It is supported only on AIX, HP-UX, IBM i, Linux, Solaris, and Windows.

z/OS On z/OS, it 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 **z/OS,**
The user ID assigned to the channel-initiator started task by the z/OS started-procedures table.

TCP/IP, distributed platforms
The user ID from the `inetd.conf` entry, or the user that started the listener.

SNA, distributed platforms
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 64 characters on Windows and 12 characters on other platforms. 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 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 z/OS; 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 us 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 us 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 us 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.

On AIX, HP-UX, Linux, Solaris, 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.

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 65 on page 860.

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.

Table 65. Message exit format and length

Platform	Exit name format	Maximum length	Comment
AIX, HP-UX, Linux, and Solaris	*libraryname*(*functionname*)	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.
Windows	*dllname*(*functionname*)	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. *dllname* is specified without the suffix (.DLL).
IBM i	*progname libname*	20	1. You can specify the names of up to 10 exit programs by specifying multiple strings separated by commas. 2. *program name* occupies the first 10 characters and *libname* the second 10 characters. If necessary, both fields are padded to the right with blanks.
z/OS z/OS	*loadModuleName*	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.

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.

V 8.0.0.4 **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.

Message properties	Result
The message contains a property with a prefix of mcd., jms., usr. or mqext.	If the Support 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 it sent to the remote queue manager.

Message properties	Result
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 `Support` 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 `Support` 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.
The `MQRFH2` folder that would contain the message property needs to be assigned with the `content='properties'` attribute	The properties are removed to prevent `MQRFH2` headers with unsupported syntax flowing to a Version 6 or prior 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 `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. z/OS 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.

On distributed platforms, this parameter is valid only for channels with a channel type (CHLTYPE) of RCVR, RQSTR, or CLUSRCVR.

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.

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).

QSGDISP	**DEFINE**
COPY	The object is defined on the page set of the queue manager that executes the command using the QSGDISP(GROUP) object of the same name as the LIKE object.

GROUP	The object definition resides in the shared repository but only if the queue manager is in a queue-sharing group. If the definition is successful, the following command is generated. The command is sent to all active queue managers in the queue-sharing group to make or refresh local copies on page set zero:
	`DEFINE CHANNEL(channel-name) CHLTYPE(type)`
	`REPLACE QSGDISP(COPY)`
	The **DEFINE** command for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

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.

On AIX, HP-UX, Linux, Solaris, 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.

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.

On AIX, HP-UX, Linux, Solaris, 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.

On IBM i, you can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

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.

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.

On AIX, HP-UX, Linux, Solaris, 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.

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.

On AIX, HP-UX, Linux, Solaris, 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.

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.

SSLCAUTH
SSLCAUTH defines whether IBM MQ requires a certificate from the SSL client. The SSL client is the initiating end of the channel. SSLCAUTH is applied to the SSL server, to determine the behavior required of the client. The SSL 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 SSL client.

OPTIONAL
The peer SSL client system might still send a certificate. If it does, the contents of this certificate are validated as normal.

SSLCIPH *(string)*
SSLCIPH specifies the CipherSpec that is used on the channel. The maximum length is 32 characters. This parameter is valid on all channel types which use transport type TRPTYPE(TCP). If the SSLCIPH parameter is blank, no attempt is made to use SSL on the channel.

The value for this parameter is also used to set the value of SECPROT, which is an output field on the DISPLAY CHSTATUS command.

Specify the name of the CipherSpec you are using. The CipherSpecs that can be used with IBM MQ SSL support are shown in the following table. If a specific named CipherSpec is being used, the **SSLCIPH** values at the two ends of a channel must specify the same named CipherSpec.

Note: IBM i z/OS On z/OS IBM MQ for z/OS; IBM i IBM i; you can also specify the two digit hexadecimal code of a CipherSpec, whether or not it appears in the table. On IBM i, installation of AC3 is a prerequisite for the use of SSL.

Platform support "1" on page 868	CipherSpec name	Protocol used	MAC algorithm	Encryption algorithm	Encryption bits	FIPS "2" on page 868	Suite B
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Yes	No
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_256_CBC_SHA "3" on page 868	TLS 1.0	SHA-1	AES	256	Yes	No
All	ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No
All	ECDHE_ECDSA_AES_256_CBC_SHA384 "3" on page 868	TLS 1.2	SHA-384	AES	256	Yes	No
distributed	ECDHE_ECDSA_AES_128_GCM_SHA256 "4" on page 868	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	128 bit
distributed	ECDHE_ECDSA_AES_256_GCM_SHA384 "3" on page 868 "4" on page 868	TLS 1.2	AEAD AES-128 GCM	AES	256	Yes	192 bit
All	ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No
All	ECDHE_RSA_AES_256_CBC_SHA384 "3" on page 868	TLS 1.2	SHA-384	AES	256	Yes	No
All	ECDHE_RSA_AES_128_GCM_SHA256 "4" on page 868	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	No
All	ECDHE_RSA_AES_256_GCM_SHA384 "3" on page 868 "4" on page 868	TLS 1.2	AEAD AES-128 GCM	AES	SHA384	Yes	No
IBM i "5" on page 868	ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	AEAD AES-128 GCM	AES	SHA256	Yes	No
IBM i	ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	AEAD AES-128 GCM	3DES	SHA256	Yes	No

Platform support "1" on page 868	CipherSpec name	Protocol used	MAC algorithm	Encryption algorithm	Encryption bits	FIPS "2" on page 868	Suite B
IBM i	ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	AEAD AES-128 GCM	3DES	SHA256	Yes	No
IBM i	ECDHE_RSA_RC4_128_SHA256	TLS 1.2	AEAD AES-128 GCM	RSA	SHA256	Yes	No
IBM i	ECDHE_RSA_NULL_SHA256	TLS 1.2	AEAD AES-128 GCM	RSA	SHA256	Yes	No
IBM i	ECDHE_ECDSA_NULL_SHA256	TLS 1.2	AEAD AES-128 GCM	ECDSA	SHA256	Yes	No
IBM i	ECDHE_ECDSA_AES_256_GCM_SHA384 "3" on page 868 "4" on page 868	TLS 1.2	AEAD AES-128 GCM	AES	SHA384	Yes	No
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_256_CBC_SHA256 "3" on page 868	TLS 1.2	SHA-256	AES	256	Yes	No
distributed	TLS_RSA_WITH_AES_128_GCM_SHA256 "4" on page 868	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	No
distributed	TLS_RSA_WITH_AES_256_GCM_SHA384 "3" on page 868 "4" on page 868	TLS 1.2	AEAD AES-128 GCM	AES	256	Yes	No

Notes:

1. If no specific platform is noted, the CipherSpec is available on all platforms.
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. This CipherSpec cannot be used to secure a connection from the MQ Explorer to a queue manager unless the appropriate unrestricted policy files are applied to the JRE used by the Explorer.
4. Following a recommendation by NIST, GCM CipherSpecs have a restriction which means that after 2^32 TLS records are sent, using the same session key, the connection is terminated with message AMQ9288.

   To prevent this error from happening: avoid using GCM Ciphers, enable secret key reset, or start your IBM MQ queue manager with the environment variable GSK_ENFORCE_GCM_RESTRICTION=GSK_FALSE set.

   Important: The GCM restriction is active, regardless of the FIPS mode being used.
5. The CipherSpecs listed as supported on IBM i apply to Versions 7.2 and 7.3 of IBM i.

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 z/OS, UNIX, Linux, and Windows systems, 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.
- On UNIX, Linux, and Windows systems, 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 SSL 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 SSL or TLS Subject distinguished name, is to use channel authentication records. With channel authentication records, different SSL or 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 66. Attribute types supported by SSLPEER.

A two column table describing the attributes supported by the SSLPEER parameter

Attribute	Description
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

Table 66. Attribute types supported by SSLPEER.

A two column table describing the attributes supported by the SSLPEER parameter

(continued)

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 / 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\*')
```

The maximum length of the parameter is 1024 bytes on Windows, IBM i, UNIX and Linux platforms

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.

On z/OS, 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.

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.

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. z/OS See Configuration parameters for an LU 6.2 connection

On Windows SNA Server, z/OS 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.

V 8.0.0.4 **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. 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.

On AIX, HP-UX, Linux, IBM i, Solaris, Windows z/OS, and z/OS, this parameter is optional because, if you do not enter a value, 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.

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
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
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

V 8.0.0.4 **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.

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.

On distributed platforms 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 encrypted and the LU 6.2 software is using a different encryption method, the channel fails to 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.

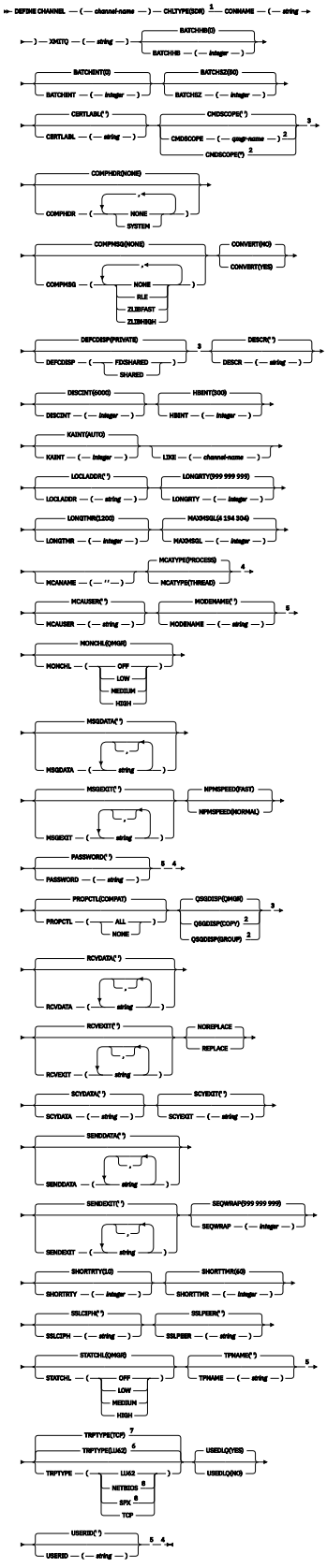
There is a separate syntax diagram for each type of channel:

Sender channel

Syntax diagram for a sender channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



Notes:

1 This parameter must follow immediately after the channel name except on z/OS.

[2] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Not valid on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Default for z/OS.
[7] Default for distributed platforms.
[8] Valid only on Windows.

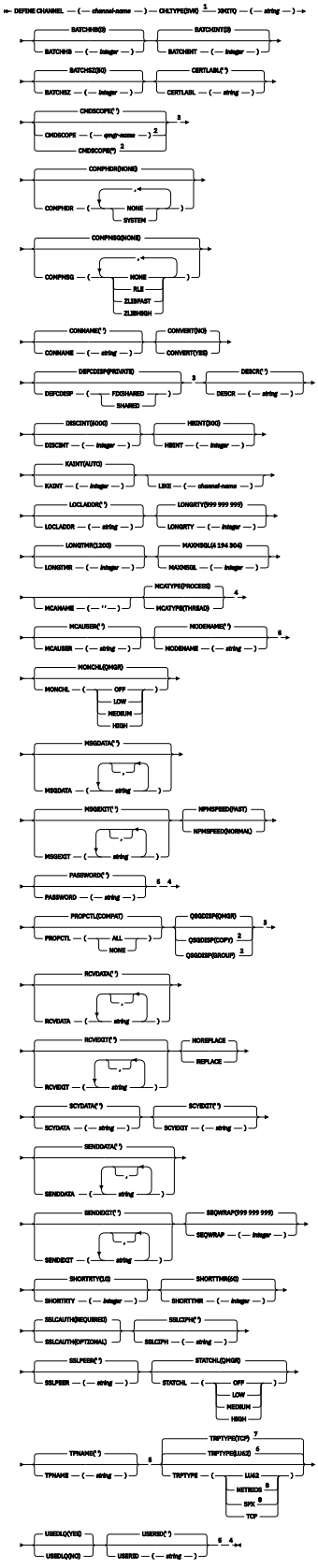
The parameters are described in "DEFINE CHANNEL" on page 836.

Server channel

Syntax diagram for a server channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



Notes:

1 This parameter must follow immediately after the channel name except on z/OS.

[2] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Not valid on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Default for z/OS.
[7] Default for distributed platforms.
[8] Valid only on Windows.

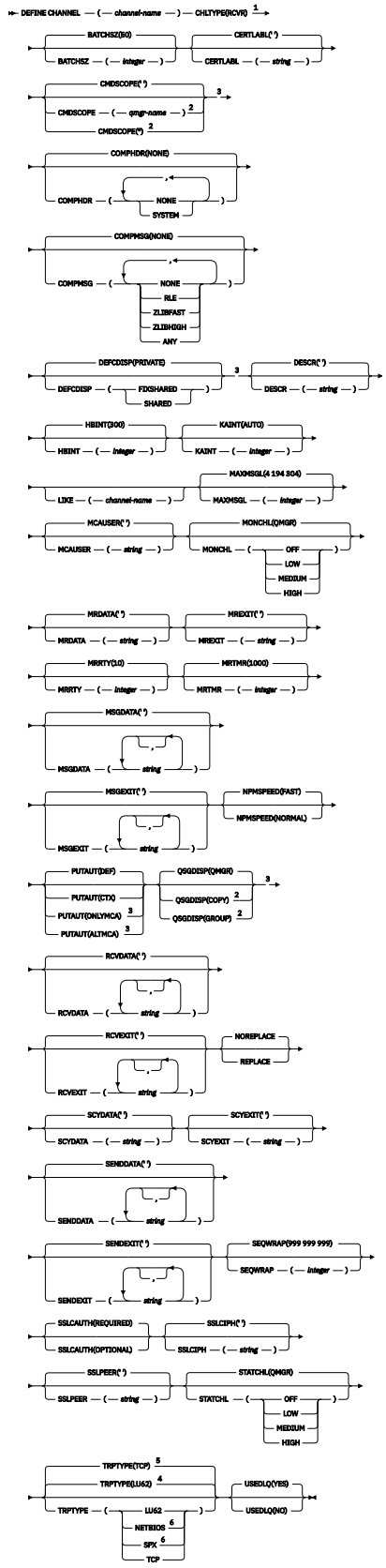
The parameters are described in "DEFINE CHANNEL" on page 836.

Receiver channel

Syntax diagram for a receiver channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



Notes:

1 This parameter must follow immediately after the channel name except on z/OS.

[2] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Default for z/OS.
[5] Default for distributed platforms.
[6] Valid only on Windows.

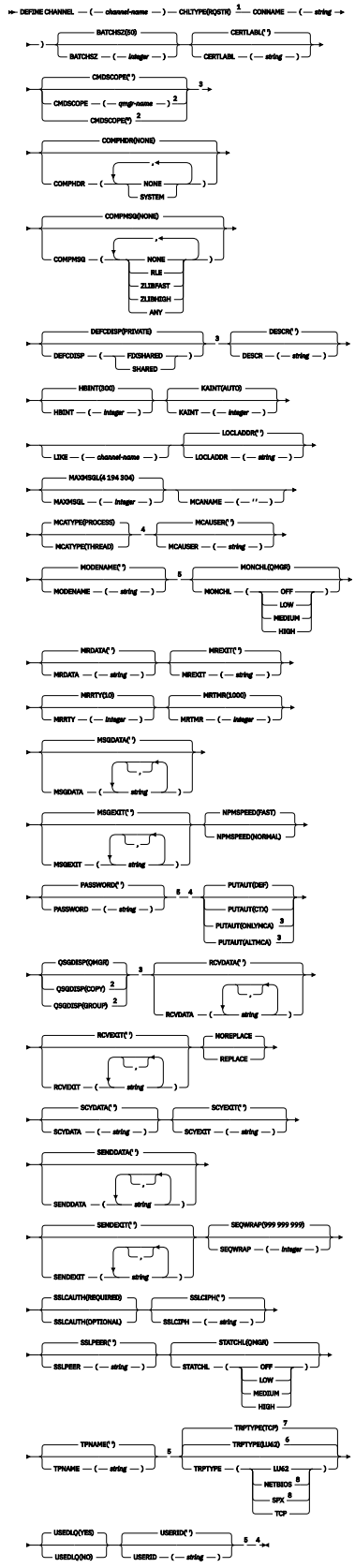
The parameters are described in "DEFINE CHANNEL" on page 836.

Requester channel

Syntax diagram for a requester channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



Notes:

1 This parameter must follow immediately after the channel name except on z/OS.

[2] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Not valid on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Default for z/OS.
[7] Default for distributed platforms.
[8] Valid only on Windows.

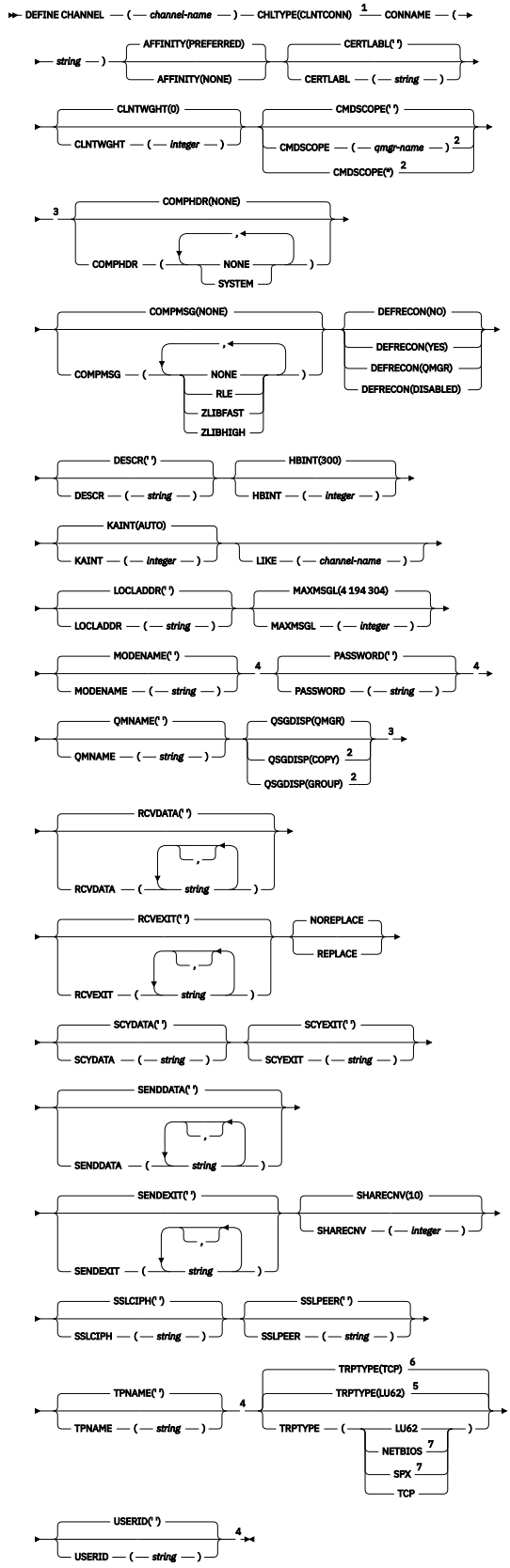
The parameters are described in "DEFINE CHANNEL" on page 836.

Client-connection channel

Syntax diagram for a client-connection channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



Notes:

1 This parameter must follow immediately after the channel name except on z/OS.

[2] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Valid only if TRPTYPE is LU62.
[5] Default for z/OS.
[6] Default for distributed platforms.
[7] Valid only for clients to be run on DOS or Windows.

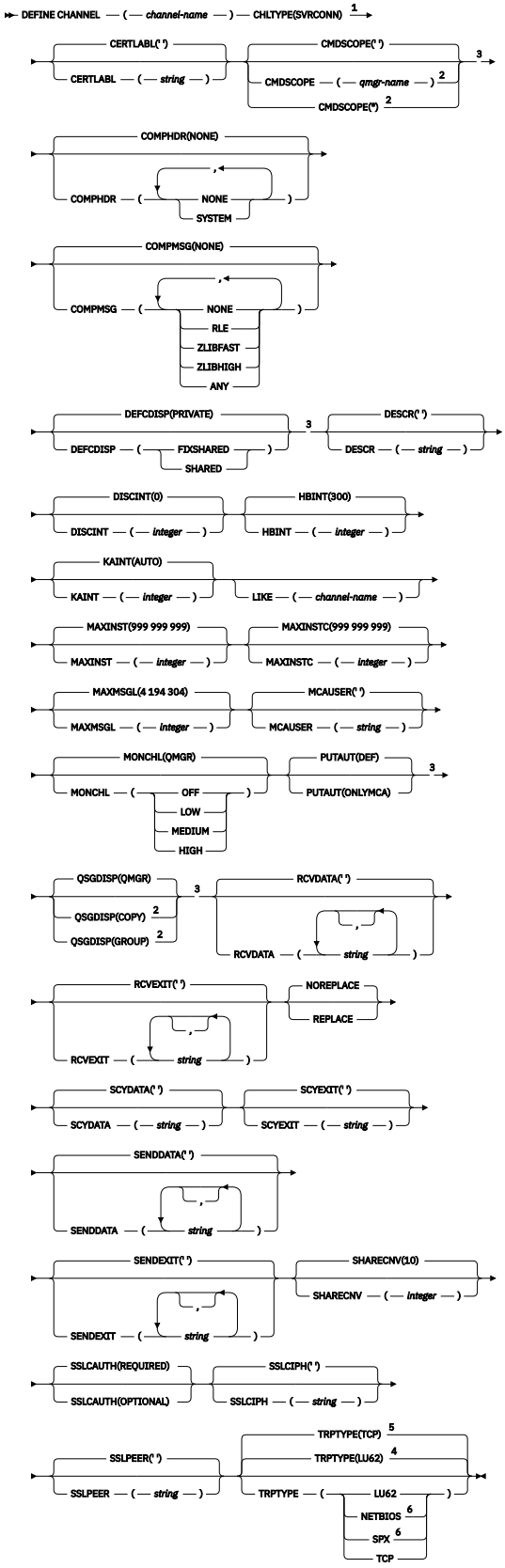
The parameters are described in "DEFINE CHANNEL" on page 836.

Server-connection channel

Syntax diagram for a server-connection channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



Notes:

1 This parameter must follow immediately after the channel name except on z/OS.

[2] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[3] Valid only on z/OS.
[4] Default for z/OS.
[5] Default for distributed platforms.
[6] Valid only for clients to be run on Windows.

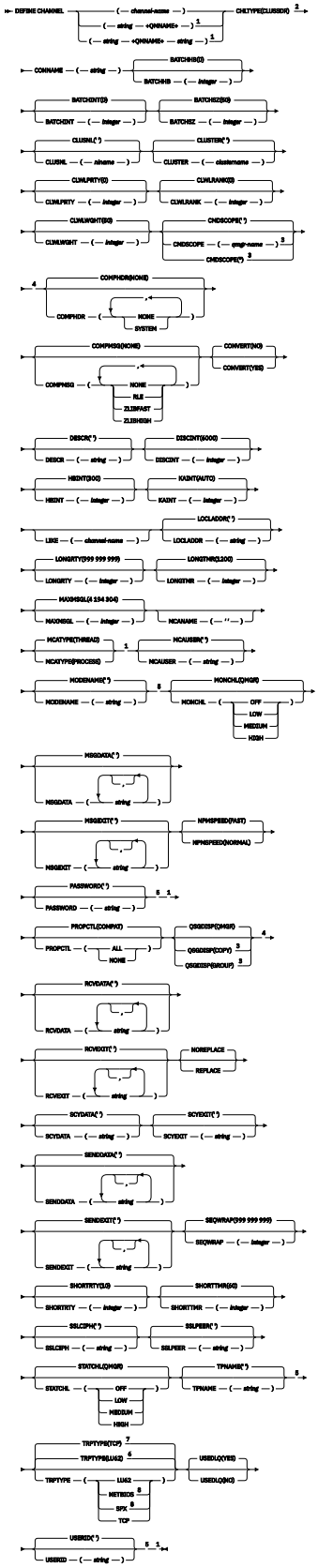
The parameters are described in "DEFINE CHANNEL" on page 836.

Cluster-sender channel

Syntax diagram for a cluster-sender channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



Notes:

1 Not valid on z/OS.

[2] This parameter must follow immediately after the channel name except on z/OS.
[3] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[4] Valid only on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Default for z/OS.
[7] Default for distributed platforms.
[8] Valid only on Windows.

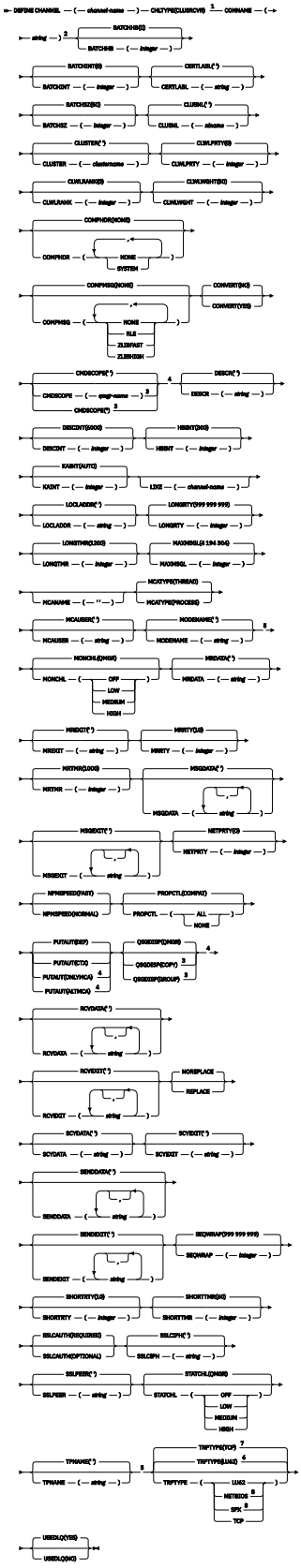
The parameters are described in "DEFINE CHANNEL" on page 836.

Cluster-receiver channel

Syntax diagram for a cluster-receiver channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



Notes:

[1] This parameter must follow immediately after the channel name except on z/OS.

[2] This parameter is optional if TRPTYPE is TCP.
[3] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[4] Valid only on z/OS.
[5] Valid only if TRPTYPE is LU62.
[6] Default for z/OS.
[7] Default for distributed platforms.
[8] Valid only on Windows.

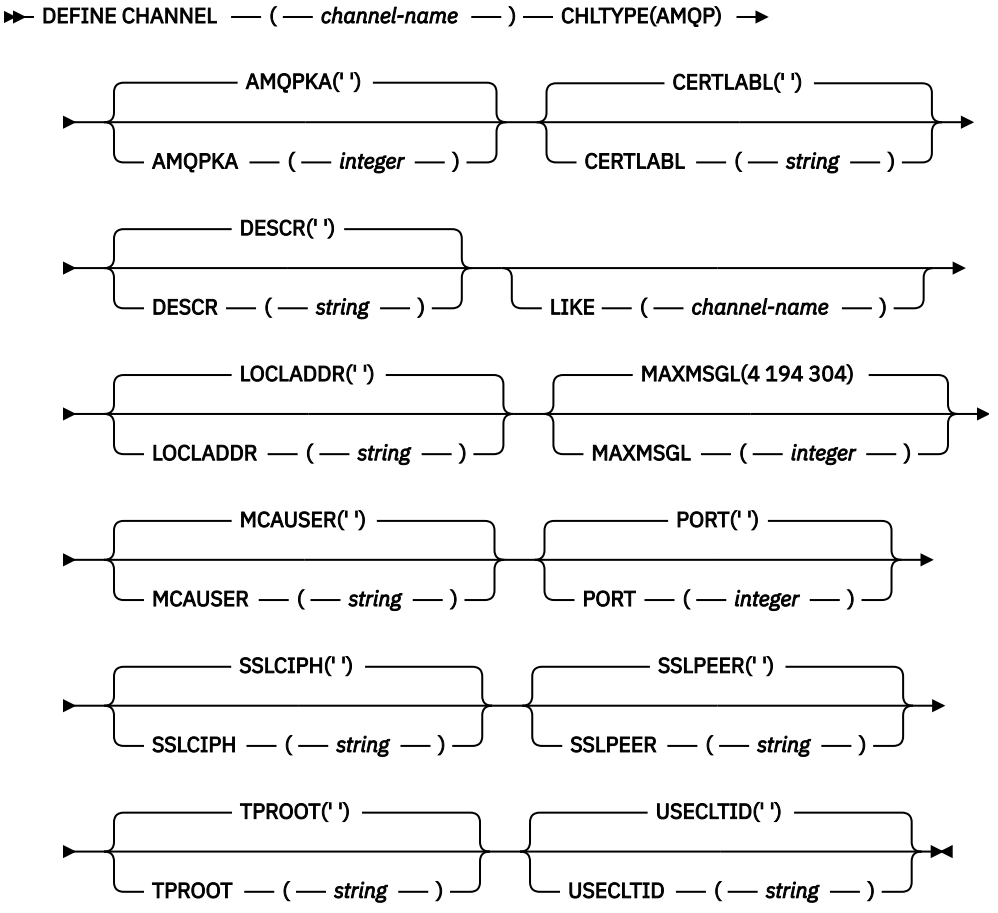
The parameters are described in "DEFINE CHANNEL" on page 836.

distributed V 8.0.0.4 *AMQP channel*

Syntax diagram for an AMQP channel when using the DEFINE CHANNEL command.

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE CHANNEL



The parameters are described in "DEFINE CHANNEL" on page 836.

DEFINE CHANNEL (MQTT)

Syntax diagram for a telemetry channel when using the **DEFINE CHANNEL** command.

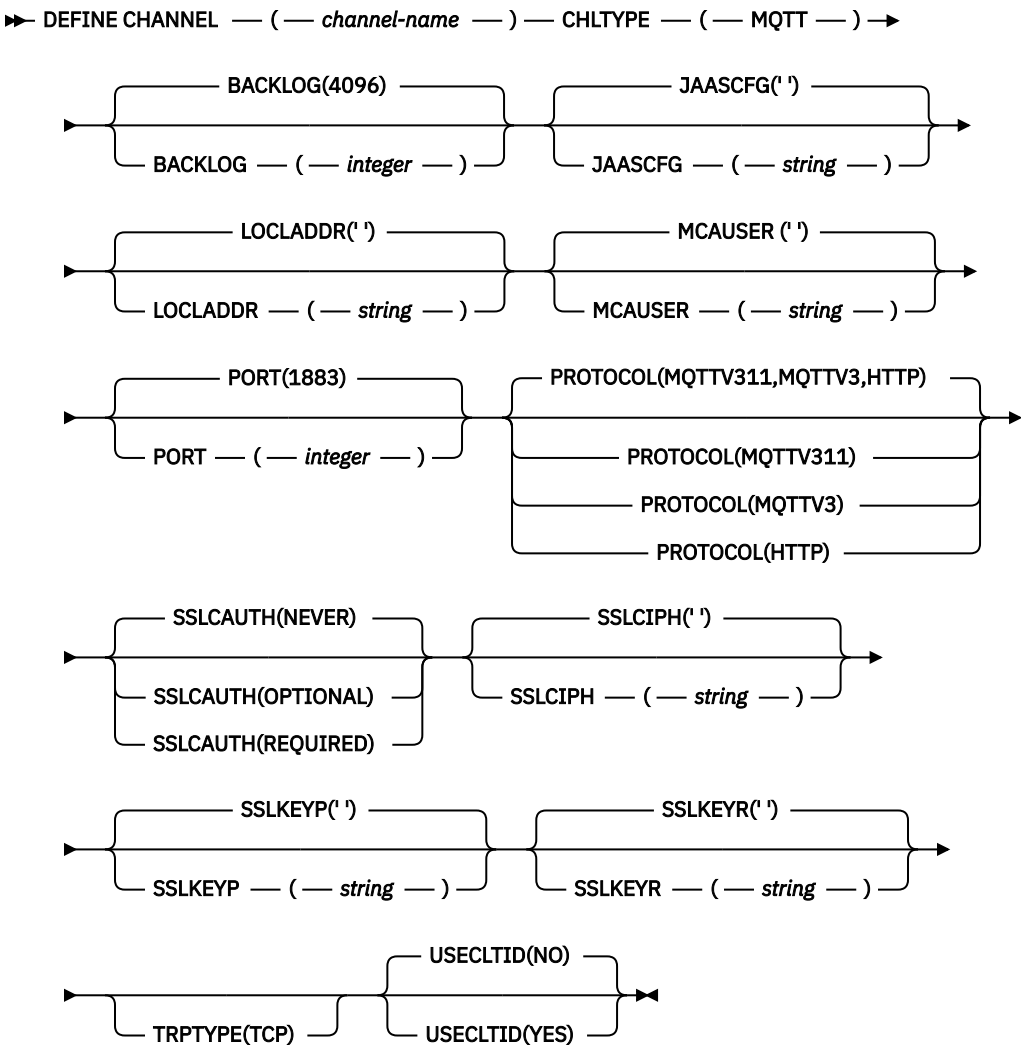
Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	AIX and Linux	Windows	z/OS
	✓	✓	

Note: For the telemetry server, AIX is the only supported UNIX platform.

DEFINE CHANNEL (MQTT)



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:

V 8.0.0.3 **MQTTV311**

The channel accepts connections from clients using the protocol defined by the MQTT Version 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 IBM 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.

V 8.0.0.3 If you are using IBM MQ Version 8.0.0, Fix Pack 3 or later, and your configuration includes an MQTT channel that was last modified in an earlier version of IBM MQ, 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 SSL client. The initiating end of the channel acts as the SSL client, so this parameter applies to the end of the channel that receives the initiation flow, which acts as the SSL server.

NEVER

IBM MQ never requests a certificate from the SSL client.

REQUIRED

IBM MQ requires and validates a certificate from the SSL client.

OPTIONAL

IBM MQ lets the SSL 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 "SSL Cipher Suite". The SSL cipher suite is the one supported by the JVM that is running the telemetry (MQXR) service. If the `SSLCIPH` parameter is blank, no attempt is made to use SSL 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 Secure Sockets Layer (SSL) key repository.

SSLKEYR (*string*)

The full path name of the Secure Sockets Layer (SSL) key repository file, the store for digital certificates and their associated private keys. If you do not specify a key file, SSL is not used.

The maximum length of the string is 256 characters. On UNIX and Linux, the name is of the form *pathname/keyfile* and on Windows *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 SSL
Telemetry channel configuration for channel authentication using SSL
CipherSpecs and CipherSuites
Related reference
“ALTER CHANNEL (MQTT)” on page 705
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

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 Performing local administration tasks using MQSC commands.

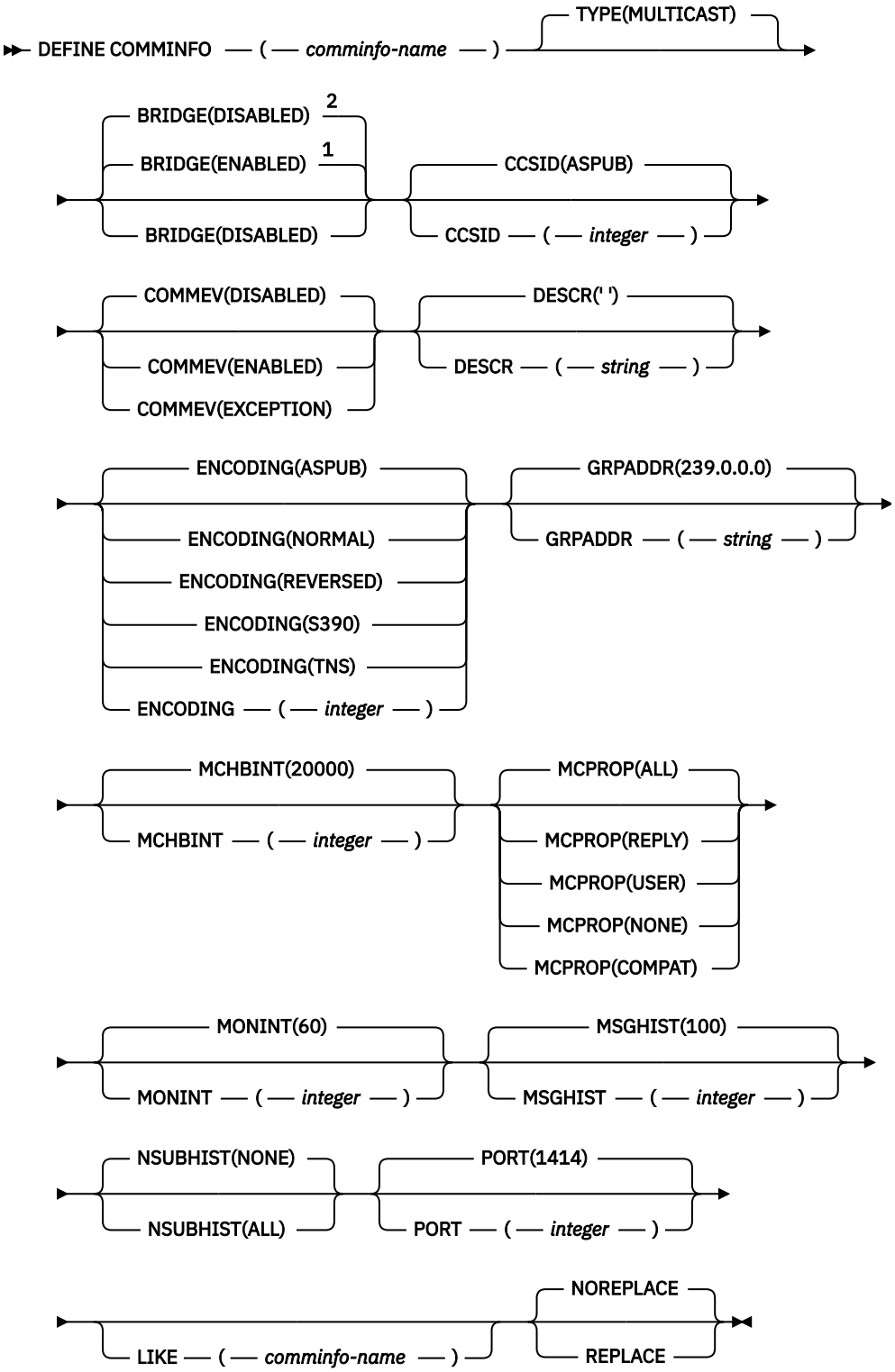
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Parameter descriptions for DEFINE COMMINFO” on page 895

Synonym: DEF COMMINFO

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See “How to read railroad diagrams” on page 7.

DEFINE COMMINFO



Notes:

[1] Default for platforms other than IBM i.

[2] 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” on page 1079).

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.

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.

DEFINE LISTENER

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 Performing local administration tasks using MQSC commands.

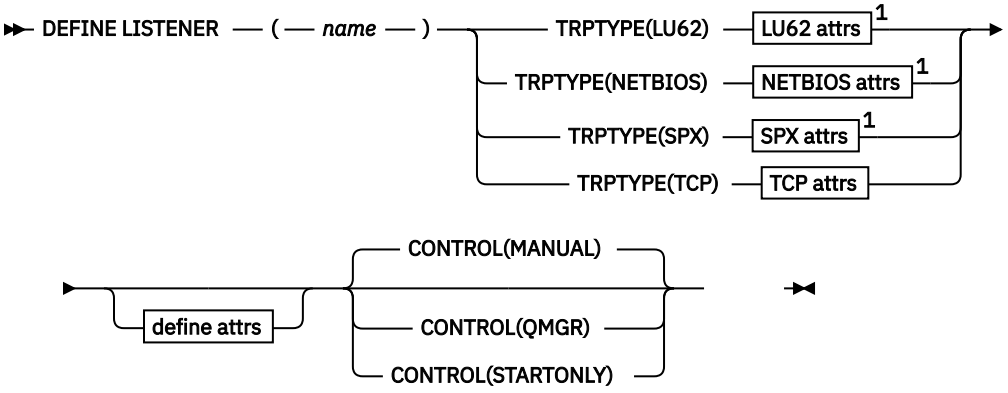
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Parameter descriptions for DEFINE LISTENER” on page 899

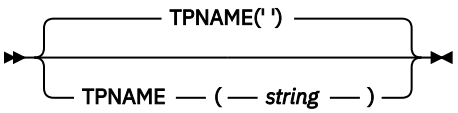
Synonym: DEF LSTR

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See “How to read railroad diagrams” on page 7.

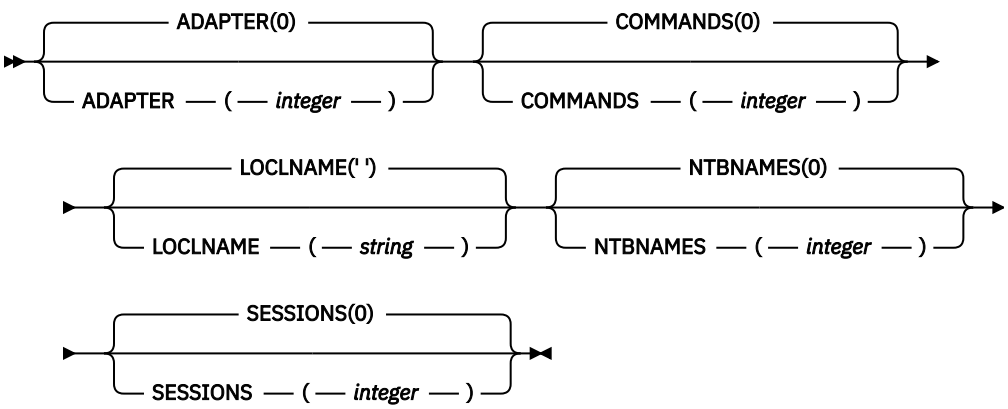
DEFINE LISTENER



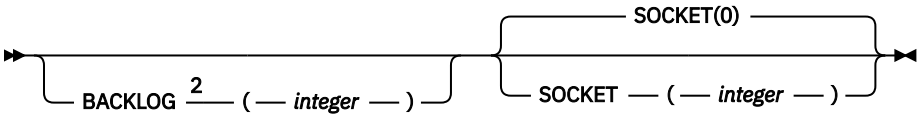
LU62 attrs



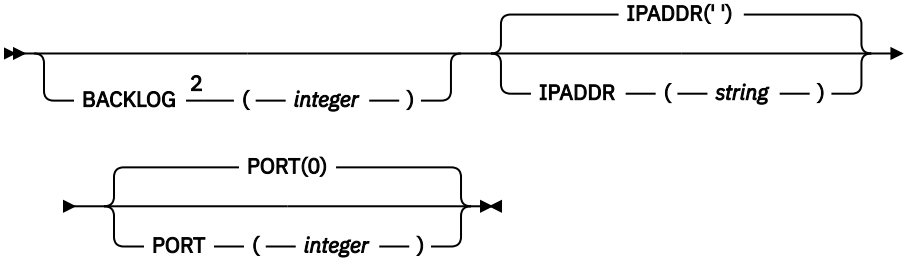
NETBIOS attrs



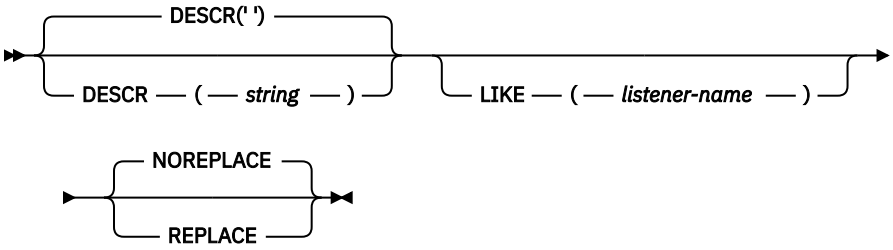
SPX attrs



TCP attrs



Define attrs



Notes:

[1] Valid only on Windows.

[2] When the BACKLOG attribute is left unchanged or when it is explicitly set to zero, the attribute is stored as zero by default in the listener object created by the **DEFINE LISTENER** command. However, when the listener is started, the default backlog value takes effect. For information about the default value of the BACKLOG attribute, see Using the TCP listener backlog option.

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).

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.

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” on page 1099).

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.

LOCLNAME(*string*)
The NetBIOS local name that the listener uses. This parameter is valid only on Windows when TRPTYPE is NETBIOS.

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.

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.

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:

LU62
SNA LU 6.2. This is valid only on Windows.

NETBIOS
NetBIOS. This is valid only on Windows.

SPX
Sequenced packet exchange. This is valid only on Windows.

TCP
TCP/IP.

z/OS DEFINE LOG

Use the MQSC command DEFINE LOG to add a new active log data set in the ring of active logs.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			CR

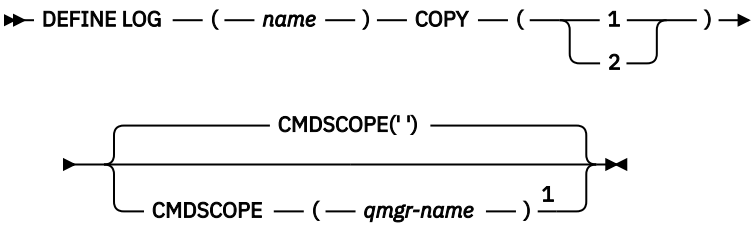
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

The named data set is dynamically allocated to the running queue manager, added to either the COPY1 or COPY2 active log and the BSDS updated with the information so it is retained over a queue manager restart. The data set is added to the active log ring in a position such that it will be the next active log used when the current active log fills and an active log switch occurs.

- Syntax diagram
- "Usage note for DEFINE LOG" on page 901
- "Parameter descriptions for DEFINE LOG" on page 901

Synonym: DEF LOG

DEFINE LOG



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

Usage note for DEFINE LOG

If a log data set has to be added because there is no more log space and the queue manager is waiting, you must issue the command from the z/OS console, and not through the command server.

Parameter descriptions for DEFINE LOG

(name)

The name of the new log data set. This is required and is the name of a VSAM linear data set which will have already been defined by Access Method Services (and, optionally, formatted by utility CSQJUFMT). This is allocated dynamically to the queue manager.

The maximum length of the string is 44 characters. The string must conform to z/OS data set naming conventions.

COPY

Specifies the number of an active log ring to which to add the new log data set. This is either 1 or 2 and is required.

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 it was entered, only if you are using a shared queue environment and if the command server is enabled.

z/OS DEFINE MAXSMSGS

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

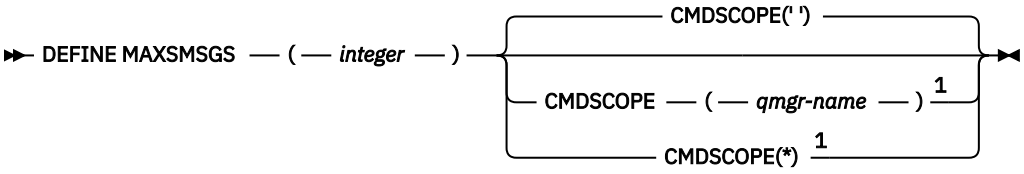
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 902
- "Parameter descriptions for DEFINE MAXSMSGS" on page 902

Synonym: DEF MAXSM

DEFINE MAXSMSGS



Notes:

1 Valid only when the queue manager is a member of a queue-sharing group.

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 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 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 this is the same as entering the command on every queue manager in the queue-sharing group.

DEFINE NAMELIST

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 Performing local administration tasks using MQSC commands.

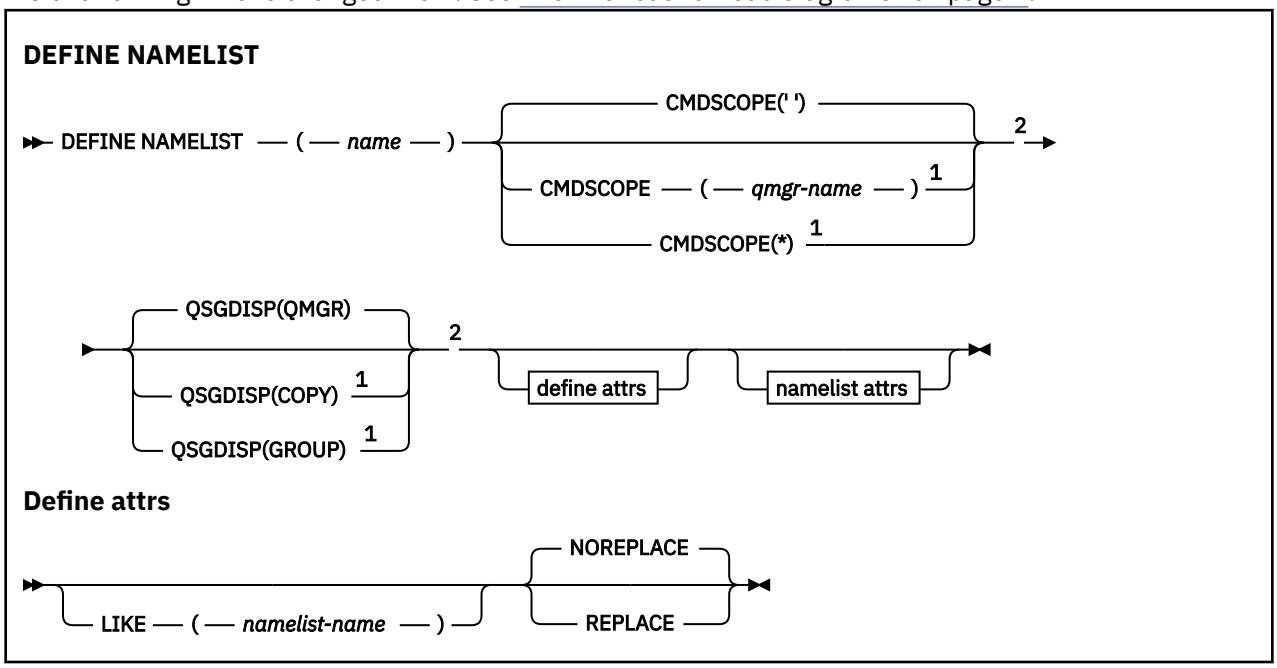
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

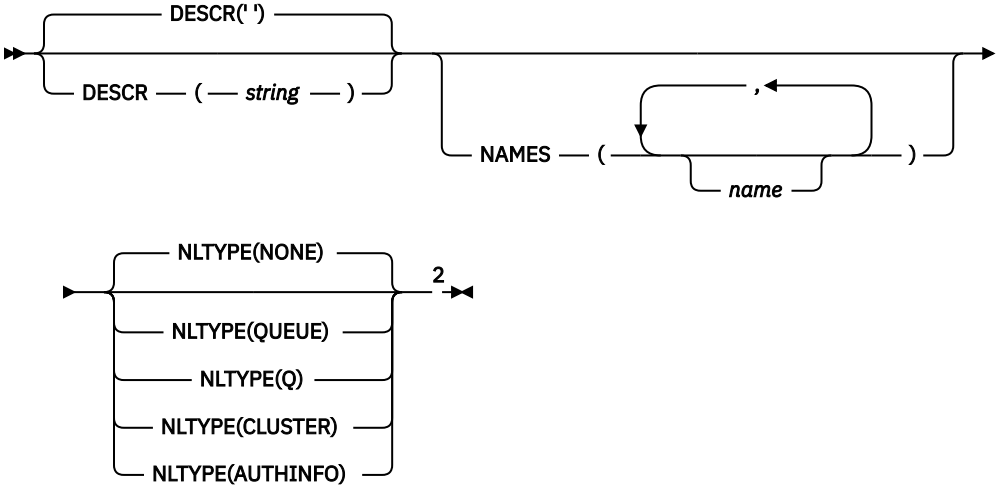
- Syntax diagram
- “Usage notes” on page 904
- “Parameter descriptions for DEFINE NAMELIST” on page 904

Synonym: DEF NL

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See “How to read railroad diagrams” on page 7.



Namelist attrs



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.

Usage notes

- On UNIX systems, the command is valid only on AIX, HP-UX, Linux and Solaris.
- Successful completion of the command does not mean that the action completed. To check for true completion, see the DEFINE NAMELIST step in Checking that async commands for distributed networks have finished.

Parameter descriptions for DEFINE 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 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 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" on page 1108).

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.

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 SSL and contains a list of authentication information object names.

Namelists used for clustering must have NLTYPE(CLUSTER) or NLTYPE(NONE).

Namelists used for SSL 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	DEFINE
COPY	The object is defined on the page set of the queue manager that executes the command using the QSGDISP(GROUP) object of the same name as the 'LIKE' object.

GROUP	The object definition resides in the shared repository but only if the queue manager is in a queue-sharing group. If the definition is successful, the following 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:
	`DEFINE NAMELIST(name)`
	`REPLACE QSGDISP(COPY)`
	The DEFINE for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

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.

DEFINE PROCESS

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

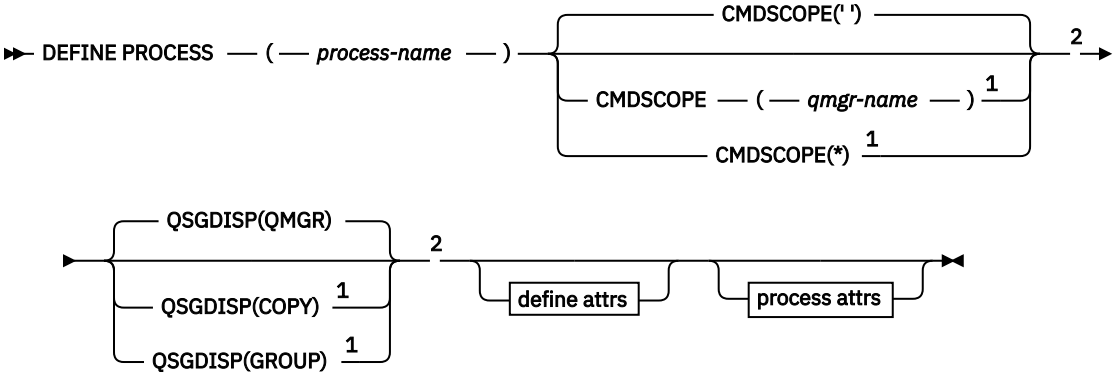
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for DEFINE PROCESS” on page 908

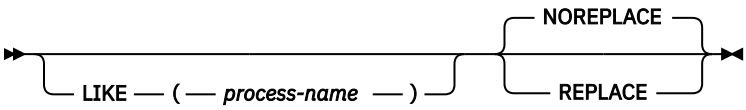
Synonym: DEF PRO

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See “How to read railroad diagrams” on page 7.

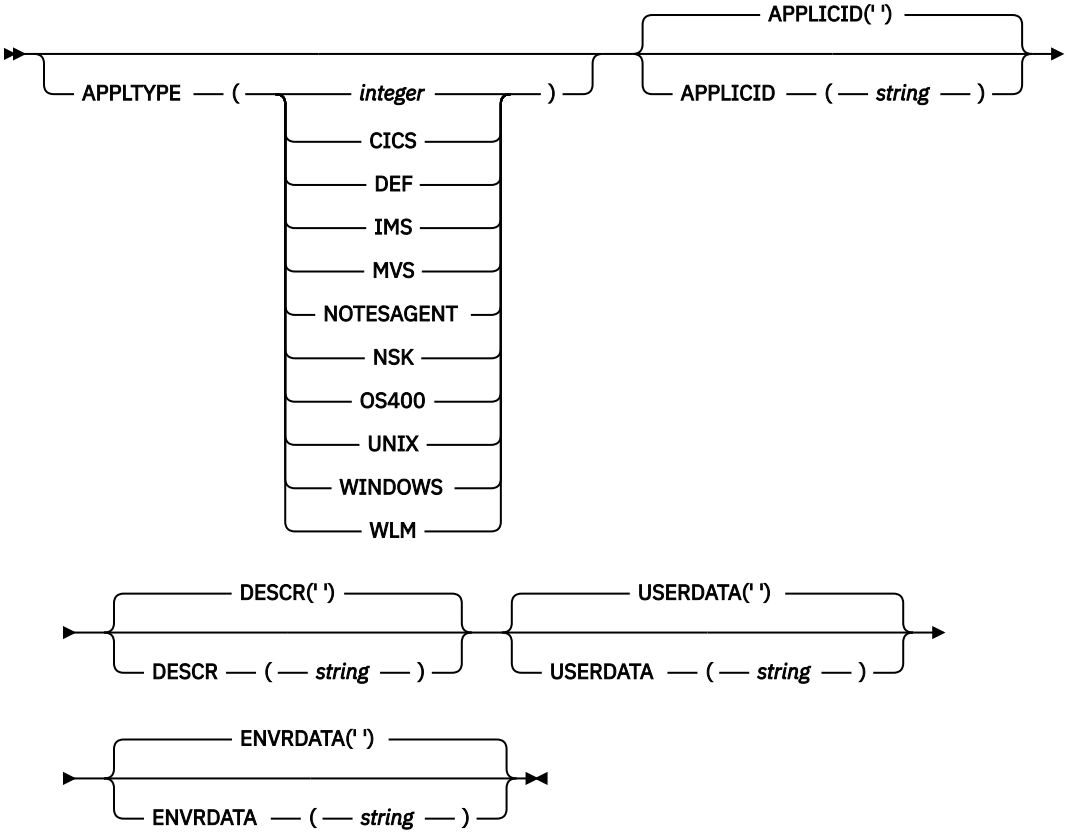
DEFINE PROCESS



Define attrs



Process attrs



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] The default depends on the platform, and can be changed by your installation.

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).

NOTESAGENT
Represents a Lotus Notes agent.

NSK
Represents an HP Integrity NonStop Server application.

IBM i **OS400**
Represents an IBM i application.

UNIX
Represents a UNIX 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
- On IBM i, OS400, CICS, and DEF are supported
- On UNIX systems, UNIX, WINDOWS, CICS, and DEF are supported
- On Windows, WINDOWS, UNIX, CICS, and DEF are supported

z/OS **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.

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 is executed 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. z/OS On z/OS, ENVRDATA is not used by the trigger-monitor applications provided by IBM MQ.
2. 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.

3. On UNIX systems, 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.

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.

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	**DEFINE**
COPY	The object is defined on the page set of the queue manager that executes the command. It uses the QSGDISP(GROUP) object of the same name as the 'LIKE' object.

GROUP	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.
	`DEFINE PROCESS(process-name)`
	`REPLACE QSGDISP(COPY)`
	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 DEFINE for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

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 z/OS (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**.

z/OS DEFINE PSID

Use the MQSC command DEFINE PSID to define a page set and associated buffer pool.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

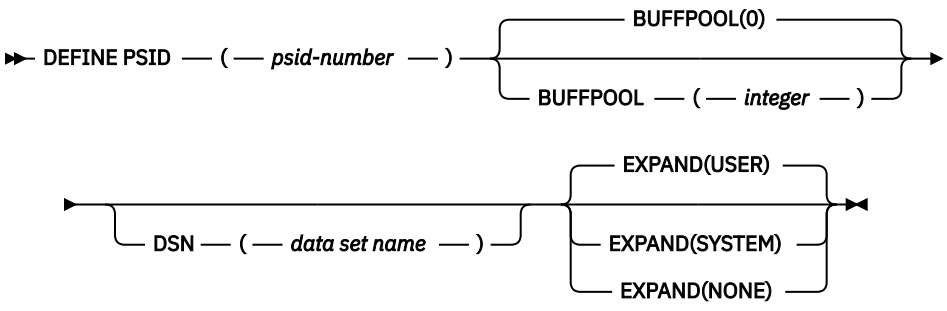
IBM i	UNIX and Linux	Windows	z/OS
			1CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for DEFINE PSID" on page 912
- "Parameter descriptions for DEFINE PSID" on page 912

Synonym: DEF PSID

DEFINE 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” on page 1220).

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 (in the range zero through 15). If OPMODE is set to OPMODE=(NEWFUNC, 800), this 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 Performing local administration tasks using MQSC commands.

This section contains the following commands:

- "DEFINE QALIAS" on page 936
- "DEFINE QLOCAL" on page 938
- "DEFINE QMODEL" on page 941
- "DEFINE QREMOTE" on page 944

Define a reply-to queue or queue-manager alias with the "DEFINE QREMOTE" on page 944 command.

These commands are supported on the following platforms:

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on 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. z/OS 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:
  1. 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.
  2. 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.
  3. 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.
  4. 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.
  5. 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 67 on page 914 shows the parameters that are relevant for each type of queue. There is a description of each parameter after the table.

Table 67. DEFINE and ALTER QUEUE parameters

Parameter	Local queue	Model queue	Alias queue	Remote queue
ACCTQ	✓	✓		
BOQNAME	✓	✓		
BOTHRESH	✓	✓		

Table 67. DEFINE and ALTER QUEUE parameters (continued)				
Parameter	**Local queue**	**Model queue**	**Alias queue**	**Remote queue**
z/OS z/OS CFSTRUCT	✓	✓		
CLCHNAME	✓			
CLUSNL	✓		✓	✓
CLUSTER	✓		✓	✓
CLWLPRTY	✓		✓	✓
CLWLRANK	✓		✓	✓
CLWLUSEQ	✓			
z/OS z/OS CMDSCOPE	✓	✓	✓	✓
CUSTOM	✓	✓	✓	✓
DEFBIND	✓		✓	✓
DEFPRESP	✓	✓	✓	✓
DEFPRTY	✓	✓	✓	✓
DEFPSIST	✓	✓	✓	✓
DEFREADA	✓	✓	✓	
DEFSOPT	✓	✓		
DEFTYPE		✓		
DESCR	✓	✓	✓	✓
DISTL	✓	✓		
FORCE	✓		✓	✓
GET	✓	✓	✓	
HARDENBO or NOHARDENBO	✓	✓		
INDXTYPE	✓	✓		
INITQ	✓	✓		
LIKE	✓	✓	✓	✓
MAXDEPTH	✓	✓		

Table 67. DEFINE and ALTER QUEUE parameters (continued)				
Parameter	**Local queue**	**Model queue**	**Alias queue**	**Remote queue**
MAXMSGL	✓	✓		
MONQ	✓	✓		
MSGDLVSQ	✓	✓		
NOREPLACE	✓	✓	✓	✓
NPMCLASS	✓	✓		
PROCESS	✓	✓		
PROPCTL	✓	✓	✓	
PUT	✓	✓	✓	✓
queue-name	✓	✓	✓	✓
QDEPTHHI	✓	✓		
QDEPTHLO	✓	✓		
QDPHIEV	✓	✓		
QDPLOEV	✓	✓		
QDPMAXEV	✓	✓		
z/OS z/OS QSGDISP	✓	✓	✓	✓
QSVCIEV	✓	✓		
QSVCINT	✓	✓		
REPLACE	✓	✓	✓	✓
RETINTVL	✓	✓		
RNAME				✓
RQMNAME				✓
SCOPE	✓		✓	✓
SHARE or NOSHARE	✓	✓		
STATQ	✓	✓		
z/OS z/OS STGCLASS	✓	✓		

Table 67. DEFINE and ALTER QUEUE parameters (continued)				
Parameter	**Local queue**	**Model queue**	**Alias queue**	**Remote queue**
TARGET			✓	
TARGQ			✓	
TARGTYPE			✓	
TRIGDATA	✓	✓		
TRIGDPTH	✓	✓		
TRIGGER or NOTRIGGER	✓	✓		
TRIGMPRI	✓	✓		
TRIGTYPE	✓	✓		
USAGE	✓	✓		
XMITQ				✓

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 MQCONNX 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 switch on 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.

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.

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 nonblank; you cannot specify a value for both.

On local queues, this parameter cannot be set for transmission, SYSTEM.CHANNEL. *xx* , SYSTEM.CLUSTER. *xx* , or SYSTEM.COMMAND. *xx* queues z/OS, and on z/OS only, for SYSTEM.QSG. *xx* queues.

This parameter is valid only on AIX, HP-UX, Linux, Solaris, and Windows z/OS, and z/OS.

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 nonblank; you cannot specify a value for both.

On local queues, this parameter cannot be set for transmission, SYSTEM.CHANNEL. *xx* , SYSTEM.CLUSTER. *xx* , or SYSTEM.COMMAND. *xx* queues z/OS, and on z/OS only, for SYSTEM.QSG. *xx* queues.

This parameter is valid only on AIX, HP-UX, Linux, Solaris, and Windows z/OS, and 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 is run on the queue manager on which it was entered.

QmgrName
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. You can specify another name, 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 * is the same as entering the command on every queue manager in the queue-sharing group.

CUSTOM(*string*)
The custom attribute for new features.

V 8.0.0.4 This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name and value pairs have the form NAME(VALUE). Single quotation marks must be escaped with another single quotation mark.

CAPEXPRY(*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.

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 CAPEXPRY 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 **CAPEXPRY**, 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 **CAPEXPRY** have the new expiry time.

The maximum length is defined by the constant MQ_CUSTOM_LENGTH and is currently set to 128 on all platforms.

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.

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, MQOO_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 only on AIX, HP-UX, Linux, IBM i, Solaris, and Windows

, and z/OS.

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.

z/OS 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 in 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

SHARED
The open request is for shared input from the queue

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.

Do not specify this value for a model queue definition with a **DEFPSIST** parameter of YES.

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.

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, HP-UX, Linux, Solaris, 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 are true:

- The **TARGET** parameter specifies a queue
- An application has this alias queue open

For a local queue, if both of the following 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 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 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 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 and 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 manger.

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: z/OS This parameter affects only IBM MQ for 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 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:

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 INDXTYPE 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	-	-
MSGTOKEN	CORRELID	No	Yes	Yes	-	-

Queue type		NON-SHARED			SHARED	
MSGTOKEN	GROUPID	No	No	Yes	-	-
GROUPID	NONE	No	No	Yes	No	Yes
GROUPID	MSGID	No	No	Yes	No	Yes
GROUPID	CORRELID	No	No	Yes	No	Yes
GROUPID	MSGTOKEN	No	No	Yes	-	-

This parameter is supported only on z/OS. On other platforms, all queues are automatically indexed.

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:

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.

MAXDEPTH(*integer*)
The maximum number of messages allowed on the queue.

This parameter is supported only on local and model queues.

On AIX, HP-UX, Linux, Solaris, and Windows z/OS, and z/OS , specify a value in the range zero through 999999999.

This parameter is valid only on AIX, HP-UX, Linux, Solaris, and Windows z/OS, and 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.

MAXMSGL(*integer*)
The maximum length (in bytes) of messages on this queue.

This parameter is supported only on local and model queues.

On AIX, HP-UX, Linux, Solaris, 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, IBM MQ 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: z/OS 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.

z/OS 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 AIX, HP-UX, IBM i, Linux, Solaris, and Windows z/OS, and 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. z/OS For more information about the effect that shared queues on z/OS have on this event; see Shared queues and queue depth events (IBM MQ for 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. z/OS For more information about the effect that shared queues on z/OS have on this event; see Shared queues and queue depth events (IBM MQ for 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.

Note: The value of this parameter can change implicitly z/OS, and shared queues on z/OS affect the event. See the description of Queue Depth High.

ENABLED
Queue Depth High events are generated

DISABLED
Queue Depth High events are not generated

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.

Note: The value of this parameter can change implicitly z/OS, and shared queues on z/OS affect the event. For more information about this event, see Queue Depth Low.

ENABLED
Queue Depth Low events are generated

DISABLED
Queue Depth Low events are not generated

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.

Note: The value of this parameter can change implicitly z/OS, and shared queues on z/OS affect the event. For more information about this event, see Queue Full.

ENABLED
Queue Full events are generated

DISABLED
Queue Full events are not generated

z/OS **QSGDISP**

This parameter applies to z/OS only.

Specifies the disposition of the object within the group.

Table 68. QSGDISP parameters.

Definitions of the QSGDISP parameters when defining a queue.

QSGDISP	DEFINE
COPY	The object is defined on the page set of the queue manager that executes the command using the QSGDISP(GROUP) object of the same name as the LIKE object. For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.

GROUP	The object definition resides in the shared repository but only if there is a shared queue manager environment. If the definition is successful, the following command is generated. The command is sent to all active queue managers to attempt to make or refresh local copies on page set zero:
	`DEFINE QUEUE(q-name)`
	`REPLACE QSGDISP(COPY)`
	The **DEFINE** command for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

PRIVATE	Not permitted.
QMGR	The object is defined on the page set of the queue manager that executes 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 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: • CFSTRUCT is nonblank • INDXTYPE is not MSGTOKEN • The queue is not: – SYSTEM.CHANNEL.INITQ – SYSTEM.COMMAND.INPUT 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.

QSVCIEV

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 **QSVCINT** 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 **QSVCINT** 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

QSVCINT(*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 **QSVCIEV** 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: z/OS 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.

Windows Linux UNIX If **SCOPE(CELL)** is specified on UNIX, Linux, or 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.

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 UNIX and Linux systems, and Windows.

SHAREand 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.

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.

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.

This parameter is valid only on IBM i, UNIX and Linux systems, and Windows.

z/OS **STGCLASS(*string*)**
The name of the storage class.

This parameter is supported only on local and model queues.

This parameter is an installation-defined name.

This parameter is valid on z/OS only z/OS; see Storage classes .

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.

Note: You can change this parameter only if the queue is empty and closed.

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 **TARGTYPE**. 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**.

TARGTYPE(*string*)
The type of object to which the alias resolves.

QUEUE
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 on AIX, HP-UX, IBM i, Linux, Solaris, and Windows z/OS, and z/OS, 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.

This parameter is supported only on local and model queues.

This parameter can also be changed using the MQSET API call.

TRIGGER and 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 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" on page 1122 for details.

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.

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.

DEFINE QALIAS

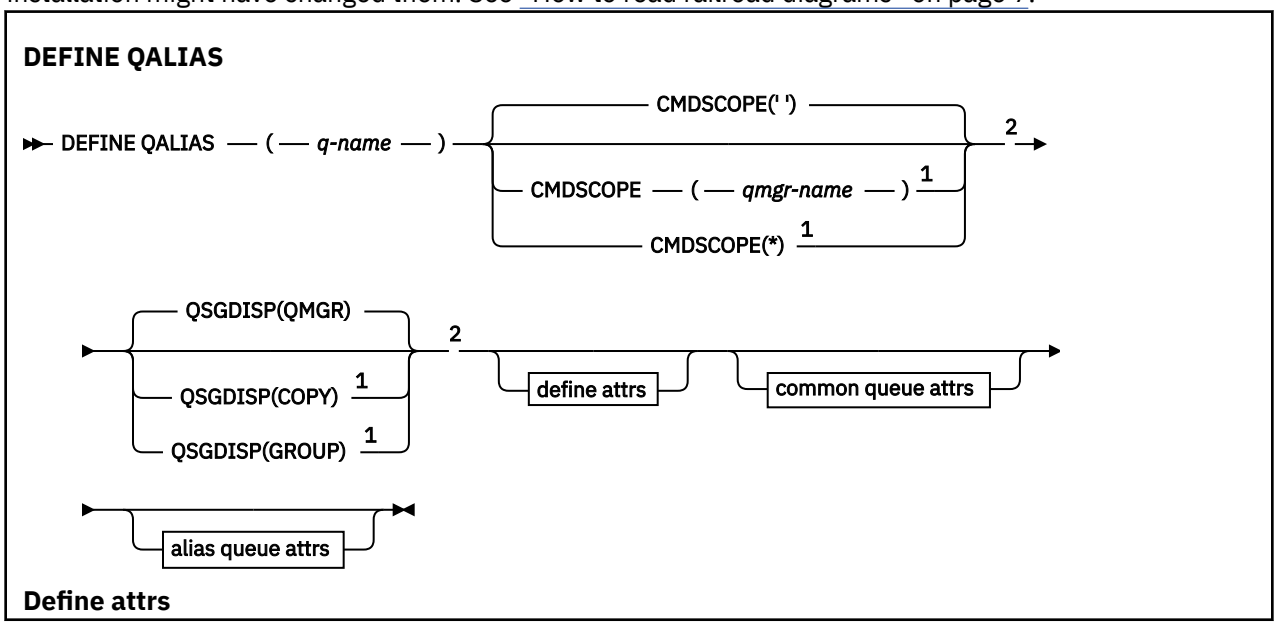
Use DEFINE QALIAS to define a new alias queue, and set its parameters.

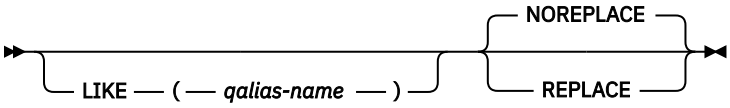
Note: An alias queue provides a level of indirection to another queue or a topic object. If the alias refers to a queue, it must be another local or remote queue, defined at this queue manager, or a clustered alias queue defined on another queue manager. It cannot be another alias queue on this queue 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 913
- “Parameter descriptions for DEFINE QUEUE and ALTER QUEUE” on page 914

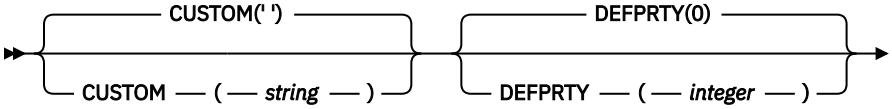
Synonym: DEF QA

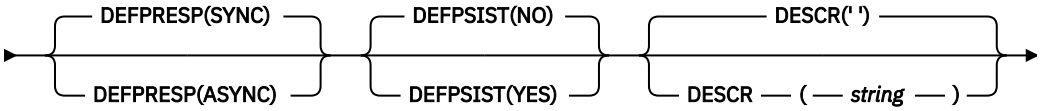
Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See “How to read railroad diagrams” on page 7.

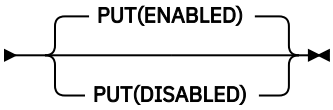




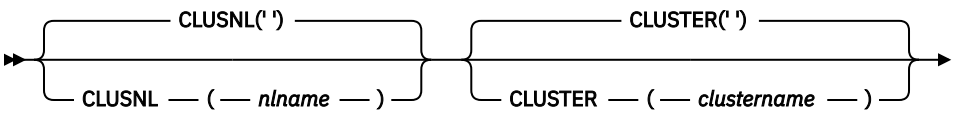
Common queue attrs

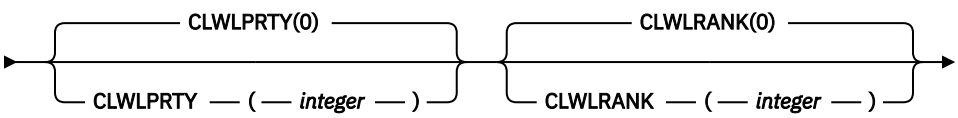


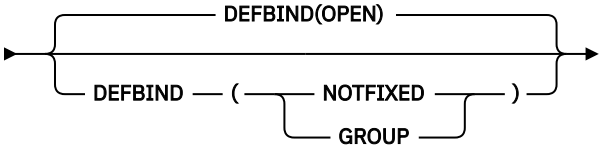


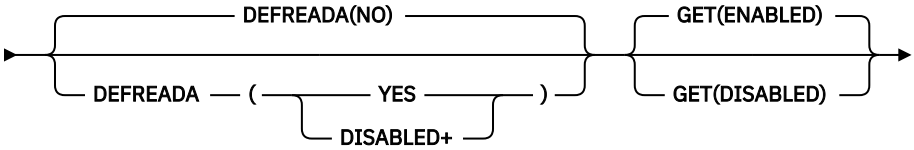


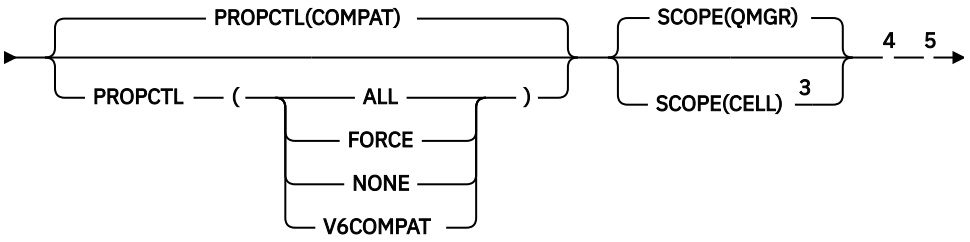
Alias queue attrs

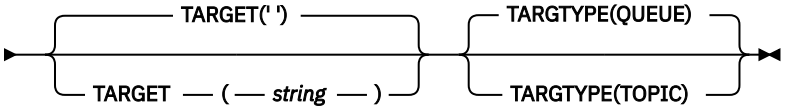












Notes:

[1] Valid only 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 UNIX and Linux systems, and Windows.

[4] Not valid on z/OS.

[5] 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.

DEFINE QLOCAL

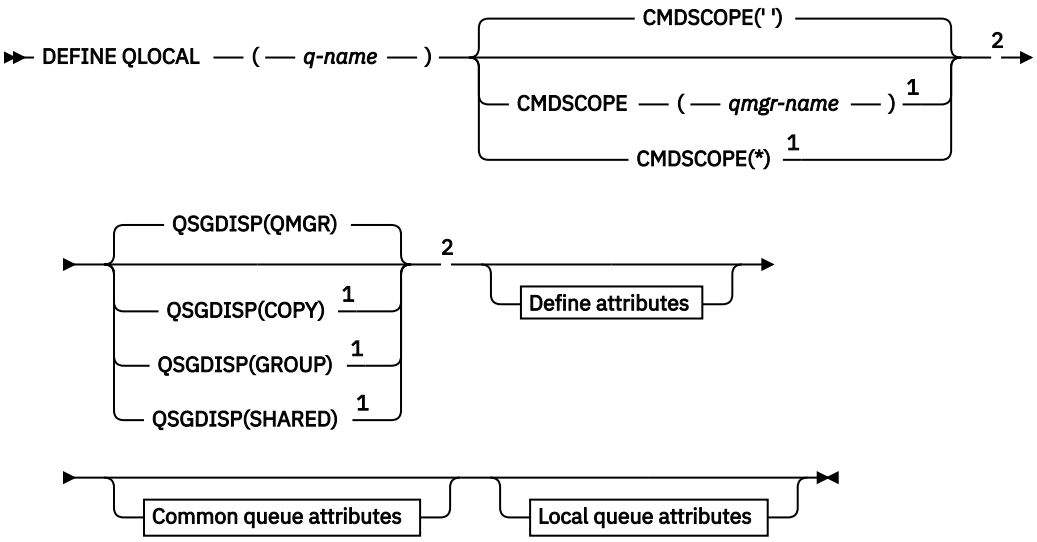
Use **DEFINE QLOCAL** to define a new local queue, and set its parameters.

- Syntax diagram
- "Usage notes for DEFINE queues" on page 913
- "Parameter descriptions for DEFINE QUEUE and ALTER QUEUE" on page 914

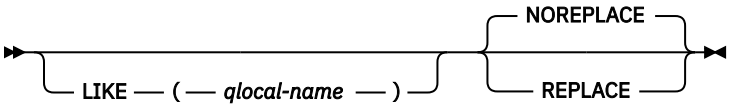
Synonym: **DEF QL**

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

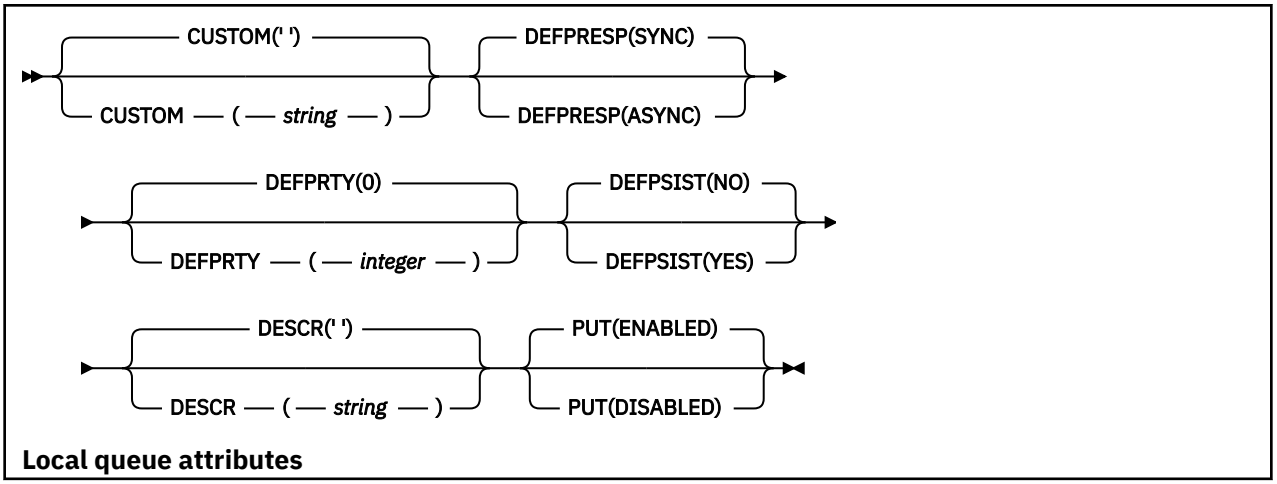
DEFINE QLOCAL



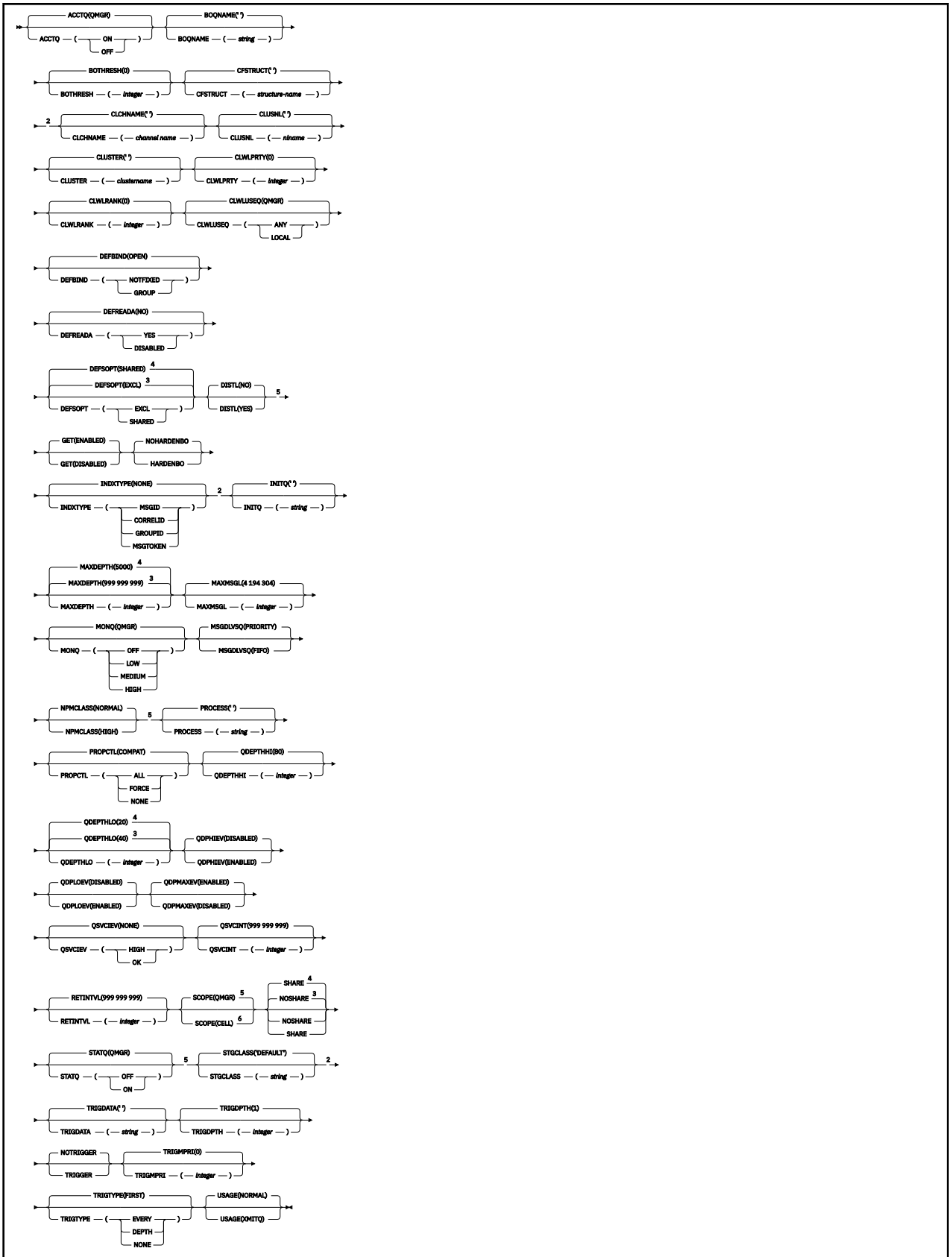
Define attributes



Common queue attributes



Local queue attributes



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.

[3] Default for z/OS.
[4] Default for distributed platforms.
[5] Not valid on z/OS.
[6] Valid only on UNIX, Linux, and Windows systems.

DEFINE QMODEL

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.

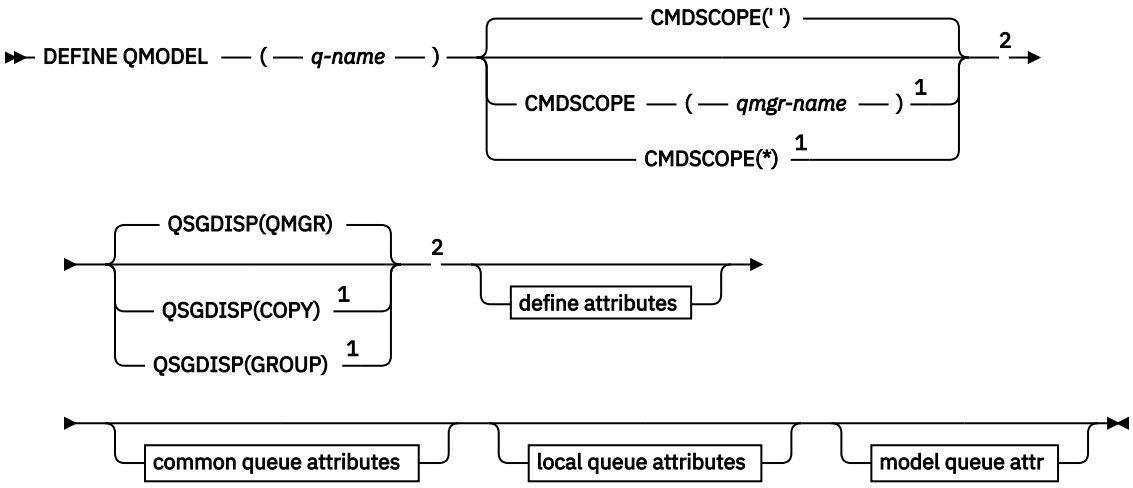
When it has been defined, a model queue (like any other queue) has a complete set of applicable attributes, even if some of these are defaults.

- Syntax diagram
- "Usage notes for DEFINE queues" on page 913
- "Parameter descriptions for DEFINE QUEUE and ALTER QUEUE" on page 914

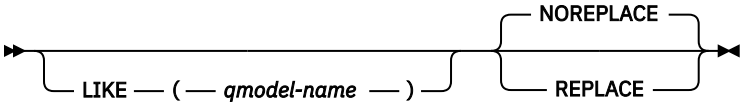
Synonym: **DEF QM**

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

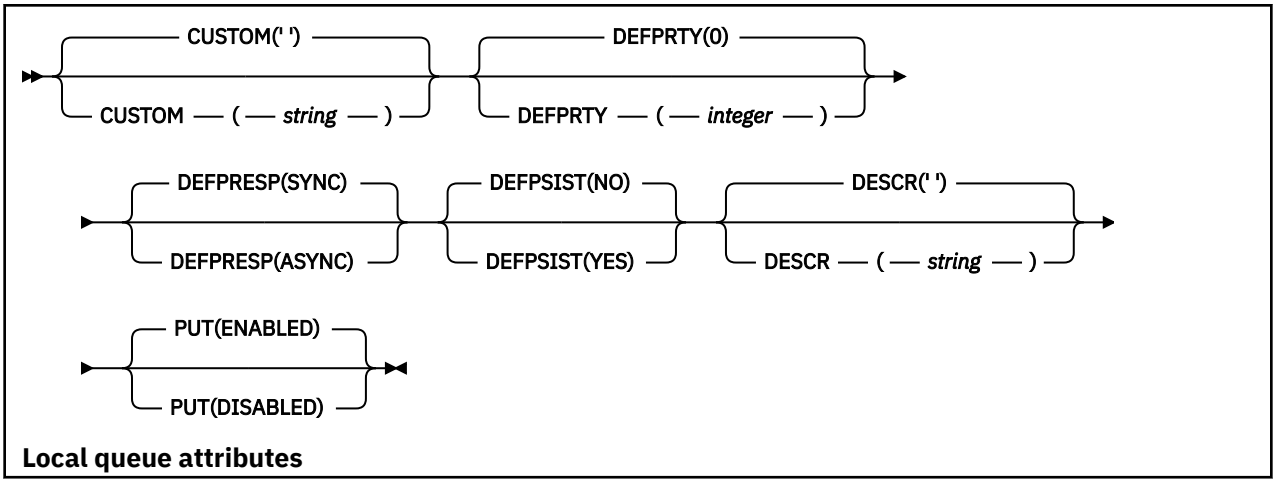
DEFINE QMODEL



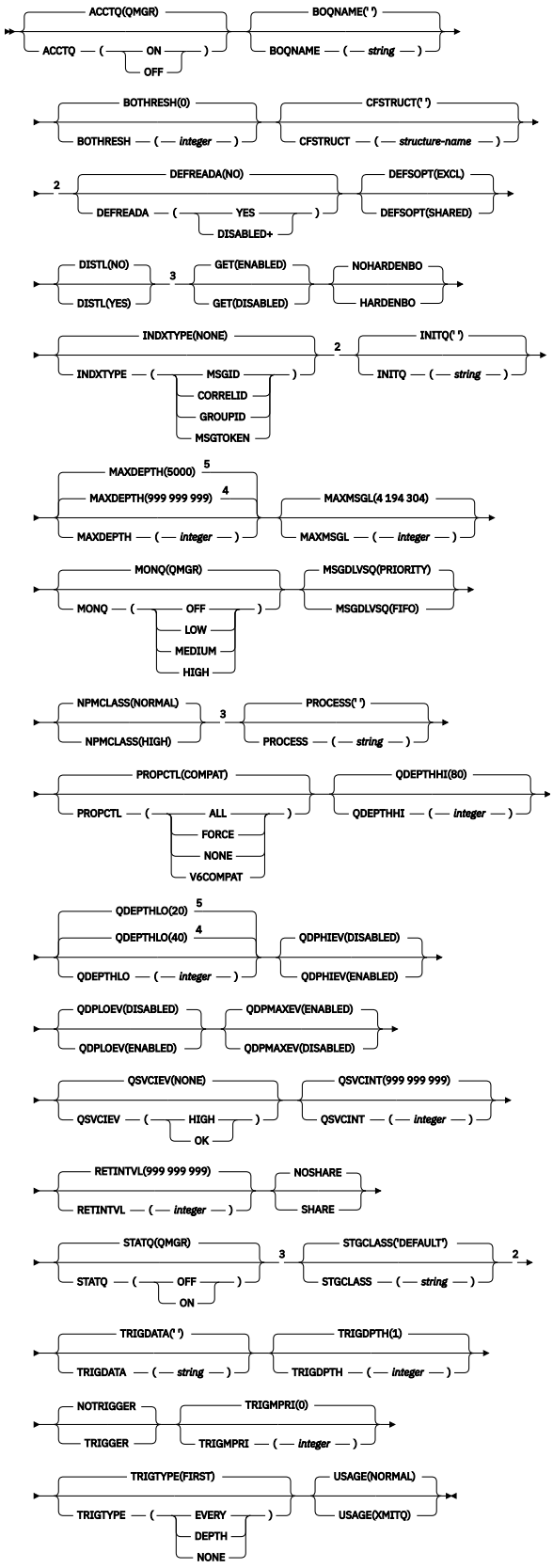
Define attributes



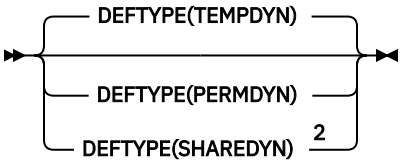
Common queue attributes



Local queue attributes



Model queue attr



Notes:

[1] Valid only on z/OS when the queue manager is a member of a queue-sharing group.
[2] Used only on z/OS.
[3] Not valid on z/OS.
[4] Default for z/OS.
[5] Default for distributed platforms.

DEFINE QREMOTE

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.

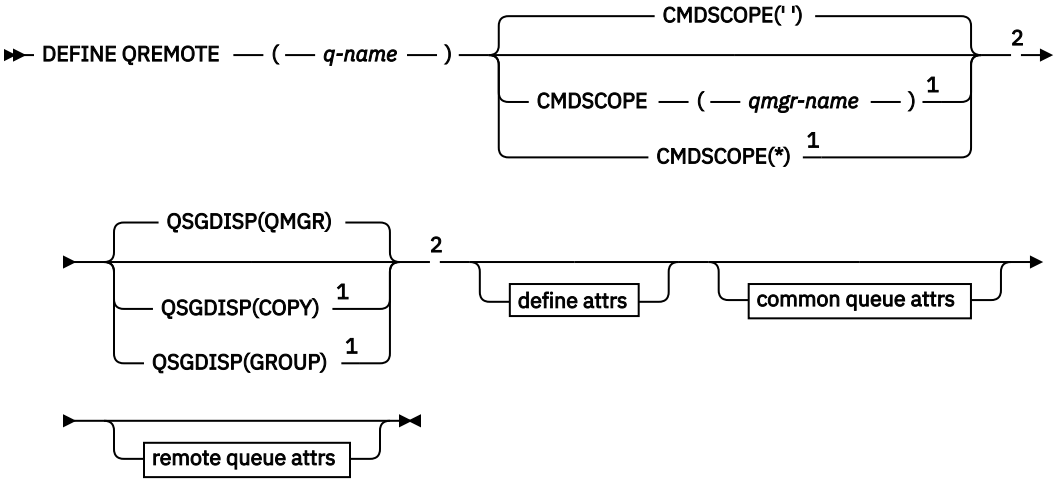
A remote queue is one that is owned by another queue manager that application processes connected to this queue manager need to access.

- Syntax diagram
- "Usage notes for DEFINE queues" on page 913
- "Parameter descriptions for DEFINE QUEUE and ALTER QUEUE" on page 914

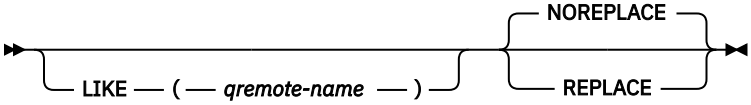
Synonym: DEF QR

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

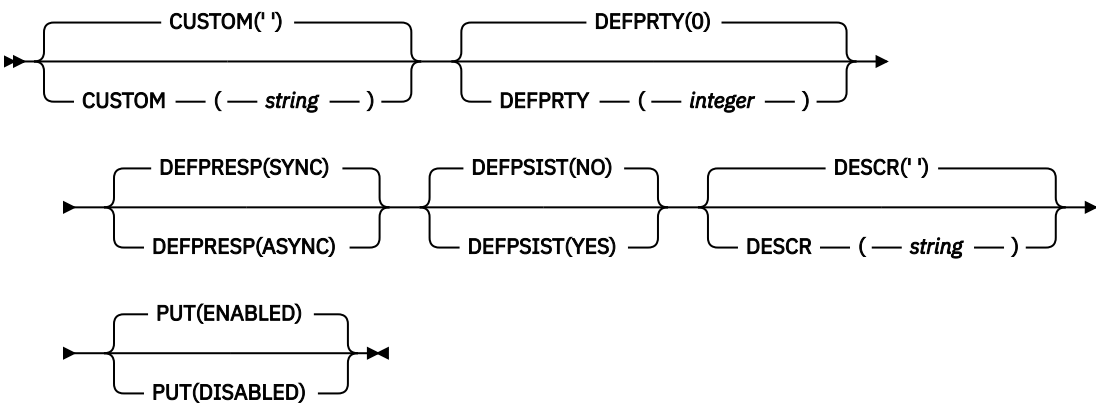
DEFINE QREMOTE



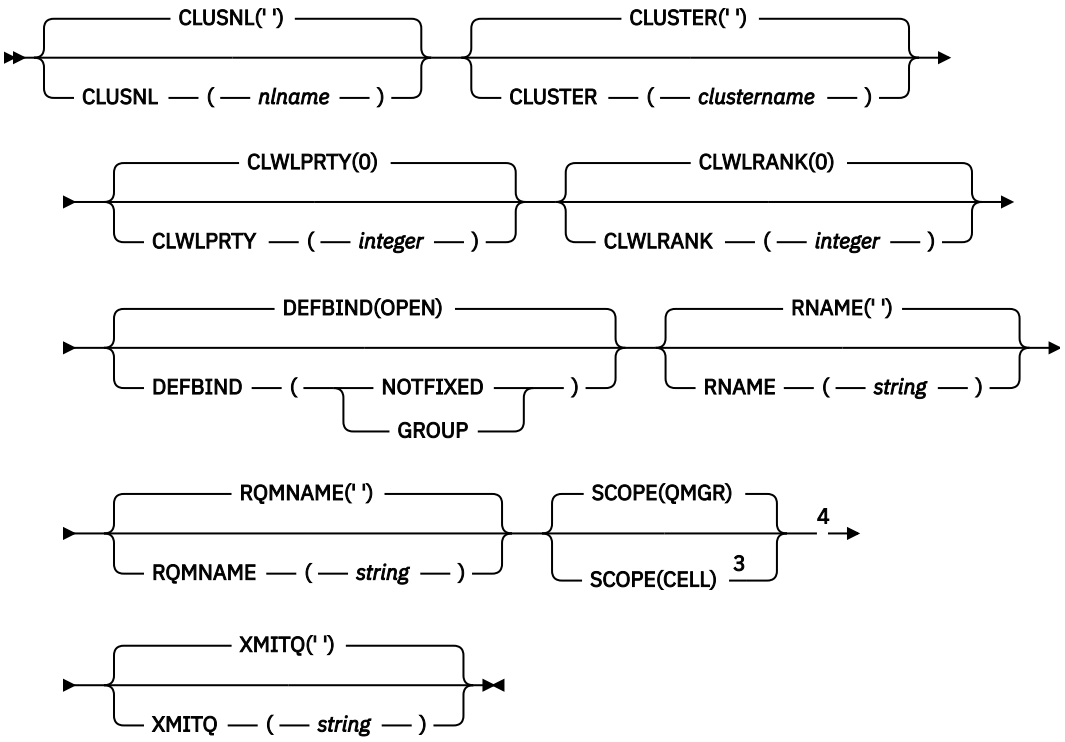
Define attrs



Common queue attrs



Remote queue attrs



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] Valid only on UNIX and Linux systems, and Windows.
[4] Not valid on z/OS.

DEFINE SERVICE

Use the MQSC command DEFINE SERVICE to define a new IBM MQ service definition, and set its parameters.

Using MQSC commands

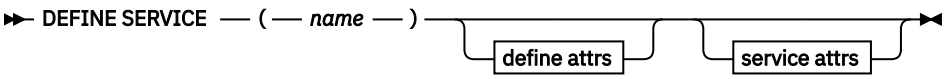
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

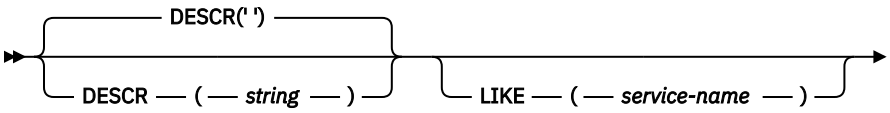
- Syntax diagram
- "Usage notes" on page 946
- "Parameter descriptions for DEFINE SERVICE" on page 947

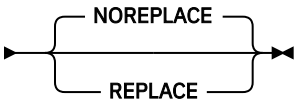
Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.

DEFINE SERVICE

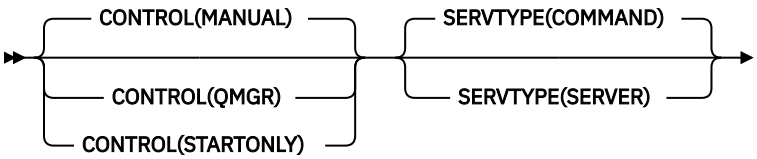


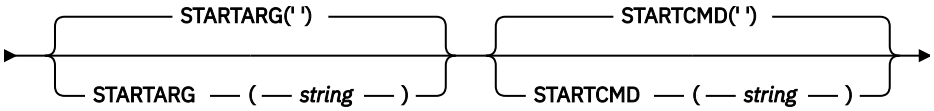
Define attrs

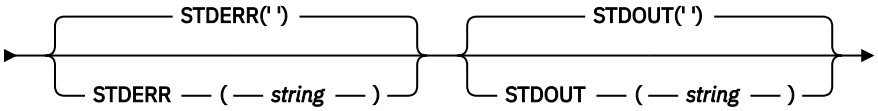


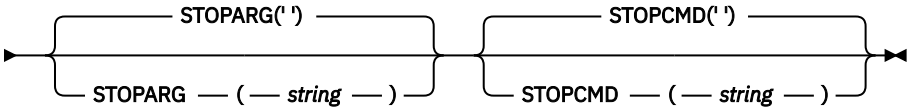


Service attrs









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” on page 1170).

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

z/OS DEFINE STGCLASS

Use the MQSC command DEFINE STGCLASS to define a storage class to page set mapping.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

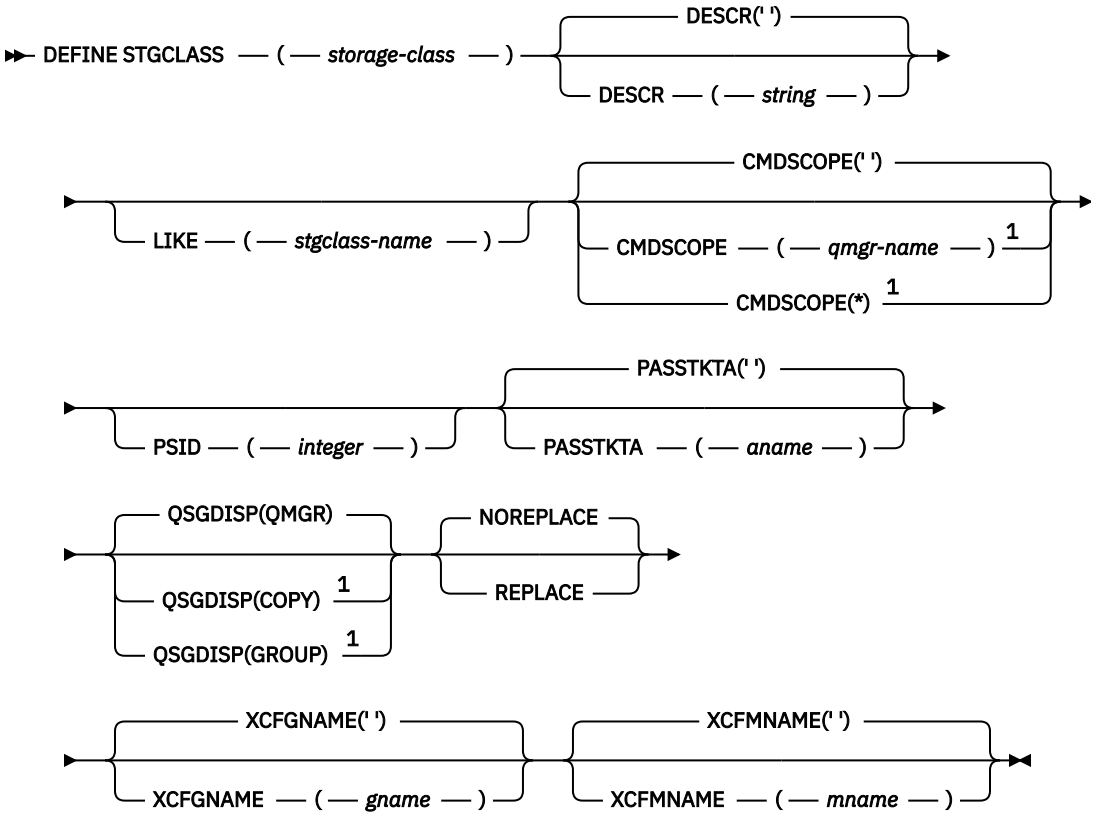
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for DEFINE STGCLASS” on page 949
- “Parameter descriptions for DEFINE STGCLASS” on page 949

Synonym: DEF STC

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See “How to read railroad diagrams” on page 7.

DEFINE STGCLASS



Notes:

[1] Valid only on z/OS when the queue manager is a member of a queue-sharing group.

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: 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 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.

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" on page 911.

QSGDISP
Specifies the disposition of the object in the group.

QSGDISP	DEFINE
COPY	The object is defined on the page set of the queue manager that executes the command using the QSGDISP(GROUP) object of the same name as the 'LIKE' object.

GROUP	The object definition resides in the shared repository but only if the queue manager is in a queue-sharing group. If the definition is successful, the following 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:
	`DEFINE STGCLASS(storage-class)`
	`REPLACE QSGDISP(COPY)`
	The DEFINE for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

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 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

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

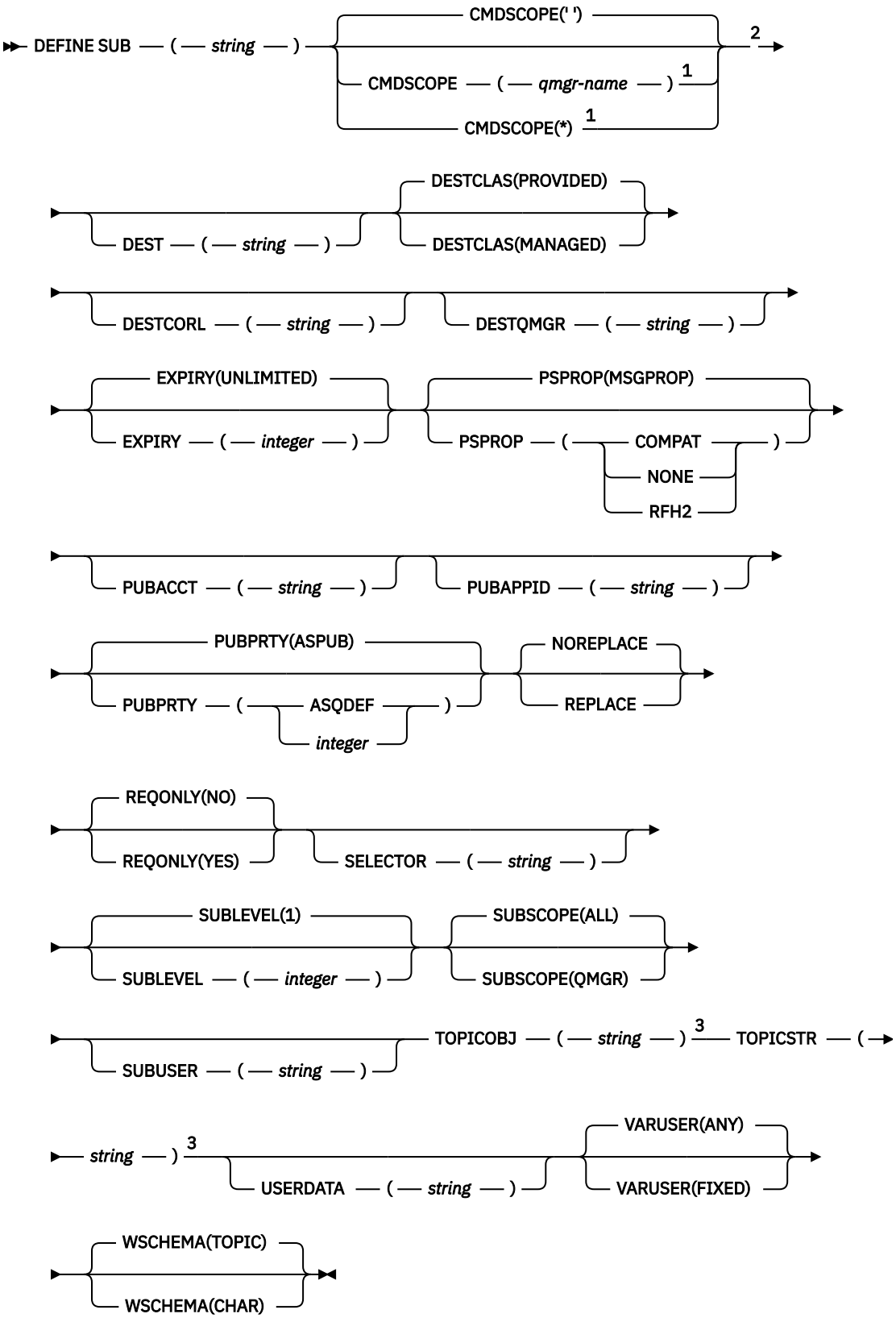
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for DEFINE SUB” on page 954
- “Parameter descriptions for DEFINE SUB” on page 954

Synonym: DEF SUB

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See “How to read railroad diagrams” on page 7.

DEFINE SUB



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] 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.
- 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, WSCHEMA, SELECTOR, SUBSCOPE, 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.

z/OS **CMDSCOPE**
This parameter applies to z/OS only and specifies how the command is processed when the queue manager is a member of a queue-sharing group.

' '
The command is processed on the queue manager on which it was entered. This is the default value.

qmgr-name
The command is processed 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 processed 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.

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 ' `00000000000000000000000000000000000000000000000000000000` ' (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.

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. You must define the channels to the remote queue manager, for example, the XMITQ, and a sender channel. If you do not, messages do not arrive at the destination.

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.

LIKE(*subscription-name*)

The name of a subscription, the parameters of which are used as a model for this definition.

This parameter applies only to the DEFINE SUB command.

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.

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.

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, WSCHEMA, 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.

V 8.0.0.7 From IBM MQ 8.0.0, Fix Pack 7, 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` in the JmsConstants interface with the method `javax.jms.Message.getStringProperty(java.lang.String)`. 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.

WSCHEMA
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.

DEFINE 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 Performing local administration tasks using MQSC commands.

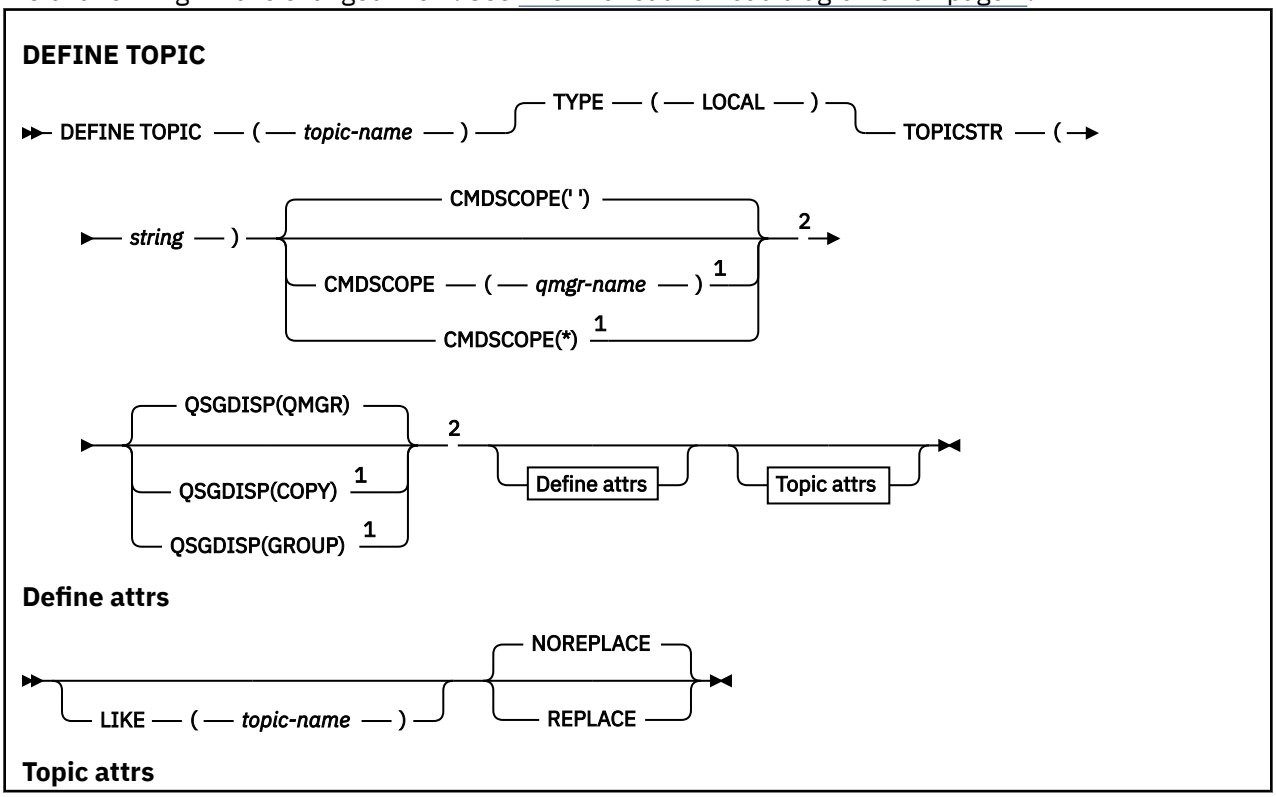
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

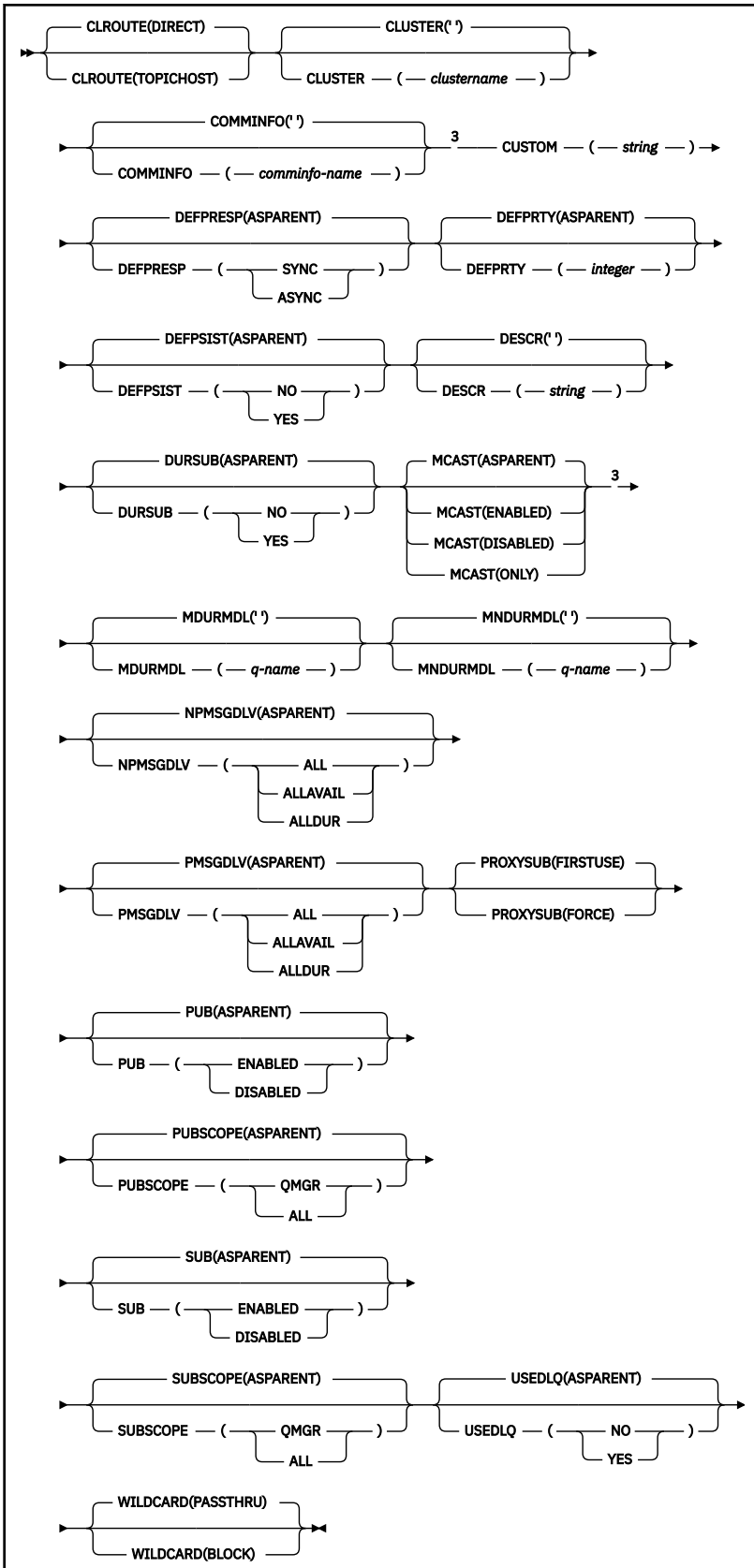
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for DEFINE TOPIC" on page 960
- "Parameter descriptions for DEFINE TOPIC" on page 960

Synonym: DEF TOPIC

Values shown above the main line in the railroad diagram are the defaults supplied with IBM MQ, but your installation might have changed them. See "How to read railroad diagrams" on page 7.





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.

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).

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 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.

COMMINFO(*comminfo-name*)
The name of the Multicast communication information object associated with this topic object.

CUSTOM (*string*)
The custom attribute for new features.

V 8.0.0.4 This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name and value pairs have the form `NAME(VALUE)`. Single quotation marks must be escaped with another single quotation mark.

CAPEXPRY (*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.

Specifying a value for CAPEXPRY 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.

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.

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.

Note: HPE NonStop V8.1 The MCAST (ENABLED) and MCAST (ONLY) options are not available on IBM MQ for HPE NonStop V8.1.0.

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 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.

QSGDISP	DEFINE
COPY	The object is defined on the page set of the queue manager that executes the command using the QSGDISP(GROUP) object of the same name as the 'LIKE' object.

QSGDISP	DEFINE

GROUP	The object definition resides in the shared repository but only if the queue manager is in a queue-sharing group. If the definition is successful, the following 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:
	`DEFINE TOPIC(name)`
	`REPLACE QSGDISP(COPY)`
	The DEFINE for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

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 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.

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.

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 NPMSGDLV 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.

DELETE AUTHINFO

Use MQSC command DELETE AUTHINFO to delete an authentication information object.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

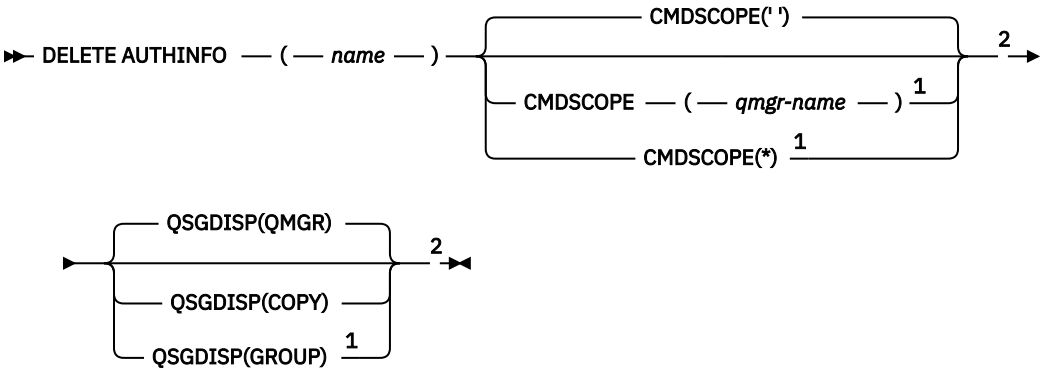
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for DELETE AUTHINFO” on page 968

Synonym: None

DELETE AUTHINFO



Notes:

1 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.
2 Valid only on z/OS.

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 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. 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**
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.

DELETE AUTHREC

Use the MQSC command DELETE AUTHREC to delete authority records associated with a profile name.

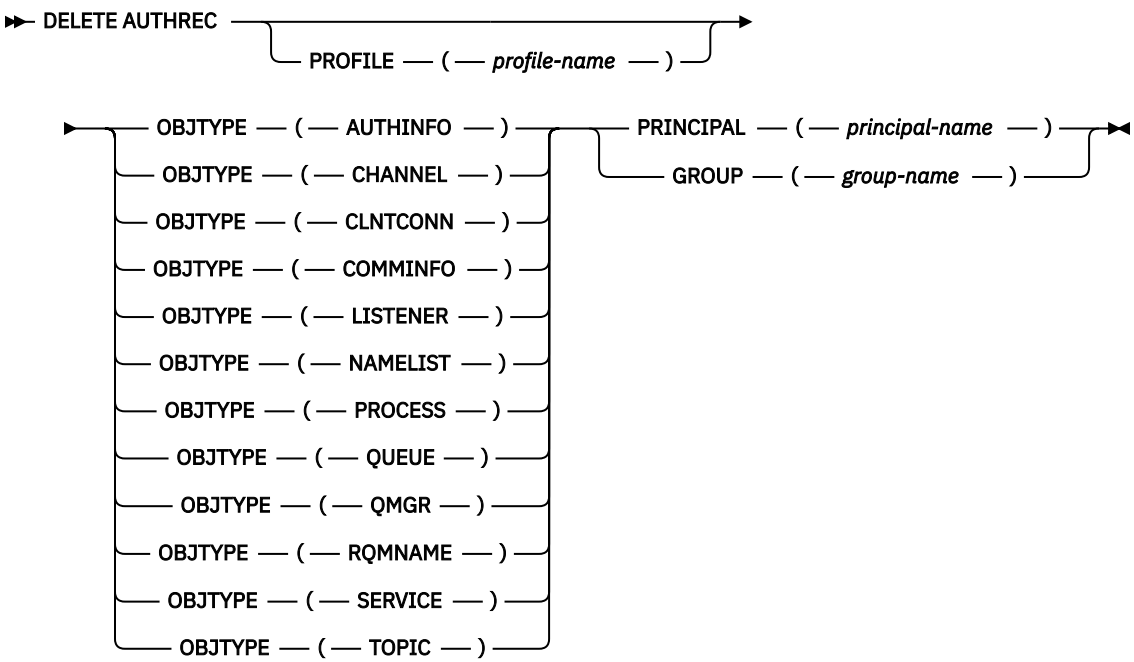
Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- "Parameter descriptions" on page 970

DELETE AUTHREC



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.

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.

z/OS DELETE BUFFPOOL

Use the MQSC command DELETE BUFFPOOL to delete a buffer pool that is used for holding messages in main storage.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

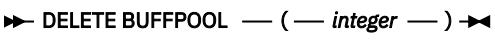
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage note for DELETE BUFFPOOL” on page 971
- “Parameter descriptions for DELETE BUFFPOOL” on page 971

Synonym: DEL BP

DELETE BUFFPOOL



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 15. If OPMODE is set to OPMODE=(NEWFUNC, 800), this parameter is an integer in the range zero through 99.

z/OS DELETE CFSTRUCT

Use the MQSC command DELETE CFSTRUCT to delete a CF application structure definition.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

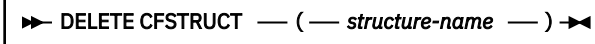
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for DELETE CFSTRUCT" on page 972
- "Keyword and parameter descriptions for DELETE CFSTRUCT" on page 972

Synonym: None

DELETE CFSTRUCT



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

Use the MQSC command DELETE CHANNEL to delete a channel definition.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

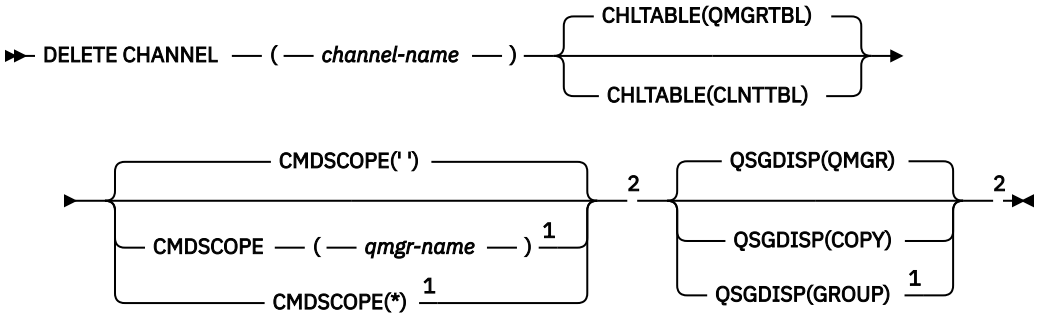
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 973
- "Parameter descriptions" on page 973

Synonym: DELETE CHL

DELETE CHANNEL



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.

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 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. 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

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.

DELETE CHANNEL (MQTT)

Use the MQSC command DELETE CHANNEL to delete an IBM MQ Telemetry channel definition.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

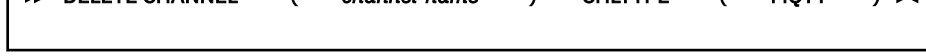
IBM i	AIX and Linux	Windows	z/OS
	✓	✓	

Note: For the telemetry server, AIX is the only supported UNIX platform.

The DELETE CHANNEL (MQTT) command is only valid for IBM MQ Telemetry channels.

Synonym: DELETE CHL

DELETE CHANNEL



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.

DELETE COMMINFO

Use the MQSC command DELETE COMMINFO to delete a communication information object.

Using MQSC commands

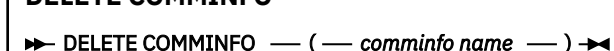
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Parameter descriptions for DELETE COMMINFO” on page 975

Synonym: DEL COMMINFO

DELETE COMMINFO



Parameter descriptions for DELETE COMMINFO

(comminfo name)
The name of the communications information object to be deleted. This is required.

DELETE LISTENER

Use the MQSC command DELETE LISTENER to delete a listener definition.

Using MQSC commands

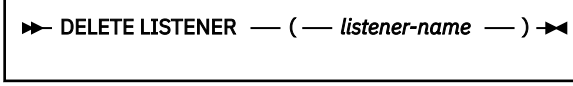
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- "Usage notes for DELETE LISTENER" on page 976
- "Keyword and parameter descriptions for DELETE LISTENER" on page 976

Synonym: DELETE LSTR

DELETE LISTENER



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 is required. The name must be that of an existing listener defined on the local queue manager.

DELETE NAMELIST

Use the MQSC command DELETE NAMELIST to delete a namelist definition.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

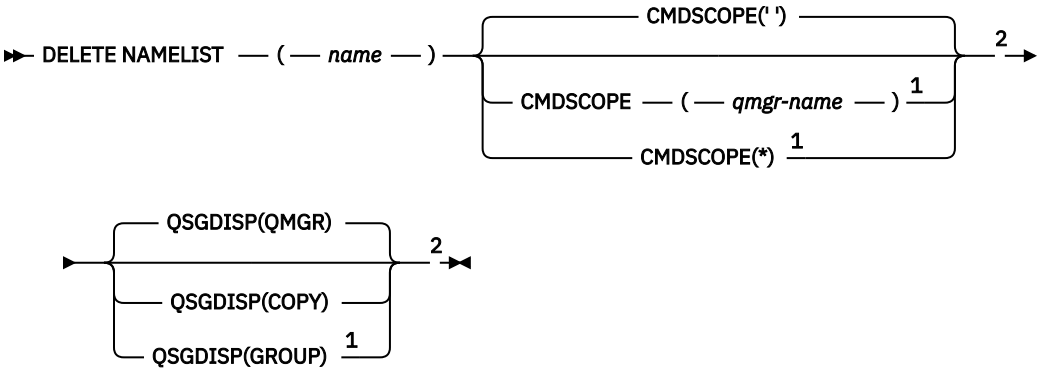
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 977
- "Parameter descriptions for DELETE NAMELIST" on page 977

Synonym: DELETE NL

DELETE NAMELIST



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.

Usage notes

- On UNIX systems, the command is valid only on AIX, HP-UX, Linux and Solaris.
- 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.

z/OS **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. 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

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.

DELETE POLICY

Use the MQSC command DELETE POLICY to delete a security policy.

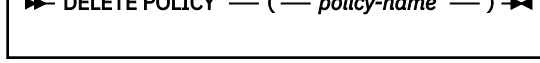
Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Parameter descriptions for DELETE POLICY” on page 978

DELETE POLICY



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.

DELETE PROCESS

Use the MQSC command DELETE PROCESS to delete a process definition.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

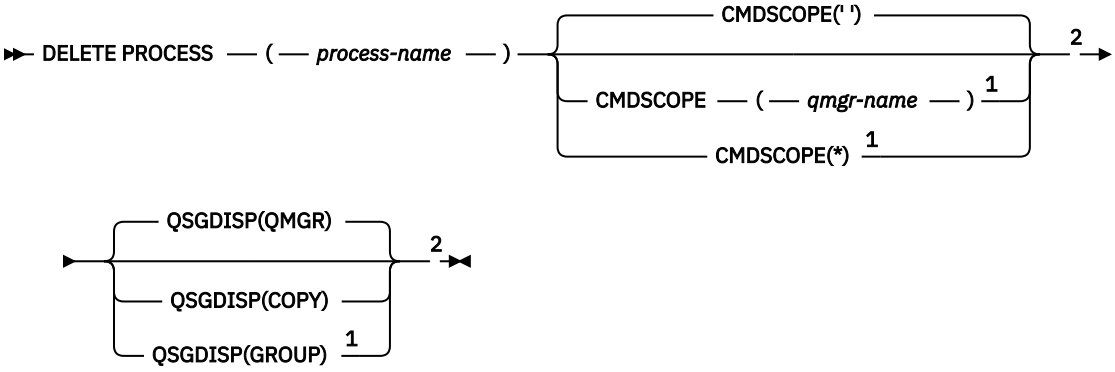
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for DELETE PROCESS” on page 979

Synonym: DELETE PRO

DELETE PROCESS



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.

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 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. 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**

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.

z/OS DELETE PSID

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

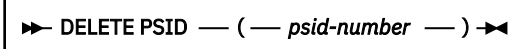
IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for DELETE PSID” on page 981
- “Parameter descriptions for DELETE PSID” on page 981

Synonym: DEL PSID

DELETE PSID



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

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

This section contains the following commands:

- “DELETE QALIAS” on page 983
- “DELETE QLOCAL” on page 984
- “DELETE QMODEL” on page 985
- “DELETE QREMOTE” on page 985

These commands are supported on the following platforms:

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on 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.

z/OS **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 or SHARED.

' '
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.

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.

z/OS **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.

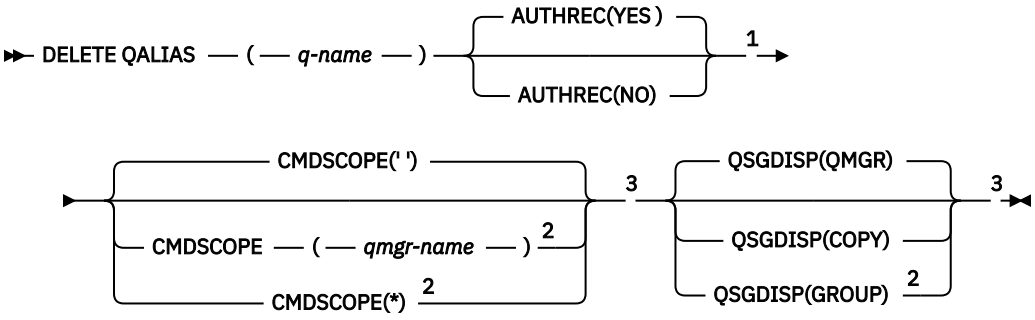
The object definition resides in the shared repository. The object was defined using a command that had the parameters QSGDISP(SHARED). Any object residing on the page set of the queue manager that executes the command, or any object defined using a command that had the parameters QSGDISP(GROUP), is not affected by this command.

DELETE QALIAS

Use DELETE QALIAS to delete an alias queue definition.

Synonym: DELETE QA

DELETE QALIAS



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.

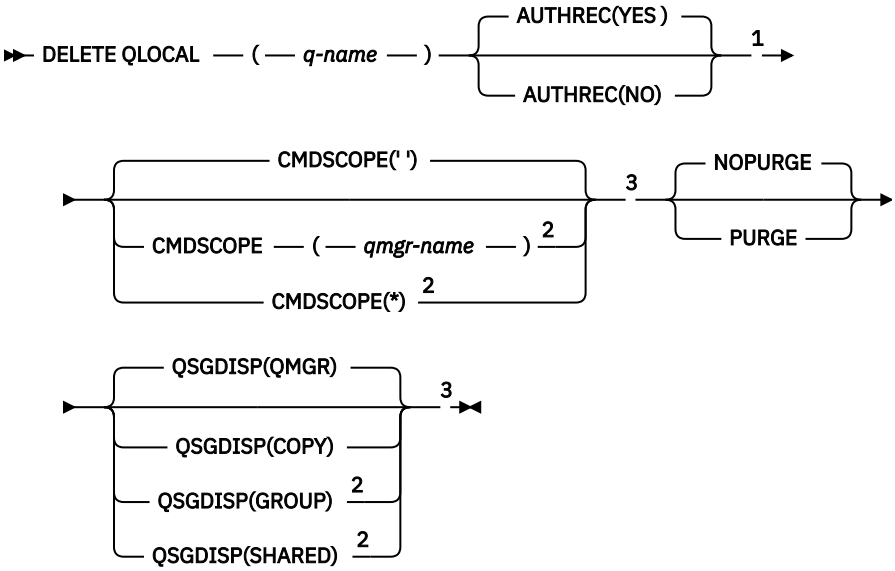
The parameters are described in "DELETE queues" on page 981.

DELETE QLOCAL

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.

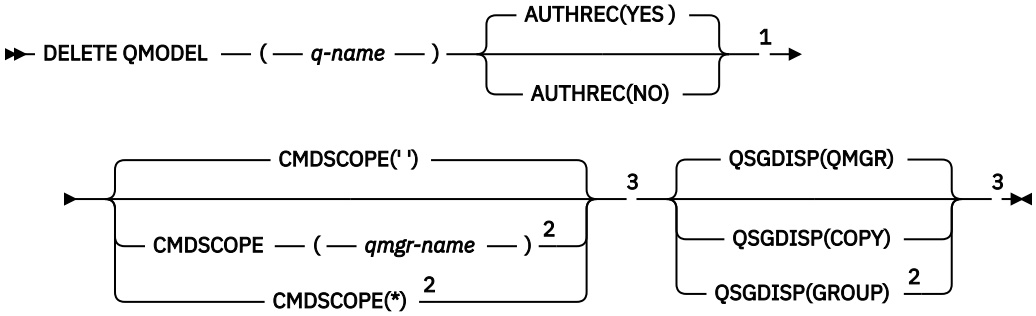
The parameters are described in "DELETE queues" on page 981.

DELETE QMODEL

Use DELETE QMODEL to delete a model queue definition.

Synonym: DELETE QM

DELETE QMODEL



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.

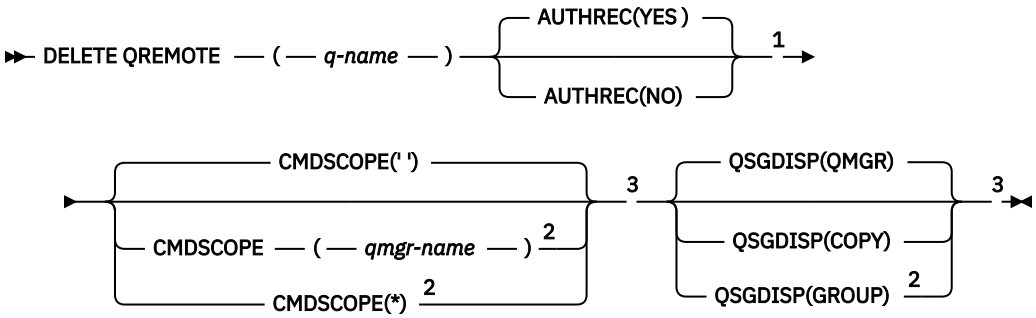
The parameters are described in "DELETE queues" on page 981.

DELETE QREMOTE

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:

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.

The parameters are described in "DELETE queues" on page 981.

DELETE SERVICE

Use the MQSC command DELETE SERVICE to delete a service definition.

Using MQSC commands

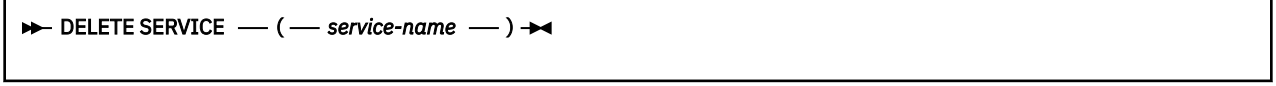
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Usage notes for DELETE SERVICE” on page 986
- “Keyword and parameter descriptions for DELETE SERVICE” on page 986

Synonym:

DELETE SERVICE



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 is required. The name must be that of an existing service defined on the local queue manager.

DELETE SUB

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 Performing local administration tasks using MQSC commands.

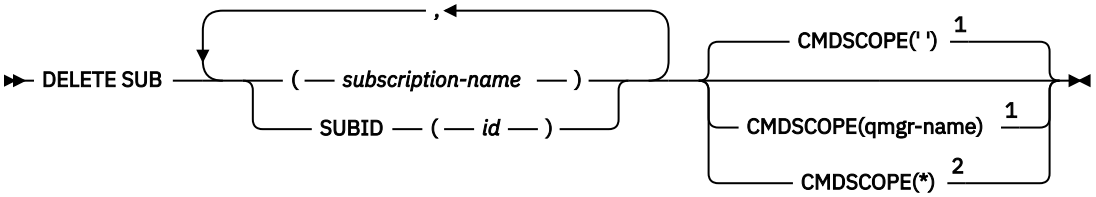
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- Usage notes for DELETE SUB
- “Parameter descriptions for DELETE SUB” on page 987

Synonym: DEL SUB

DELETE SUB



Notes:

[1] Valid only on z/OS.
[2] Valid only on z/OS when the queue manager is a member of a queue-sharing group.

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.

z/OS **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 processed on the queue manager on which it was entered. This is the default value.

qmgr-name
The command is processed 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 processed 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.

z/OS DELETE STGCLASS

Use the MQSC command DELETE STGCLASS to delete a storage class definition.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

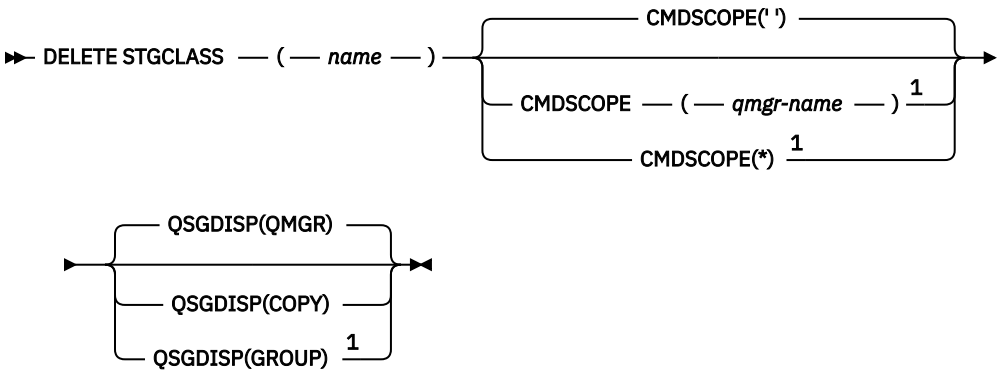
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for DELETE STGCLASS” on page 988

Synonym: DELETE STC

DELETE STGCLASS



Notes:

1 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 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. 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.

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 TOPIC

Use DELETE TOPIC to delete an IBM MQ administrative topic node.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

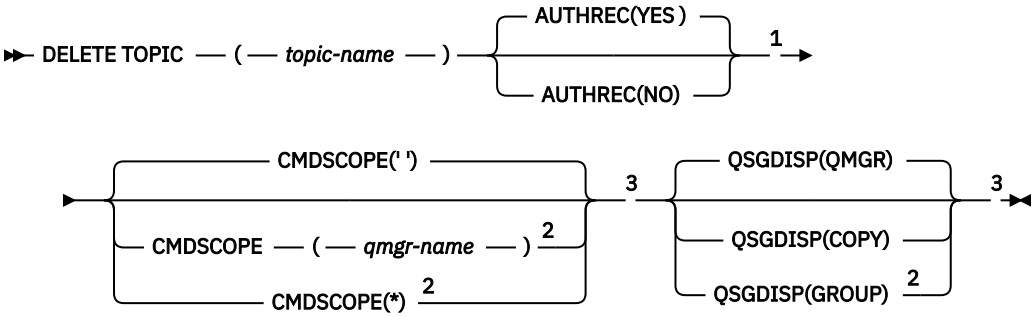
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram

- "Usage notes for DELETE TOPIC" on page 990
- "Parameter descriptions for DELETE TOPIC" on page 990

Synonym: None

DELETE TOPIC



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.

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 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. 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**

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.

z/OS DISPLAY ARCHIVE

Use the MQSC command DISPLAY ARCHIVE to display archive system parameters and information.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

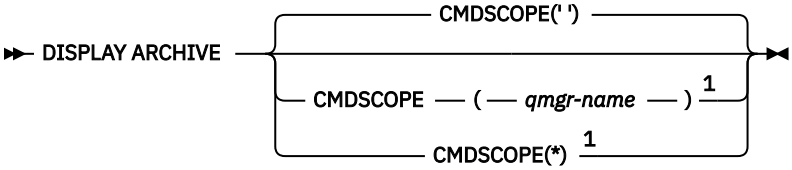
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for DISPLAY ARCHIVE” on page 992
- “Parameter descriptions for DISPLAY ARCHIVE” on page 992

Synonym: DIS ARC

DISPLAY ARCHIVE



Notes:

[1] 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” on page 1263.
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 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 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.

DISPLAY AUTHINFO

Use the 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 Performing local administration tasks using MQSC commands.

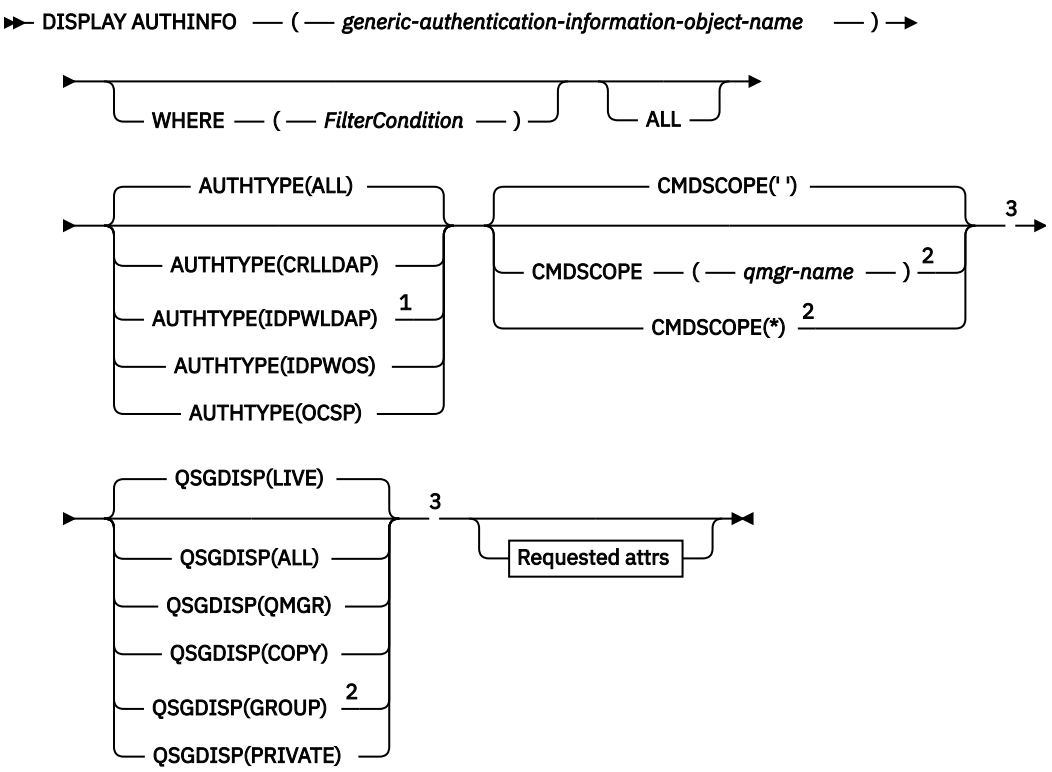
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

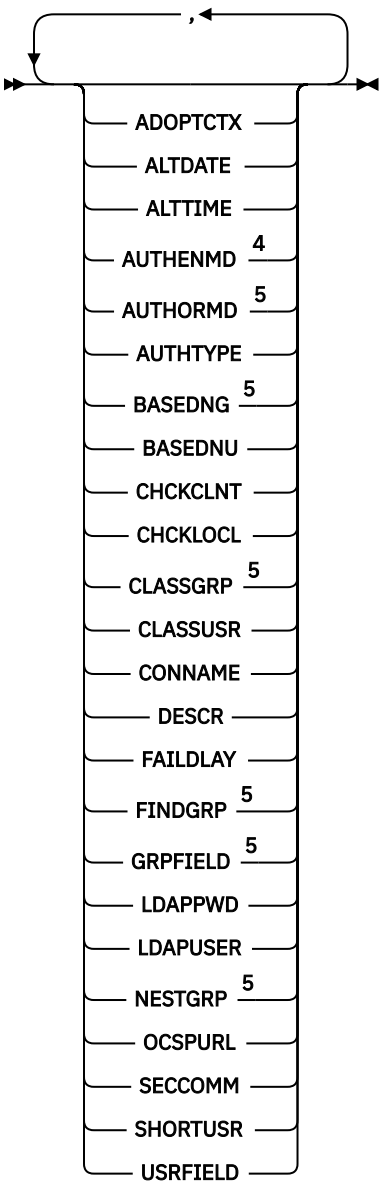
- Syntax diagram
- “Parameter descriptions for DISPLAY AUTHINFO” on page 995
- “Requested parameters” on page 998

Synonym: DIS AUTHINFO

DISPLAY AUTHINFO



Requested attrs



Notes:

[1] Not valid on IBM MQ for z/OS.
[2] 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.
[3] Valid only on z/OS.
[4] Not valid on z/OS and AUTHENMD PAM value valid only on UNIX platforms.
[5] Not valid on Windows.

Parameter descriptions for DISPLAY AUTHINFO

(generic-authentication-information-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 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. 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.

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).

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
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`

ALTTIME
The time at which the definition was last altered, in the form `hh.mm.ss`

V 8.0.0.3 **AUTHENMD**
Authentication method. Possible values are:

OS
Displays the traditional UNIX platforms password verification method permissions.

PAM
Displays the Pluggable Authentication Method permissions.

You can set the PAM value only on UNIX and Linux platforms.

V 8.0.0.2 **AUTHORMD**
Authorization method. Possible values are:

OS
Displays the operating system groups determining permissions

SEARCHGRP
Displays a group entry in the LDAP server, containing an attribute listing all the users belonging to that group.

SEARCHUSR
Displays a user entry in the LDAP server containing an attribute listing all the groups to which the user belongs.

This attribute is not applicable to Windows platforms.

AUTHTYPE
The type of the authentication information

V 8.0.0.2 **BASEDNG**
Displays the Base DN for groups.

This attribute is not applicable to Windows platforms.

BASEDNU
Displays the base distinguished name to search for users within the LDAP server.

CHCKLOCL or CHCKCLNT
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. z/OS (This setting is not allowed on z/OS systems.)

V 8.0.0.2 **CLASSGRP**
Displays the LDAP object class for group records.

This attribute is not applicable to Windows platforms.

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.

V 8.0.0.2 **FINDGRP**
Displays the name of the attribute within an LDAP entry to determine group membership.

This attribute is not applicable to Windows platforms.

V 8.0.0.2 **GRPFIELD**
Displays the LDAP attribute that represents a simple name for the group.

This attribute is not applicable to Windows platforms.

LDAPPWD
Password associated with the Distinguished Name of the user on the LDAP server. If nonblank, this is displayed as asterisks z/OS 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).

V 8.0.0.2 **NESTGRP**
Displays whether a group is a member of another group..

This attribute is not applicable to Windows platforms.

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 819 for more information about individual parameters.

DISPLAY AUTHREC

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

Using MQSC commands

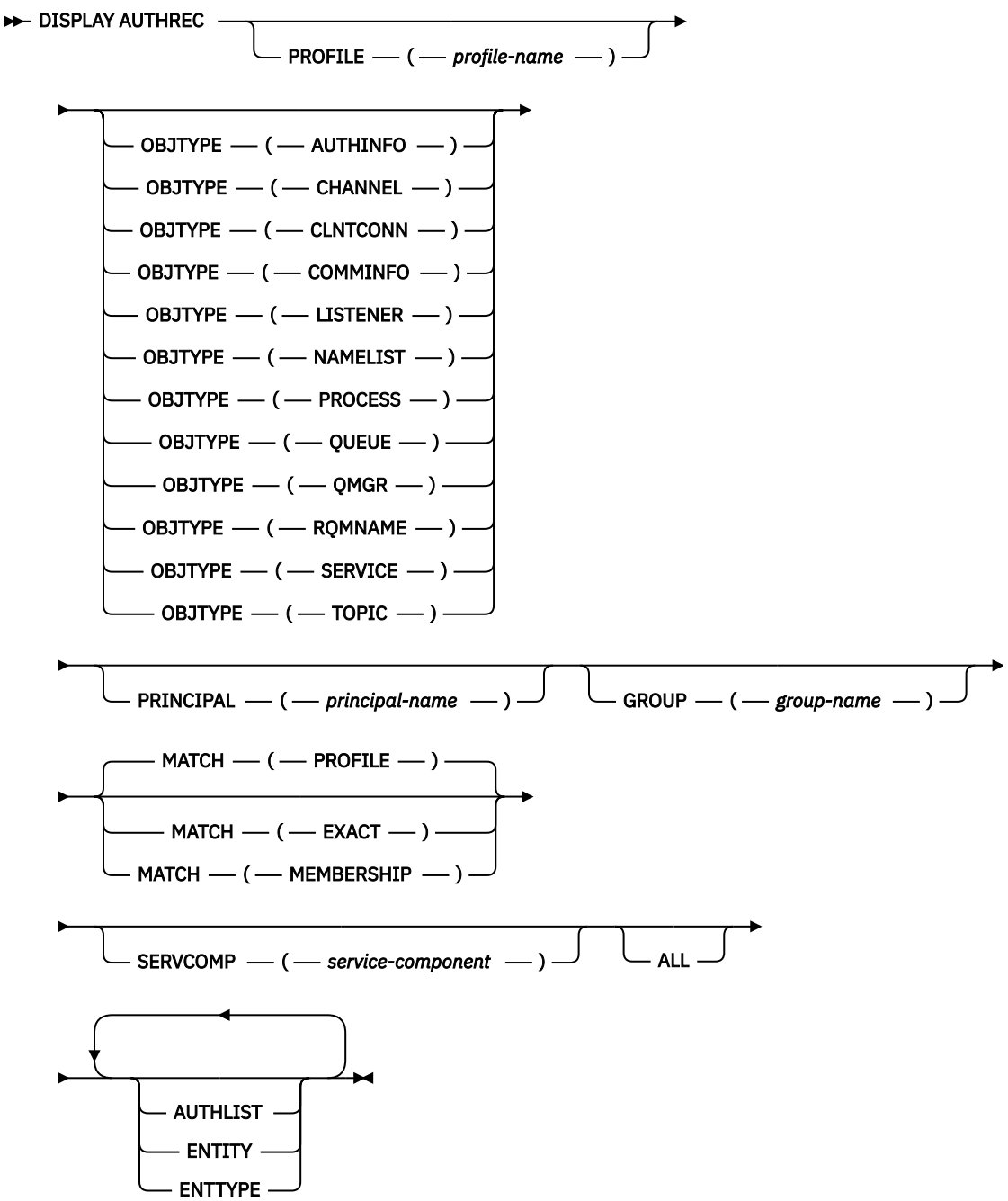
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- "Parameter descriptions" on page 1001
- "Requested parameters" on page 1003

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.

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.

DISPLAY AUTHSERV

Use the MQSC command DISPLAY AUTHSERV to display information about the level of function supported by the installed authorization services.

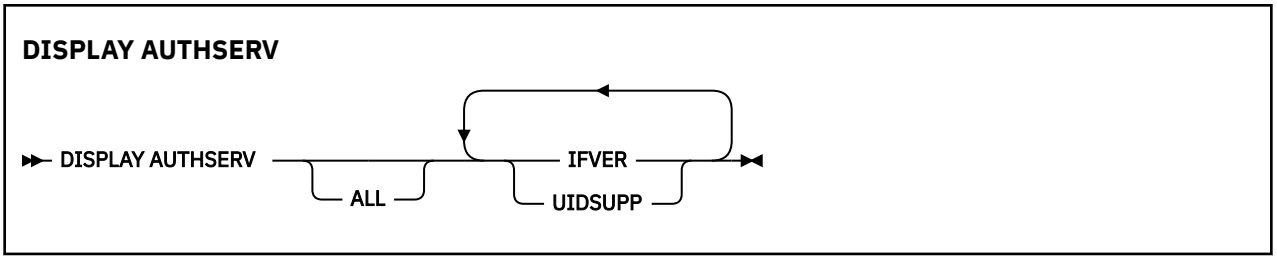
Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	

- Syntax diagram
- “Parameter descriptions” on page 1004
- “Requested parameters” on page 1004

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.

z/OS DISPLAY CFSTATUS

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

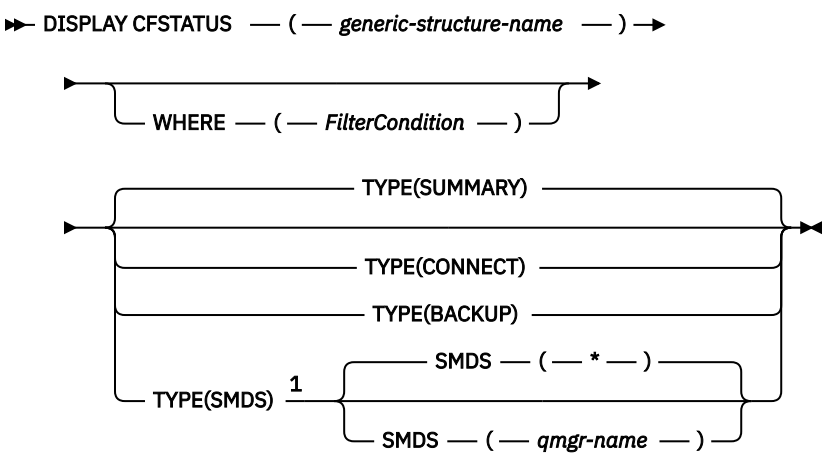
IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Keyword and parameter descriptions for DISPLAY CFSTATUS” on page 1005
- “Summary status” on page 1007
- “Connection status” on page 1008
- “Backup status” on page 1009
- “SMDS status” on page 1010

Synonym: DIS CFSTATUS

DISPLAY CFSTATUS



Notes:

1 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 remains 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" on page 1252.

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().

z/OS DISPLAY CFSTRUCT

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

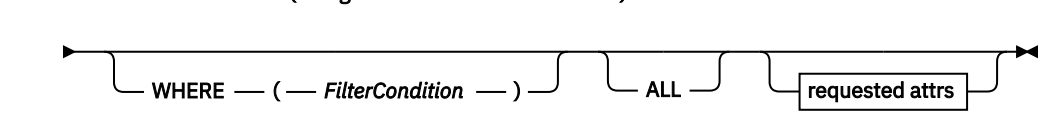
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram

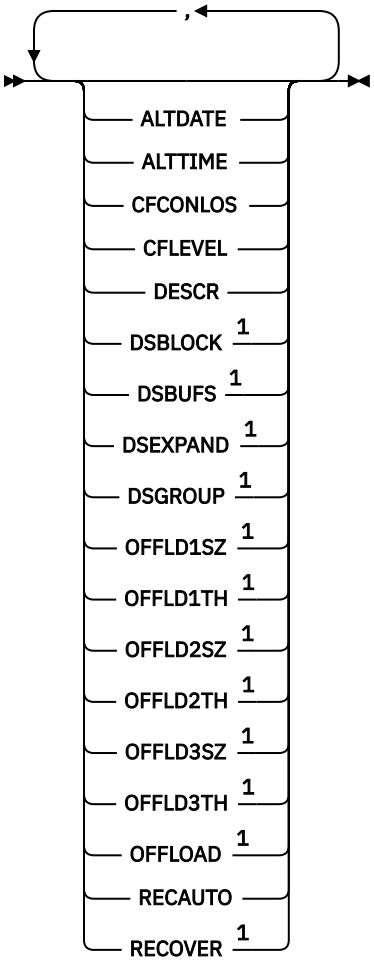
- "Usage notes for DISPLAY CFSTRUCT" on page 1012
- "Keyword and parameter descriptions for DISPLAY CFSTRUCT" on page 1012
- "Requested parameters" on page 1014

Synonym: DIS CFSTRUCT

DISPLAY CFSTRUCT



Requested attrs



Notes:

[1] 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

Use the MQSC command DISPLAY CHANNEL to display a channel definition.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

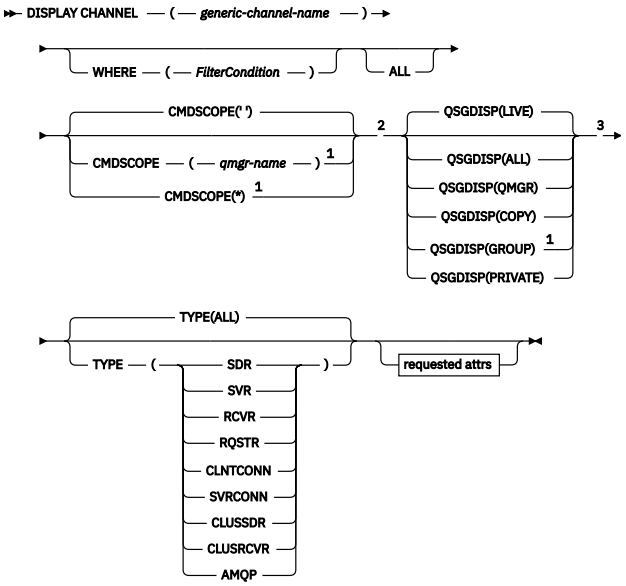
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

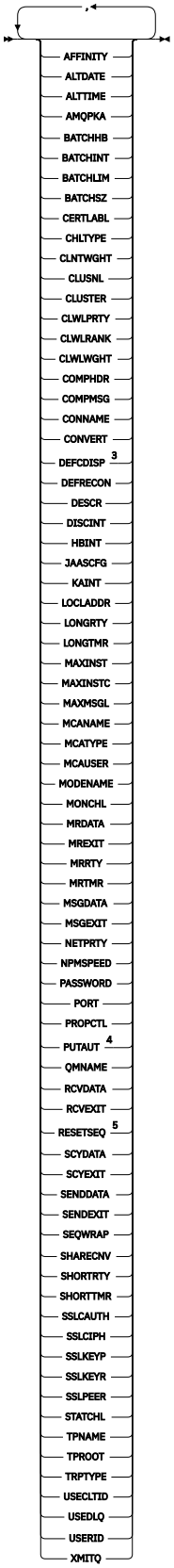
- Syntax diagram
- "Usage notes" on page 1018
- "Parameter descriptions for DISPLAY CHANNEL" on page 1018
- "Requested parameters" on page 1021

Synonym: DIS CHL

DISPLAY CHANNEL



Requested attrs



Notes:

[1] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.

[2] Not valid for z/OS client-connection channels.

[3] Valid only on z/OS.
[4] Valid only for RCVR, RQSTR, CLUSRCVR and (for z/OS only) SVRCONN channel types.
[5] 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.

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.

If no parameters are specified (and the ALL parameter is not specified or defaulted), the default is that the channel names only are displayed.

z/OS On z/OS, the CHLTYPE and QSGDISP values are also displayed.

z/OS **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.).

V 8.0.0.4 **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.
V 8.0.0.4

Table 69. Parameters that result in data being returned from the DISPLAY CHANNEL command

Parameter	SDR	SVR	RCVR	RQSTR	CLNT- CONN	SVR- CONN	CLUS- SDR	CLUS- RCVR	V 8.0.0.4 AMQP
AFFINITY					✓				
ALTDATE	✓	✓	✓	✓	✓	✓	✓	✓	✓
ALTTIME	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 69. 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	V 8.0.0.4 AMQP
V 8.0.0.4 AMQPKA									✓
AUTOSTART		✓	✓	✓		✓			
BATCHHB	✓	✓					✓	✓	
BATCHINT	✓	✓		✓			✓	✓	
BATCHLIM	✓	✓					✓	✓	
BATCHSZ	✓	✓	✓	✓			✓	✓	
CERTLABL	✓	✓	✓	✓	✓	✓	✓	✓	✓
channel-name	✓	✓	✓	✓	✓	✓	✓	✓	✓
CHLTYPE	✓	✓	✓	✓	✓	✓	✓	✓	✓
CLNTWGHT					✓				
CLUSNL							✓	✓	
CLUSTER							✓	✓	
CLWLPRTY							✓	✓	
CLWLRANK							✓	✓	
CLWLWGHT							✓	✓	
COMPHDR	✓	✓	✓	✓	✓	✓	✓	✓	
COMPMSG	✓	✓	✓	✓	✓	✓	✓	✓	
CONNAME	✓	✓		✓	✓		✓	✓	
CONVERT	✓	✓					✓	✓	
DEFCDISP	✓	✓	✓	✓		✓			

Table 69. 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	V 8.0.0.4 AMQP
DEFRECON					✓				
DESCR	✓	✓	✓	✓	✓	✓	✓	✓	✓
DISCINT	✓	✓				✓	✓	✓	
HBINT	✓	✓	✓	✓	✓	✓	✓	✓	
KAINT	✓	✓	✓	✓	✓	✓	✓	✓	
LOCLADDR	✓	✓		✓	✓		✓	✓	✓
LONGRTY	✓	✓					✓	✓	
LONGTMR	✓	✓					✓	✓	
MAXINST						✓			✓
MAXINSTC						✓			
MAXMSGL	✓	✓	✓	✓	✓	✓	✓	✓	✓
MCANAME	✓	✓		✓			✓	✓	
MCATYPE	✓	✓		✓			✓	✓	
MCAUSER	✓	✓	✓	✓		✓	✓	✓	✓
MODENAME	✓	✓		✓	✓		✓	✓	
MONCHL	✓	✓	✓	✓		✓	✓	✓	
MRDATA			✓	✓				✓	
MREXIT			✓	✓				✓	
MRRTY			✓	✓				✓	
MRTMR			✓	✓				✓	
MSGDATA	✓	✓	✓	✓			✓	✓	
MSGEXIT	✓	✓	✓	✓			✓	✓	

Table 69. 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	V 8.0.0.4 AMQP
NETPRTY								✓	
NPMSPEED	✓	✓	✓	✓			✓	✓	
PASSWORD	✓	✓		✓	✓		✓		
V 8.0.0.4 PORT									✓
PROPCTL	✓	✓					✓		
PUTAUT			✓	✓		✓ "1" on page 1025		✓	
QMNAME					✓				
RESETSEQ	✓	✓	✓	✓			✓	✓	
RCVDATA	✓	✓	✓	✓	✓	✓	✓	✓	
RCVEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
SCYDATA	✓	✓	✓	✓	✓	✓	✓	✓	
SCYEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
SENDDATA	✓	✓	✓	✓	✓	✓	✓	✓	
SENDEXIT	✓	✓	✓	✓	✓	✓	✓	✓	
SEQWRAP	✓	✓	✓	✓			✓	✓	
SHARECNV						✓			
SHORTRTY	✓	✓					✓	✓	
SHORTTMR	✓	✓					✓	✓	
SSLCAUTH		✓	✓	✓		✓		✓	✓
SSLCIPH	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 69. 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	V 8.0.0.4 AMQP
SSLPEER	✓	✓	✓	✓	✓	✓	✓	✓	✓
STATCHL	✓	✓	✓	✓			✓	✓	
TPNAME	✓	✓		✓	✓	✓	✓	✓	
V 8.0.0.4 TPROOT									✓
TRPTYPE	✓	✓	✓	✓	✓	✓	✓	✓	
V 8.0.0.4 USECLTID									✓
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`.

ALTTIME
The time at which the definition was last altered, in the form `hh.mm.ss`.

V 8.0.0.4 **AMQPKA**
The keep alive time for an AMQP channel in seconds.

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 all platforms other than z/OS, 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 z/OS on all platforms except z/OS.

V 8.0.0.4 **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 **mcd.**, **jms.**, **usr.** or **mqext.**	All optional message properties (where the **Support** 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 it 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 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 **Support** 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 **Support** 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 *content='properties'* 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.

SSLCAUTH
Whether SSL client authentication is required.

SSLCIPH
Cipher specification for the SSL 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.

V 8.0.0.4 **TPROOT**
The topic root for an AMQP channel.

TRPTYPE
Transport type.

V 8.0.0.4 **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” on page 836.

DISPLAY CHANNEL (MQTT)

Use the MQSC command DISPLAY CHANNEL (MQTT) to display an IBM MQ Telemetry channel definition.

Using MQSC commands

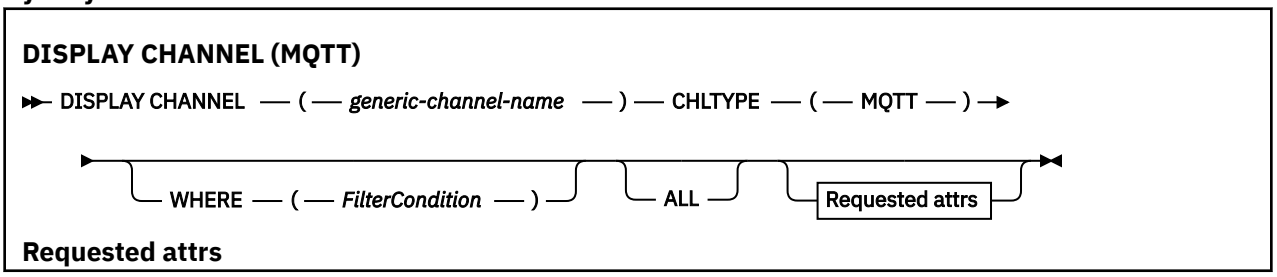
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

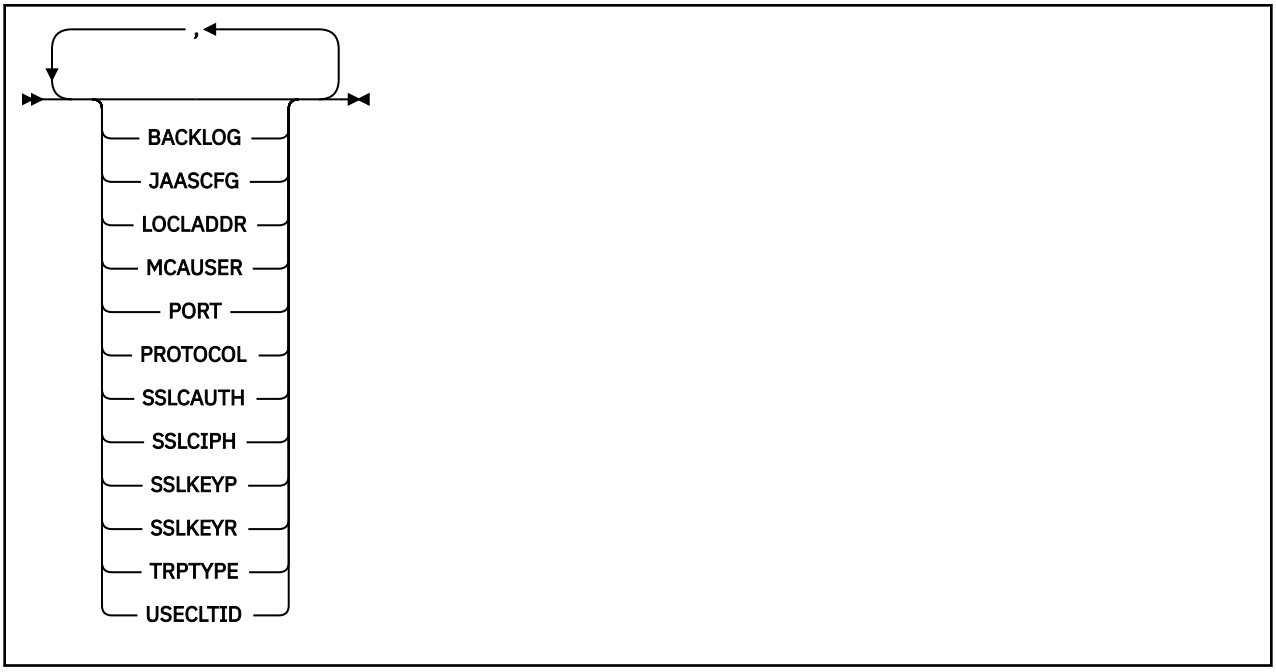
IBM i	AIX and Linux	Windows	z/OS
	✓	✓	

Note: For the telemetry server, AIX is the only supported UNIX platform.

- Syntax diagram
- “Parameter descriptions for DISPLAY CHANNEL (MQTT)” on page 1031
- “Requested parameters” on page 1033

Synonym: DIS CHL





DISPLAY CHANNEL (MQTT) command is only valid for IBM 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 SSL client.

SSLCIPH

When SSLCIPH is used with a telemetry channel, it means "SSL Cipher Suite".

SSLKEYP

The password for the key repository. If no passphrase is entered, you must use unencrypted connections.

SSLKEYR

The name of the Secure Sockets Layer (SSL) 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 (MQTT)" on page 890.

z/OS DISPLAY CHINIT

Use the MQSC command DISPLAY CHINIT to display information about the channel initiator. The command server must be running.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

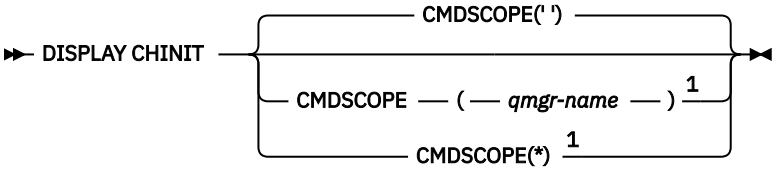
IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for DISPLAY CHINIT" on page 1034
- "Parameter descriptions for DISPLAY CHINIT" on page 1035

Synonym: DIS CHI or DIS DQM

DISPLAY CHINIT



Notes:

[1] Valid only on z/OS when the queue manager is a member of a queue-sharing group.

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 SSL 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 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.

DISPLAY CHLAUTH

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 Performing local administration tasks using MQSC commands.

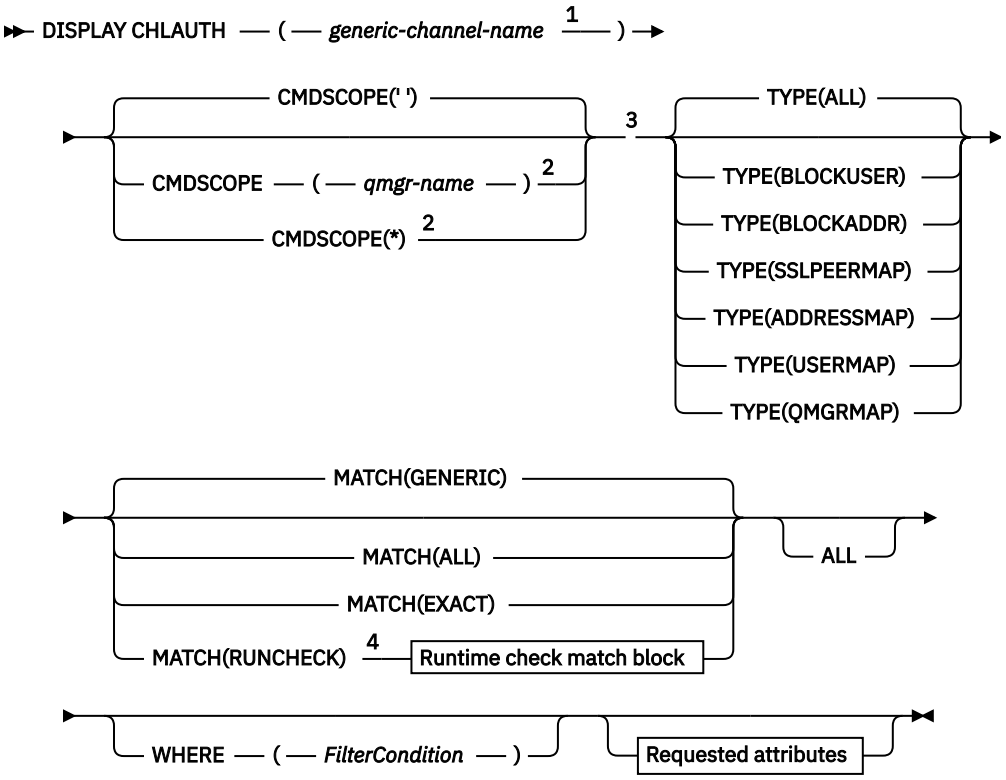
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

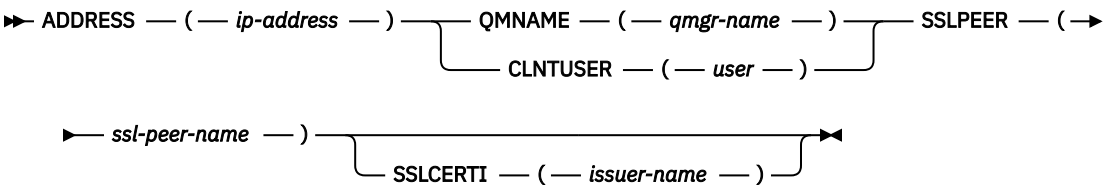
- Syntax diagram
- Parameters

Synonym: DIS CHLAUTH

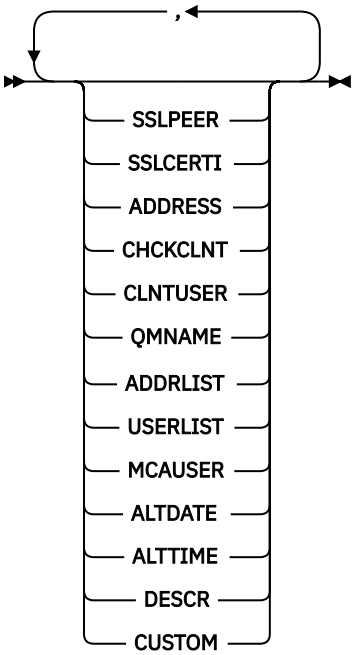
DISPLAY CHLAUTH



Runtime check match block



Requested attributes



Notes:

[1] Must be * with TYPE(BLOCKADDR) and cannot be generic with MATCH(RUNCHECK)
[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] Must be combined with TYPE(ALL)

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 hostname.

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 MQCONNX 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.

z/OS **CMDSCOPE**
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 hostname, if the queue manager is configured with **REVDNS(ENABLED)**.
- **SSLCERTI** attribute, only if the inbound channel uses SSL or TLS.
- **SSLPEER** attribute, only if the inbound channel uses SSL or 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 wild cards. 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: 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.

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 SSL 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*.

ALTTIME
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

Generic IP addresses

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 are all valid values:
  - 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 are all valid values:
  - 2001:DB8:0:0:0:0:0:*
  - 2001:DB8:0:0:0:*
  - 2001:DB8:0:0:0:*: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:*:0:0-8

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

DISPLAY CHSTATUS

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 Performing local administration tasks using MQSC commands.

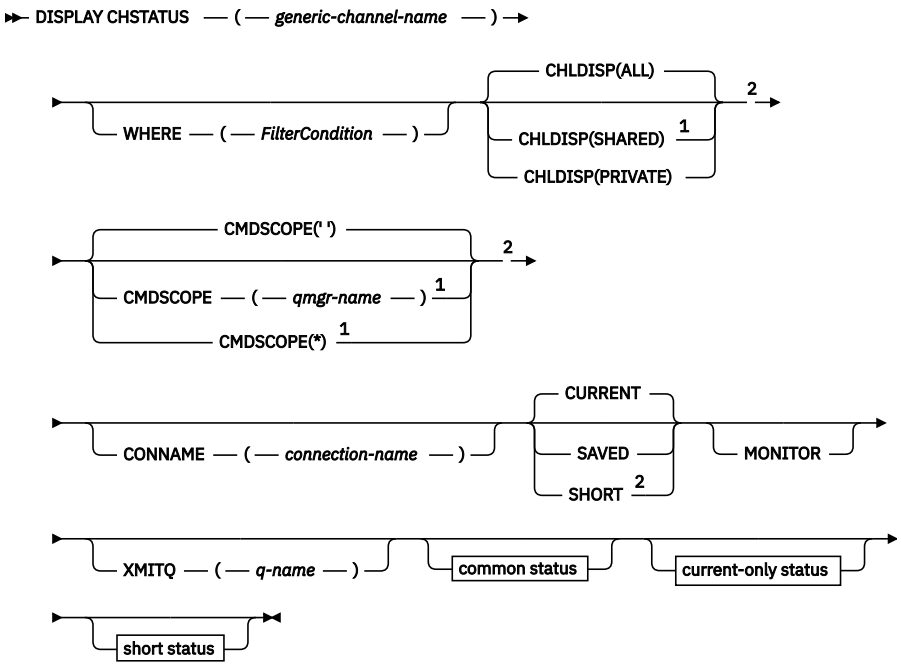
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

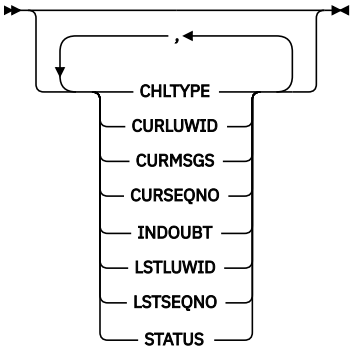
- Syntax diagram
- z/OS "Usage notes for DISPLAY CHSTATUS" on page 1045
- "Parameter descriptions for DISPLAY CHSTATUS" on page 1046
- "Summary attributes" on page 1051
- "Common status" on page 1051
- "Current-only status" on page 1054
- z/OS "Short status" on page 1061

Synonym: DIS CHS

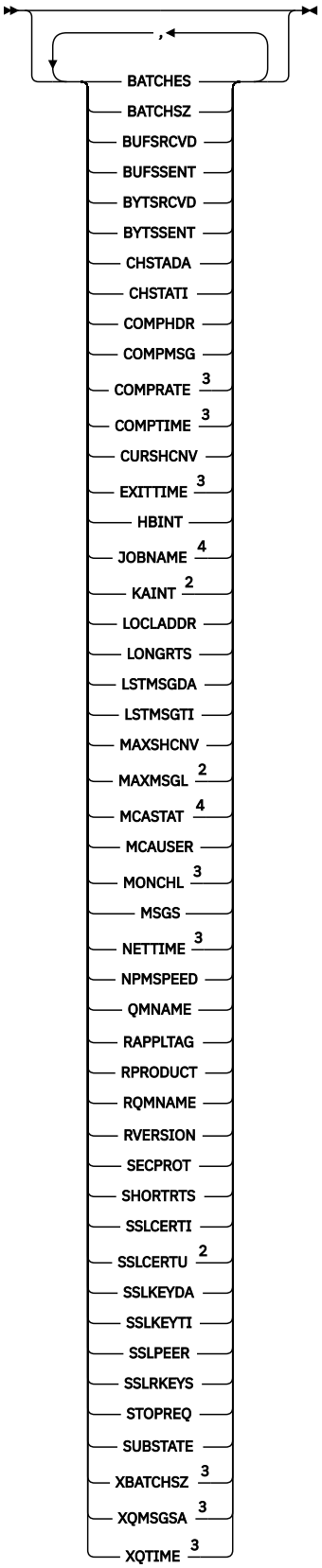
DISPLAY CHSTATUS



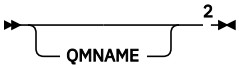
Common status



Current-only status



Short status



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] Also displayed by selection of the MONITOR parameter.
[4] Ignored if specified on z/OS.

Usage notes for DISPLAY CHSTATUS

z/OS 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 70 on page 1045, Table 71 on page 1045, and Table 72 on page 1046.

z/OS

Table 70. 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

z/OS

Table 71. 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 71. 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 1046)	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 1046)

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.

z/OS

Table 72. 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

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** z/OS, 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.
- z/OS **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.

However, you cannot use the following parameters as filter keywords on distributed platforms COMPRATE, COMPTIME, CURRENT, EXITTIME, JOBNAME, NETTIME, SAVED, SHORT, XBATCHSZ, or XQTIME as filter keywords.

z/OS Additionally, on z/OS, you cannot use the following parameters as filter keywords: CHLDISP, CMDSCOPE, MCASTAT, or MONITOR as filter keywords.

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 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 it 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.

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.

CONNAME(*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 CONNAME might not be the same as in the channel definition, and might differ between the current channel status and the saved channel status. (Using CONNAME for limiting the number of sets of status is therefore not recommended.)

For example, when using TCP, if CONNAME 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 z/OS (except on z/OS), but the saved channel status value does not.

For SAVED z/OS or SHORT status, this value could also be the queue manager name, or queue-sharing group name, of the remote system.

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, XBATCHSZ, 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 z/OS, or on z/OS only, SHORT)
- STATUS z/OS (except SAVED on z/OS)
- 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.

Summary attributes

When SUMMARY or TOTAL are added to the MQSC command DISPLAY CHSTATUS, the number of conversations is displayed as the CONVS attribute. The following attributes display a summary for either each channel when SUMMARY is specified, or for all the channels when TOTAL is specified.

ALL
Specify this to display all the status information for each relevant instance. This attribute is the default value if no attributes are requested.

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.

CURCNV
The number of current conversations.

Common status

The following information applies to all sets of channel status, whether or not the set is current. 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

V 8.0.0.4 **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.

CURMSGS
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.

z/OS 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.

z/OS On z/OS, STATUS is not displayed if saved data is requested.

On distributed platforms, 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.

V 8.0.0.4 **AMQPKA**
The keep alive time for an AMQP channel in seconds. 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.

BYTSSENT
Number of bytes sent during this session (since the channel was started). This includes control information sent by the message channel agent.

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 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 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 1061.

COMPTIME

The amount of time for each message, displayed in microseconds, spent during 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: z/OS 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 SSL 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 1061.

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 IBM MQ earlier than Version 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 1061.

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.

- On IBM i, Windows, UNIX and Linux systems, 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 of IBM MQ earlier than Version 7.0, regarding:

- Administrator stop-quiesce
- Heartbeating
- Read ahead
- Sharing conversations
- Client asynchronous consumption

z/OS **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.

This information is not available on z/OS. The parameter is ignored if specified.

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 distributed platforms is 64characters.

z/OS 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 1061.

NPMSPEED
The nonpersistent message handling technique being used for this session.

V 8.0.0.4 **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. 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 73 on page 1058.

Table 73. Product Identifier values	
Product Identifier	**Description**
MQMM	Queue manager on a distributed platform
z/OS z/OS MQMV	Queue manager on z/OS
MQCC	IBM MQ C client
MQNC	IBM MQ client for HP Integrity NonStop Server
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

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

V 8.0.0.2 **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.

Possible values are:

NONE
No security protocol

SSLV3
SSL version 3.0

TLSV1
TLS version 1.0

TLSV12
TLS version 1.2

z/OS SECPROT is not available on z/OS.

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. This limit might mean that exceptionally long Distinguished Names are truncated.

z/OS **SSLCERTU**
The local user ID associated with the remote certificate. This is valid on z/OS only.

SSLKEYDA
Date on which the previous successful SSL secret key reset was issued.

SSLKEYTI
Time at which the previous successful SSL secret key reset was issued.

SSLPEER
Distinguished Name of the peer queue manager or client at the other end of the channel.

The maximum length is 256 characters. This limit might mean that exceptionally long Distinguished Names are truncated.

SSLRKEYS
Number of successful SSL key resets. The count of SSL secret key resets is reset when the channel instance ends.

STOPREQ
Whether a user stop request is outstanding. This is either YES or NO.

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.

z/OS **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 an SSL 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.

V 8.0.0.4 **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

XBATCHSZ
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 1061.

V 8.0.0.4 **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.

z/OS 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 1061.

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 1061.

Short status

z/OS

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" on page 724 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" on page 653 for more information.

DISPLAY CHSTATUS (AMQP)

Use the MQSC command DISPLAY CHSTATUS (AMQP) to display the status of one or more AMQP channels.

Using MQSC commands

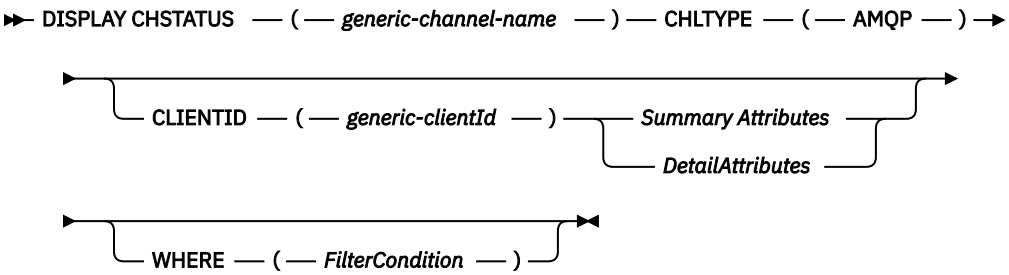
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	

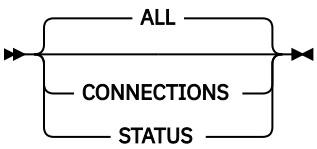
- Syntax diagram
- “Parameter descriptions for DISPLAY CHSTATUS” on page 1063
- “Summary attributes” on page 1064

Synonym: DIS CHS

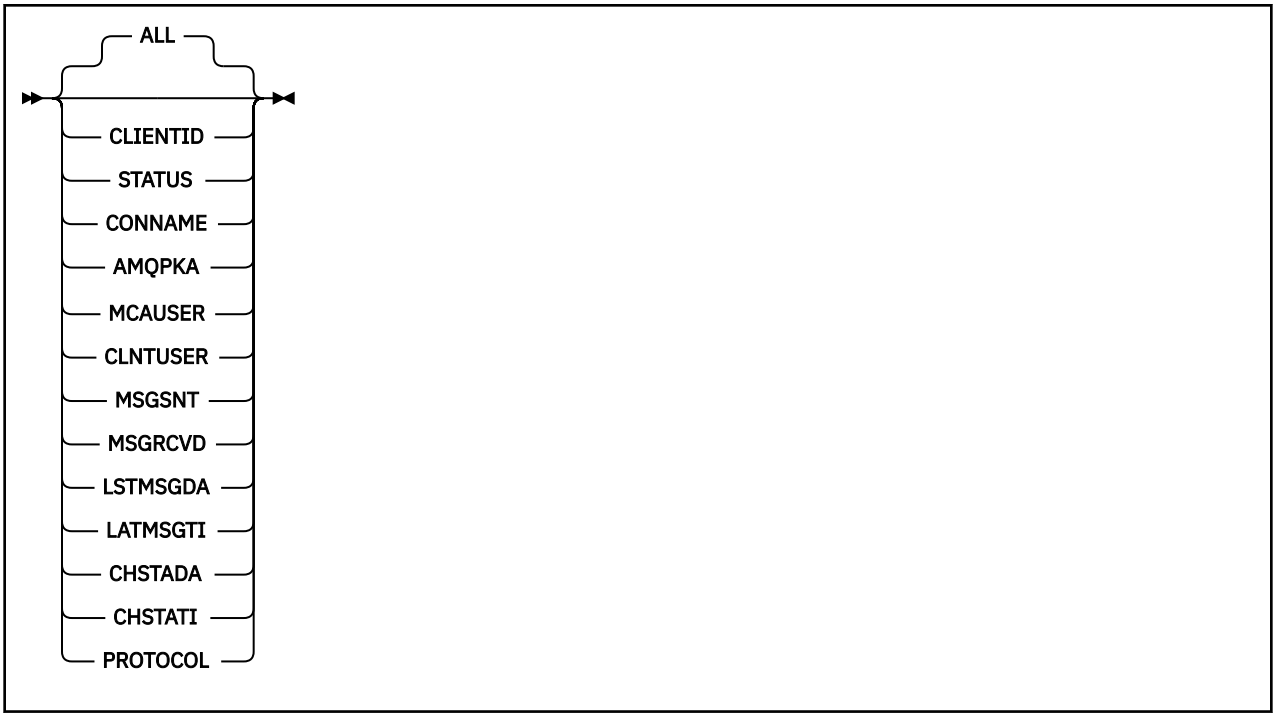
DISPLAY CHSTATUS (AMQP)



SummaryAttributes



DetailAttributes



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 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 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.

CONNECTIONS

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.

MSGSNT

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( )
   MSGSNT(0)                               MSGRCVD(0)
   LSTMSGDA( )                             LSTMSGTI( )
   CHSTADA(2015-09-18)                     CHSTATI(06.23.30)
   PROTOCOL(AMQP)
```

DISPLAY CHSTATUS (MQTT)

Use the MQSC command DISPLAY CHSTATUS (MQTT) to display the status of one or more IBM MQ Telemetry channels.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

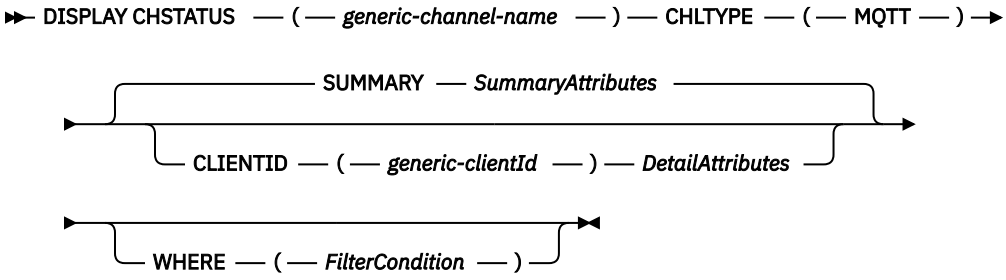
IBM i	AIX and Linux	Windows	z/OS
	✓	✓	

Note: For the telemetry server, AIX is the only supported UNIX platform.

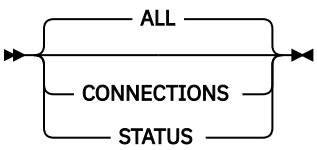
- Syntax diagram
- “Parameter descriptions for DISPLAY CHSTATUS” on page 1067
- “Summary attributes” on page 1069

Synonym: DIS CHS

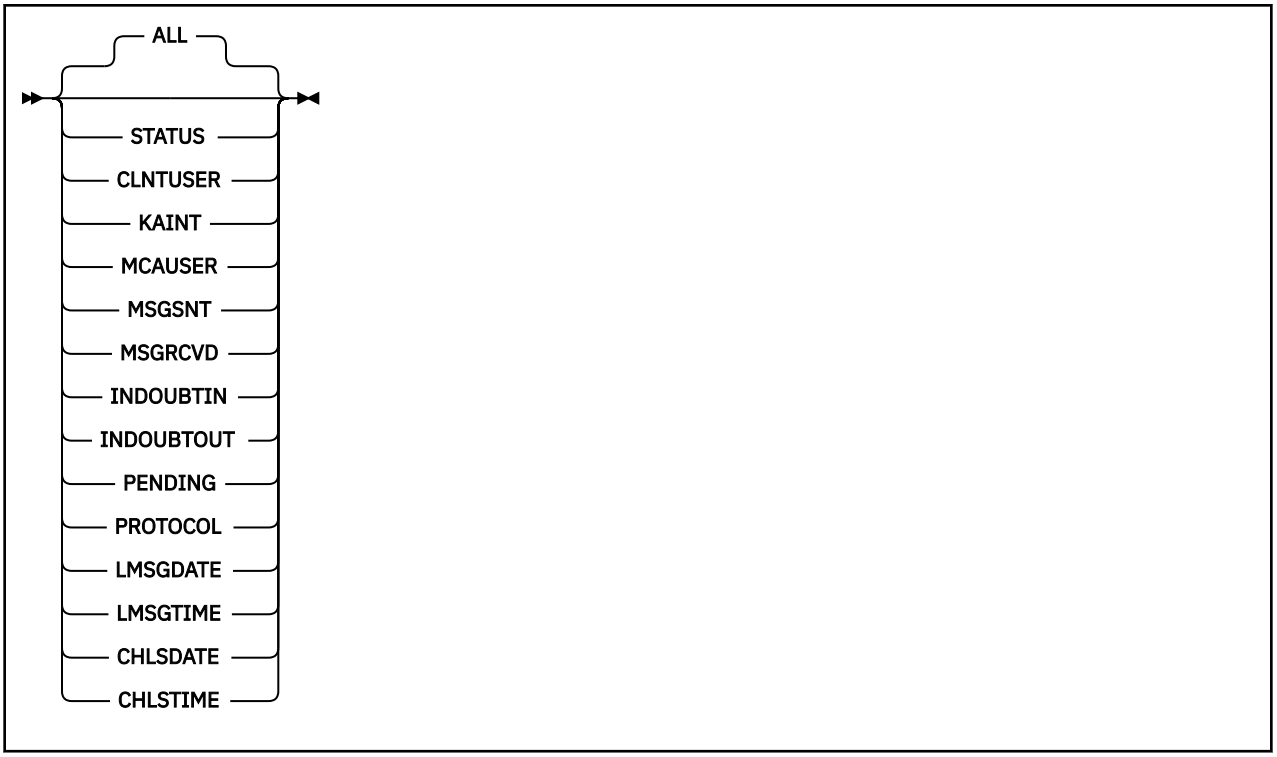
DISPLAY CHSTATUS (MQTT)



SummaryAttributes



DetailAttributes



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.
- Either **CLIENTID**, **SUMMARY**, or neither may be specified, but not both at the same time.
- The **DISPLAY CHSTATUS** command for IBM 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 IBM 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 IBM MQ Telemetry command that is truncated by the IBM 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 if 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.

MSGSNT

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

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 Performing local administration tasks using MQSC commands.

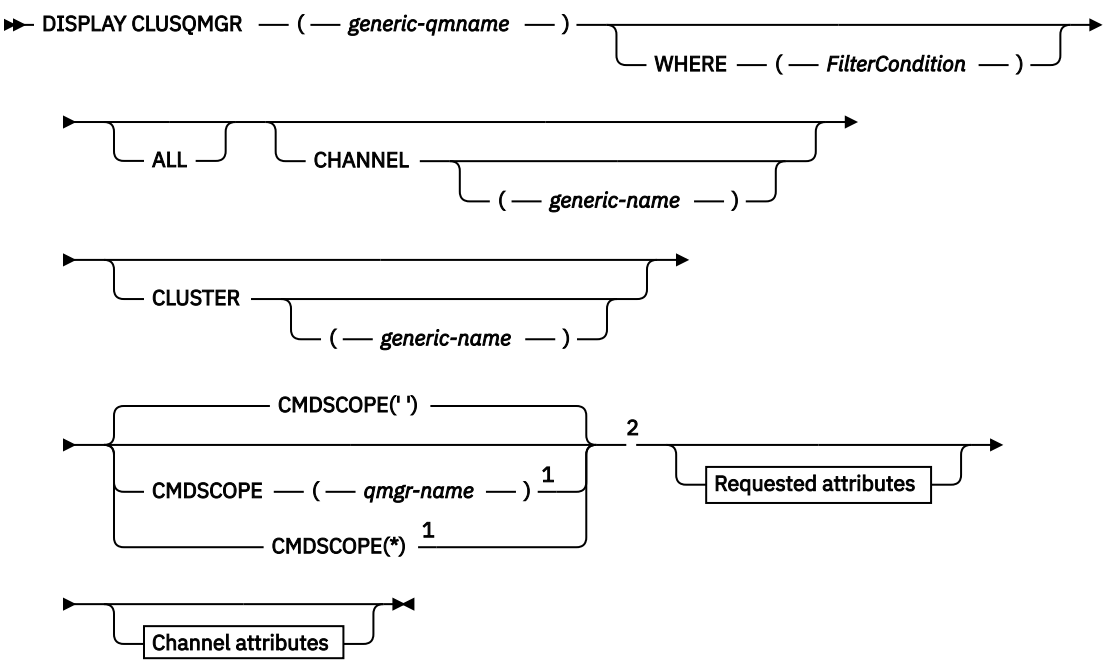
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see “Using commands on z/OS” on page 629.

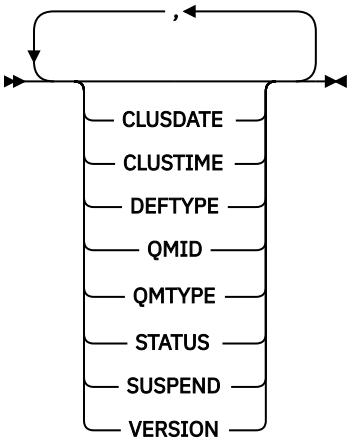
- Syntax diagram
- “Usage notes” on page 1073
- “Parameter descriptions for DISPLAY CLUSQMGR” on page 1073
- “Requested parameters” on page 1075
- “Channel parameters” on page 1076

Synonym: `DIS CLUSQMGR`

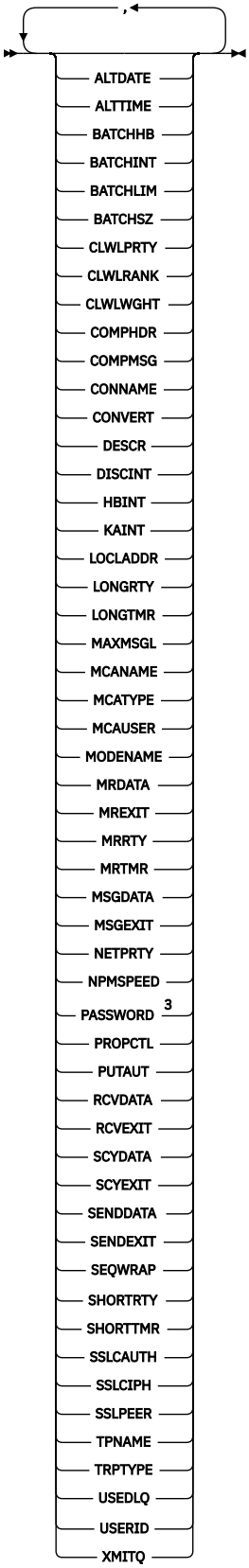
DISPLAY CLUSQMGR



Requested attributes



Channel 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.

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:

1. On UNIX systems, the command is valid only on AIX, HP-UX, Linux, and Solaris.
2. 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.

z/OS 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.

z/OS **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.

' '
The command is executed on the queue manager on which it was entered. ' ' 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. 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 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.

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.

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.

DISCINT
Disconnection interval.

HBINT
Heartbeat interval.

KAINT
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 SSL client authentication is required.

SSLCIPH
Cipher specification for the SSL 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" on page 836.

z/OS DISPLAY CMDSERV

Use the MQSC command DISPLAY CMDSERV to display the status of the command server.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

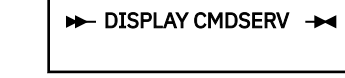
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for DISPLAY CMDSERV" on page 1079

Synonym: DIS CS

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

Use the MQSC command DISPLAY COMMINFO to display the attributes of a communication information object.

Using MQSC commands

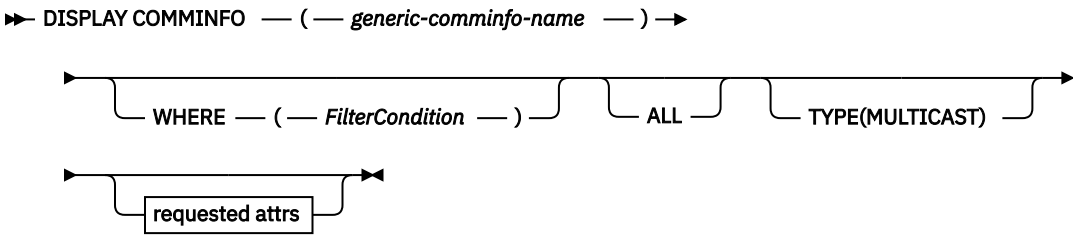
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

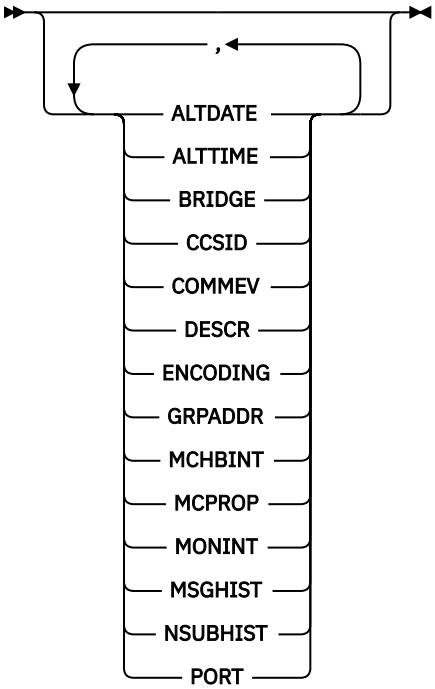
- Syntax diagram
- “Parameter descriptions for DISPLAY COMMINFO” on page 1080
- “Requested parameters” on page 1081

Synonym: DIS COMMINFO

DISPLAY COMMINFO



Requested attrs



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

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX	Windows	z/OS
✓	✓	✓	2CR

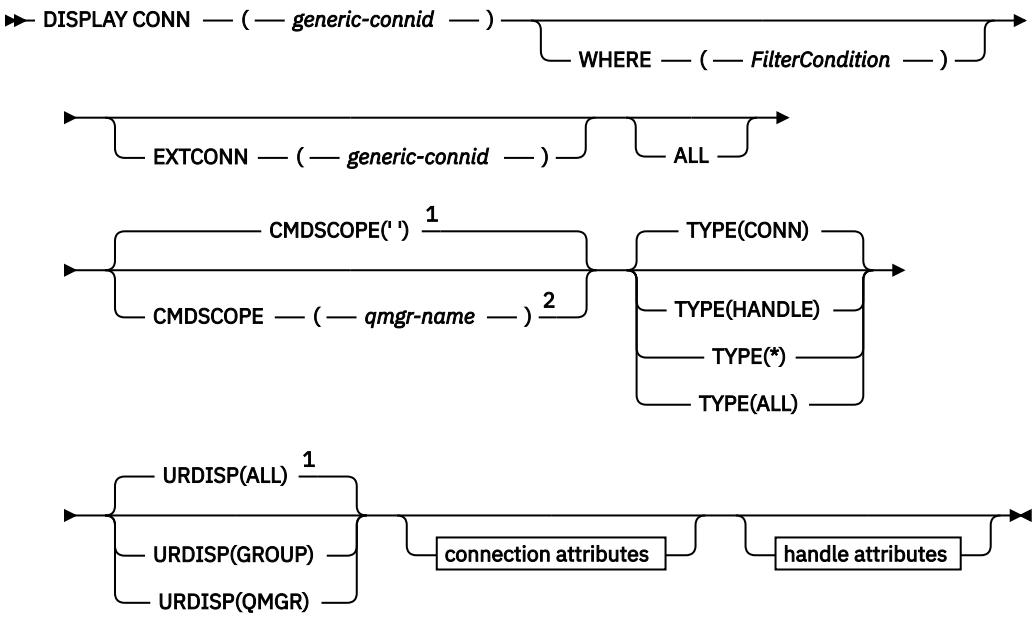
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for DISPLAY CONN" on page 1085
- "Parameter descriptions for DISPLAY CONN" on page 1085
- "Connection attributes" on page 1088

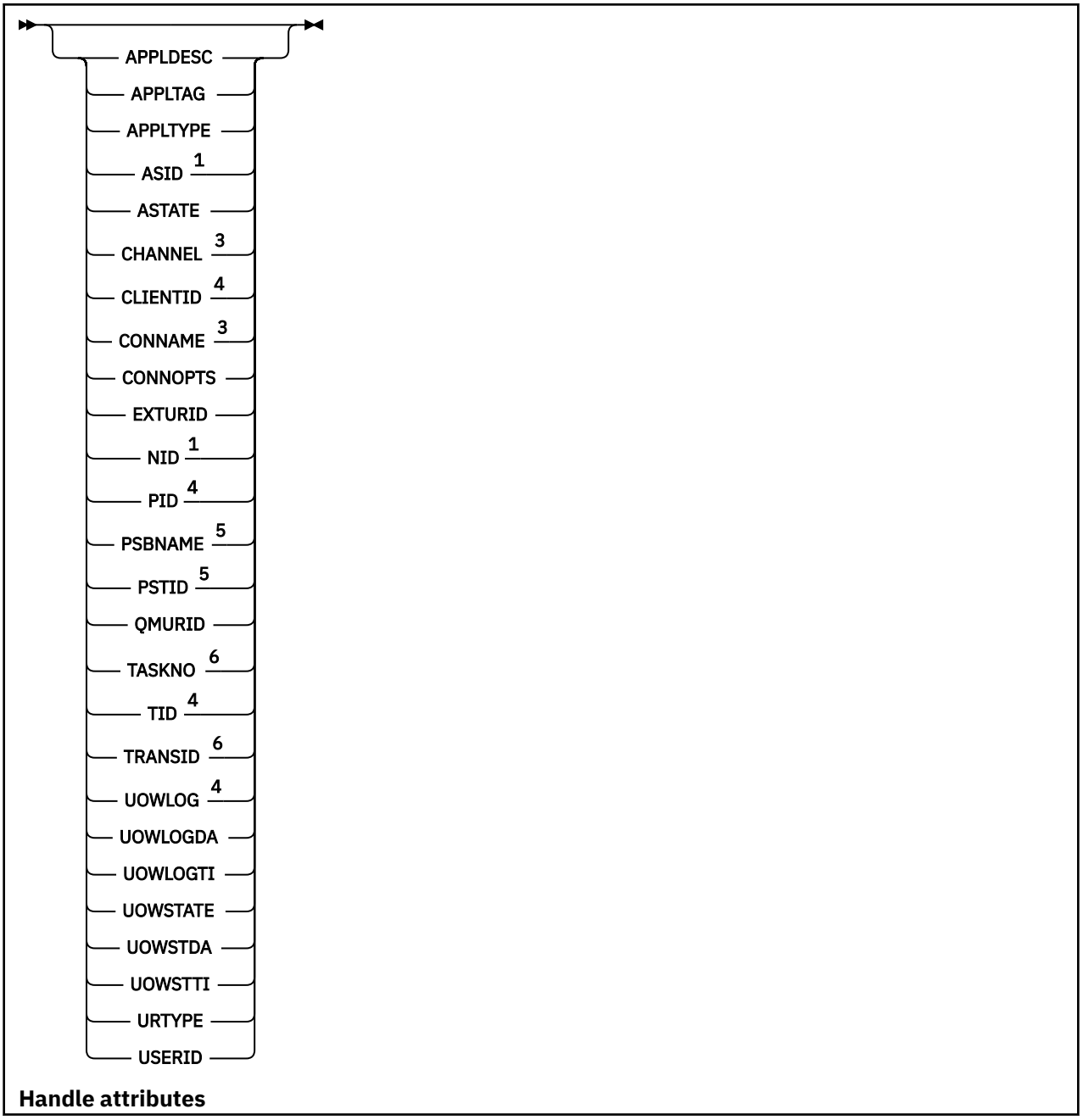
- “Handle attributes” on page 1092
- “Full attributes” on page 1095

Synonym: DIS CONN

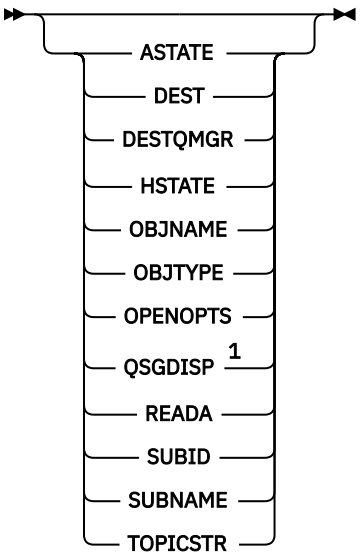
DISPLAY CONN



Connection attributes



Handle attributes



Notes:

[1] Valid only on z/OS.
[2] Valid only when the queue manager is a member of a queue-sharing group.
[3] Valid only when the connection is associated with a channel.
[4] Not valid on z/OS.
[5] IMS only.
[6] CICS for z/OS only.

Usage notes for DISPLAY CONN

1. z/OS 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.

   z/OS On z/OS, these non-printable characters will be displayed as blanks.

   On distributed 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.

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.

z/OS **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.

' '
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 it 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.

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.

z/OS 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
- UNIX process

  Notes:

  – HP-UX On HP-UX, if the process name exceeds 14 characters, only the first 14 characters are shown.

  – Solaris Linux On Linux and Solaris, 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 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 UNIX application

USER
A user application

WINDOWSNT
A Windows application

z/OS **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 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.

V 8.0.0.2 **CLIENTID**
Not valid on z/OS.

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_Q_DISABLED
- MQCNO_ACCOUNTING_Q_ENABLED
- MQCNO_ACCOUNTING_MQI_DISABLED
- MQCNO_ACCOUNTING_MQI_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

The values displayed for MQCNO_RECONNECT and MQCNO_RECONNECT_Q_MGR are only displayed if the application specifies them explicitly. If the values are picked up from an `mqclient.ini` file setting or the CLNTCONN channel definition, then neither value is displayed.

You cannot use the value MQCNO_STANDARD_BINDING as a filter value with the CT and EX operators on the WHERE parameter.

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.

On platforms other than z/OS, 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.

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 on z/OS only.

This parameter is returned only when the APPLTYPE parameter has the value CICS.

UOWLOG
The file name of the extent to which the transaction associated with this connection first wrote.

z/OS This parameter is valid only on platforms other than z/OS.

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
- z/OS 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.

DISPLAY ENTAUTH

Use the MQSC command DISPLAY ENTAUTH to display the authorizations an entity has to a specified object.

Using MQSC commands

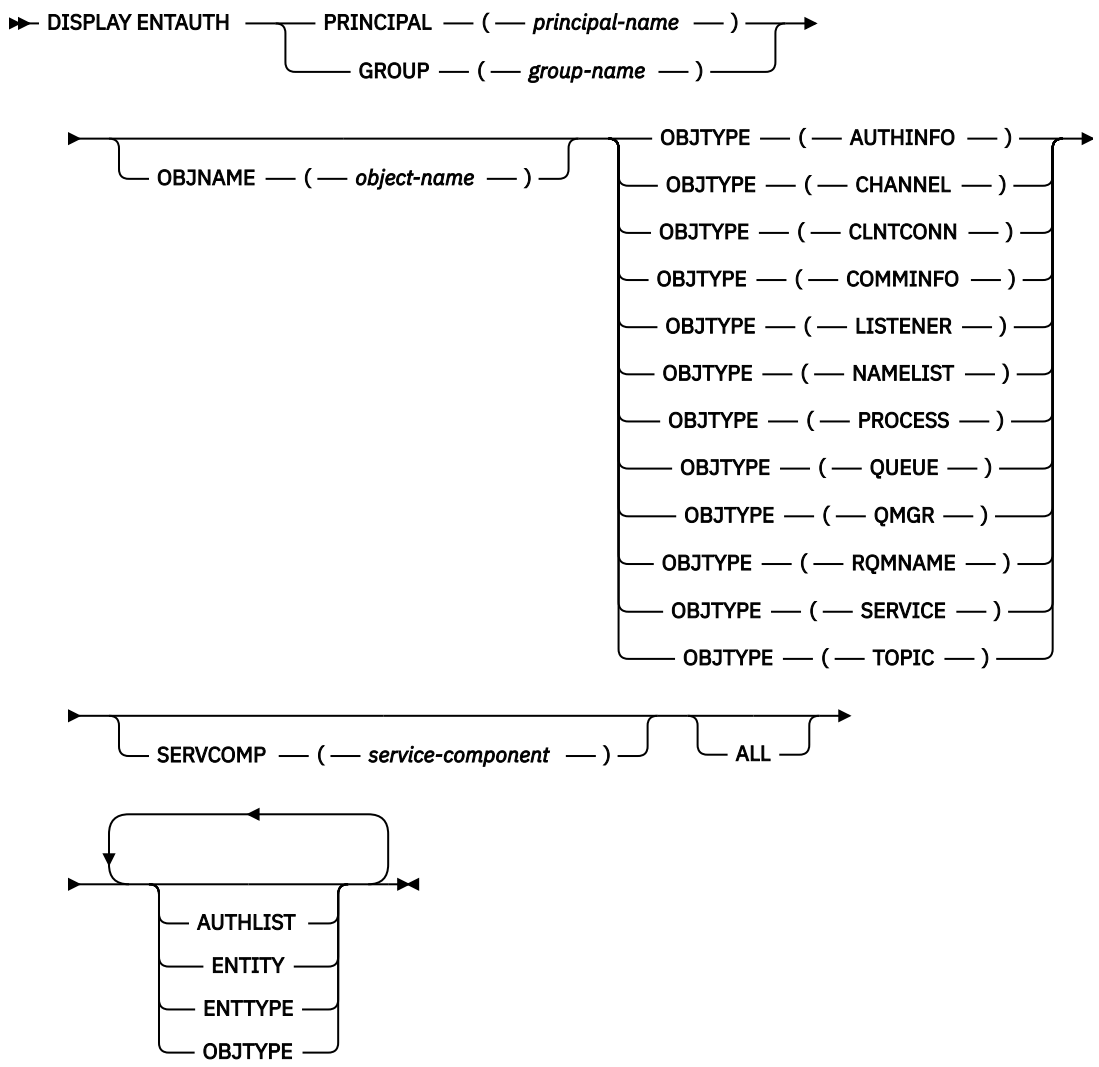
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Parameter descriptions” on page 1096
- “Requested parameters” on page 1098

Synonym: DIS ENTAUTH

DISPLAY 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.

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

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

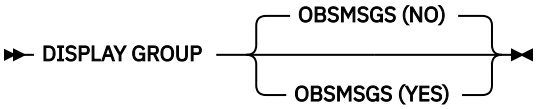
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for DISPLAY GROUP” on page 1098
- “Parameter descriptions for DISPLAY GROUP” on page 1099

Synonym: DIS GROUP

DISPLAY 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.

DISPLAY LISTENER

Use the MQSC command DISPLAY LISTENER to display information about a listener.

Using MQSC commands

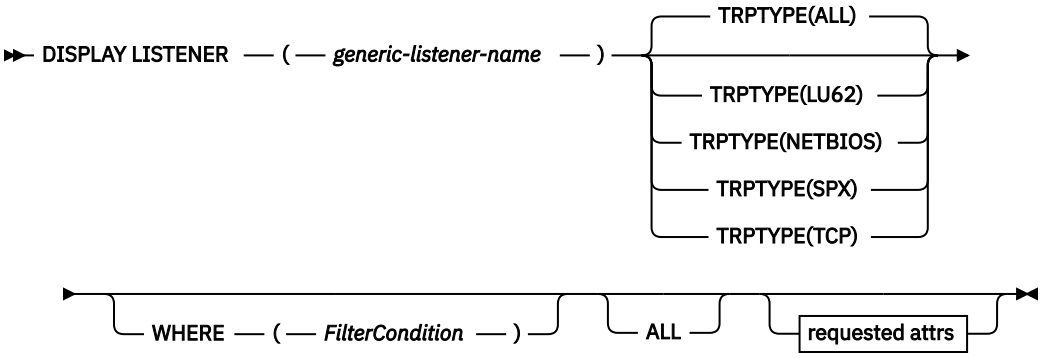
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

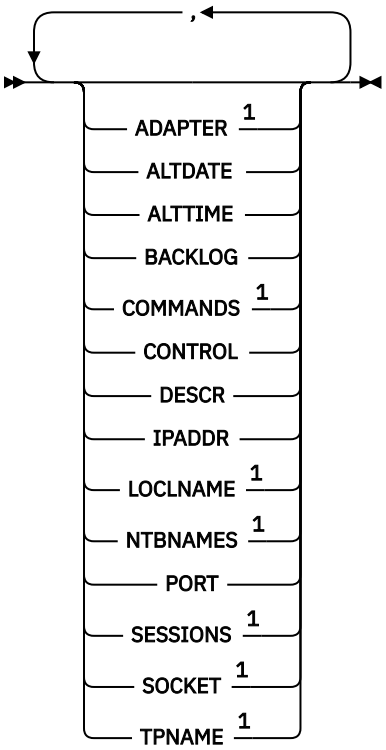
- Syntax diagram
- "Usage notes" on page 1100
- "Keyword and parameter descriptions for DISPLAY LISTENER" on page 1100
- "Requested parameters" on page 1102

Synonym: DIS LSTR

DISPLAY LISTENER



Requested attrs



Notes:

[1] Valid only on Windows.

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.

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.

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" on page 897.

z/OS DISPLAY LOG

Use the MQSC command DISPLAY LOG to display log system parameters and information.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

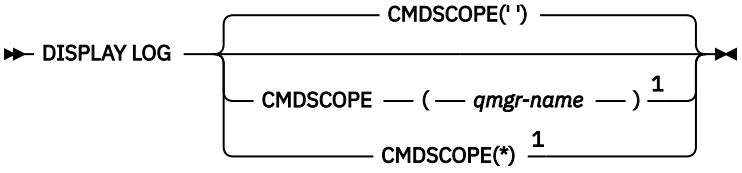
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for DISPLAY LOG” on page 1103
- “Parameter descriptions for DISPLAY LOG” on page 1103

Synonym: DIS LOG

DISPLAY LOG



Notes:

1 Valid only when the queue manager is a member of a queue-sharing group.

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).
   - 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 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 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.

DISPLAY LSSTATUS

Use the MQSC command DISPLAY LSSTATUS to display status information for one or more listeners.

Using MQSC commands

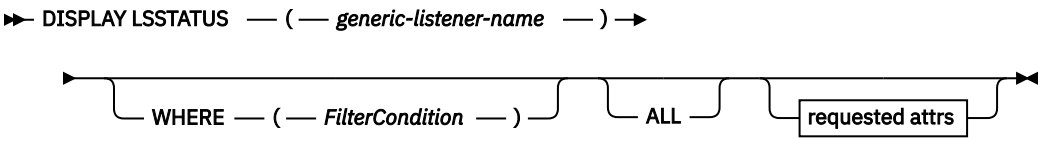
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

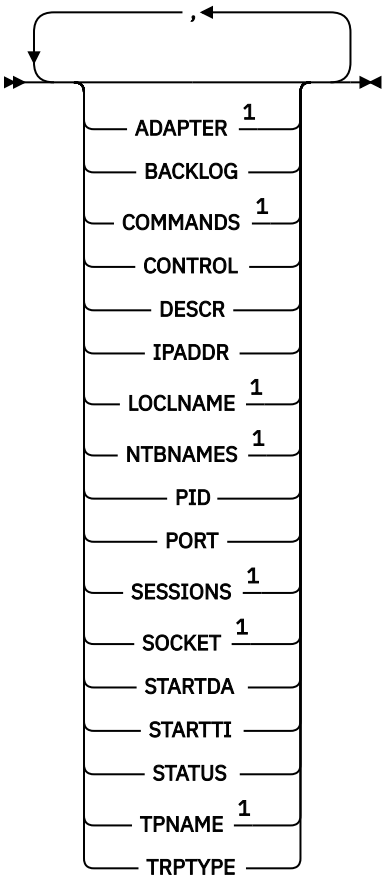
- Syntax diagram
- “Keyword and parameter descriptions for DISPLAY LSSTATUS” on page 1105
- “Requested parameters” on page 1106

Synonym: DIS LSSTATUS

DISPLAY LSSTATUS



Requested attrs



Notes:

[1] 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" on page 897.

z/OS DISPLAY MAXSMSGS

Use the MQSC command DISPLAY MAXSMSGS to see the maximum number of messages that a task can get or put within a single unit of recovery.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

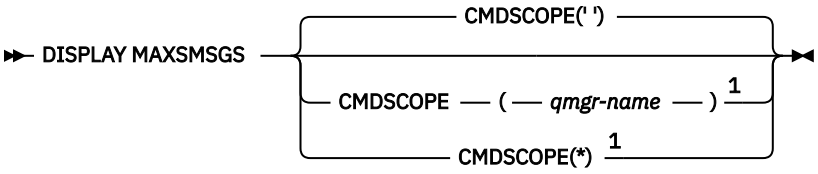
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 1108
- "Parameter descriptions for DISPLAY MAXSMSGS" on page 1108

Synonym: DIS MAXSM

DISPLAY MAXSMSGS



Notes:

[1] Valid only on full function IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.

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 MAXSMSGS

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.

DISPLAY NAMELIST

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 Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

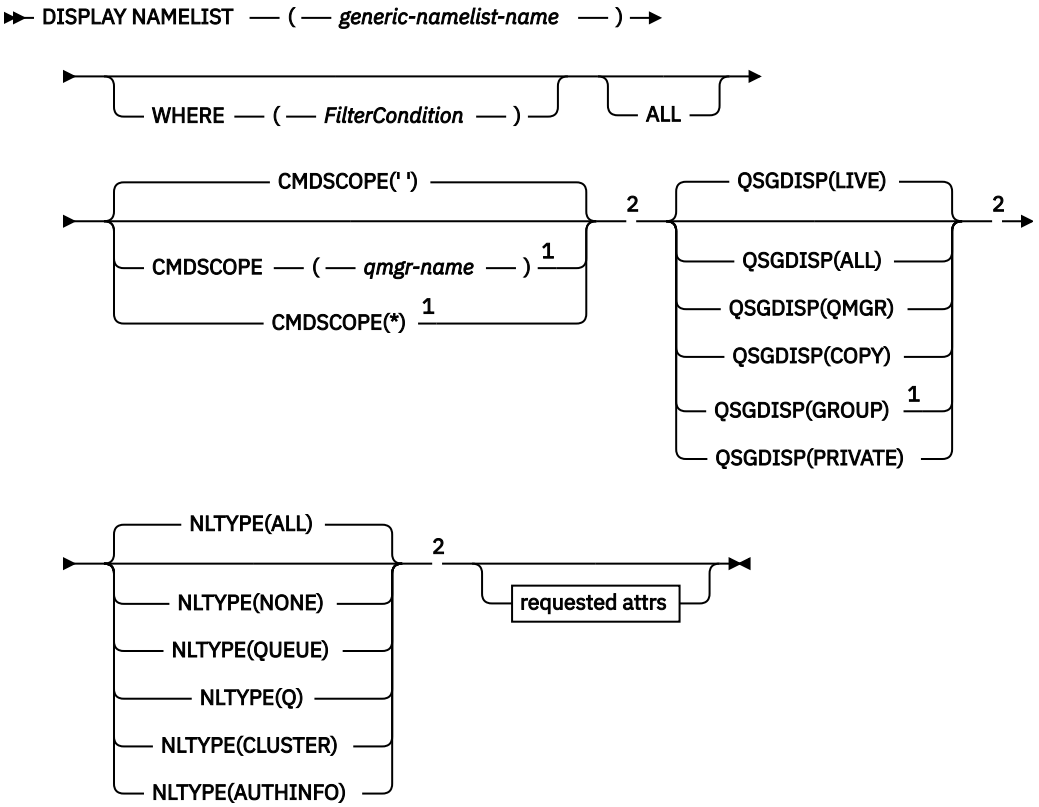
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Parameter descriptions for DISPLAY NAMELIST" on page 1109

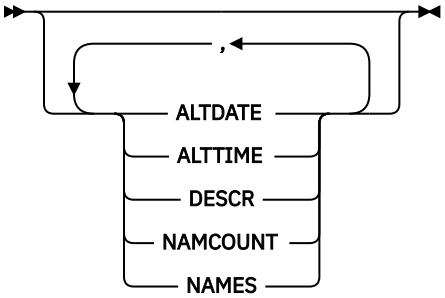
- "Requested parameters" on page 1112

Synonym: DIS NL

DISPLAY NAMELIST



Requested attrs



Notes:

[1] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[2] Valid only on z/OS.

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.

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 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).

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.

z/OS **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" on page 903 for more information about the individual parameters.

DISPLAY POLICY

Use the MQSC command DISPLAY POLICY to display a security policy.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- "Parameter descriptions for DISPLAY POLICY" on page 1113

DISPLAY POLICY

Parameter descriptions for DISPLAY POLICY

(policy-name)
Specifies the policy name, or names, to be displayed.

The name of the policy, or policies (or part of the policy name or names) to display are the same as the name of the queue, or queues, that the policies control. 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" on page 1284
Use the MQSC command SET POLICY to set a security policy.

"setmqspl" on page 144
Use the **`setmqspl`** command to define a new security policy, replace an already existing one, or remove an existing policy.

"dspmqspl" on page 73

Use the **dspmqspl** command to display a list of all policies and details of a named policy.

Related information
Managing security policies in AMS

DISPLAY PROCESS

Use the MQSC command DISPLAY PROCESS to display the attributes of one or more IBM MQ processes.

Using MQSC commands

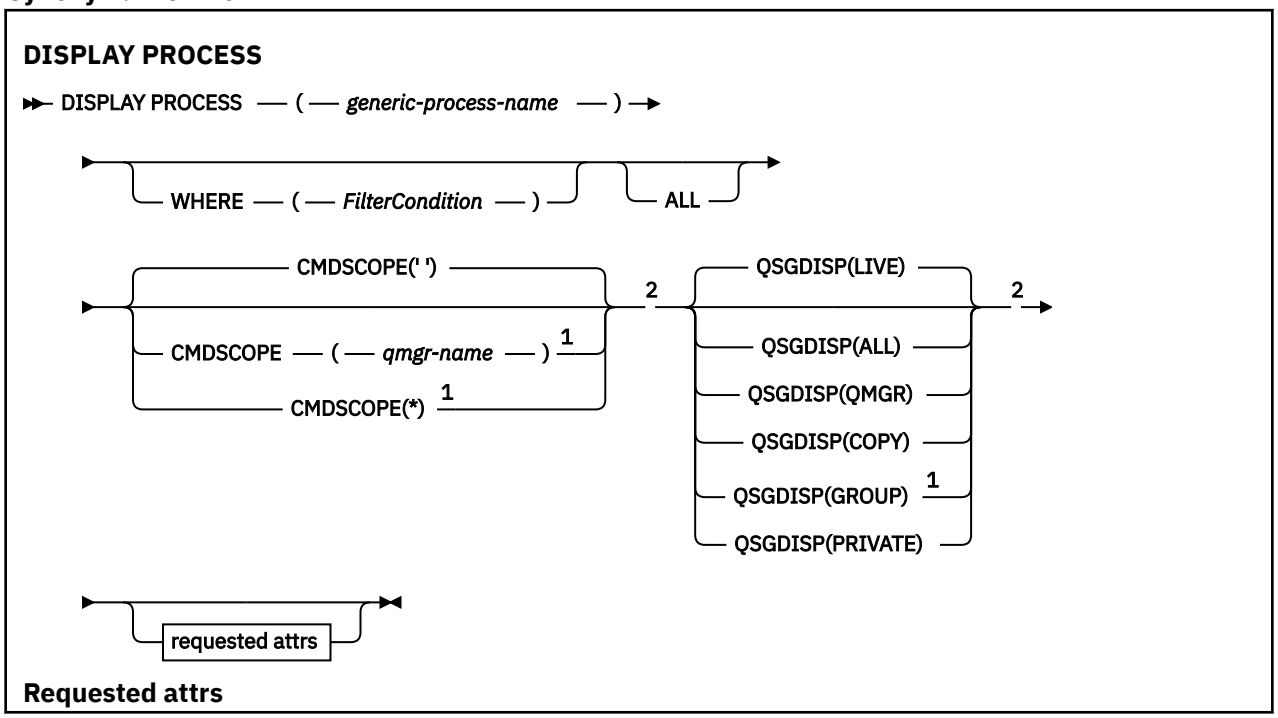
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

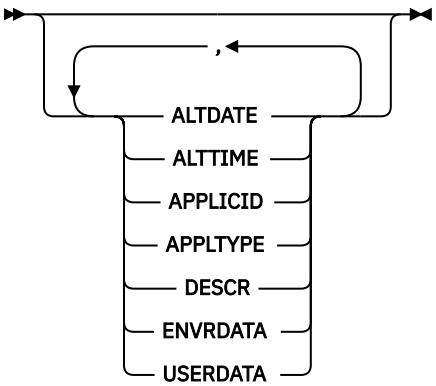
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for DISPLAY PROCESS” on page 1115
- “Requested parameters” on page 1117

Synonym: DIS PRO





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.

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. z/OS However, 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 AIX, HP-UX, Linux, IBM i, Solaris, and Windows z/OS, and z/OS, 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 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. 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.

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 the object names and, on z/OS only, 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 908, 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" on page 906 for more information about individual parameters.

DISPLAY PUBSUB

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 Performing local administration tasks using MQSC commands.

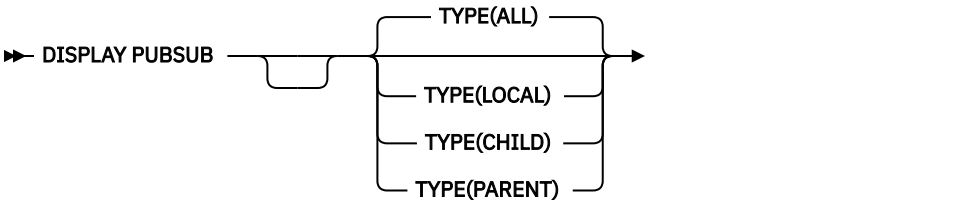
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

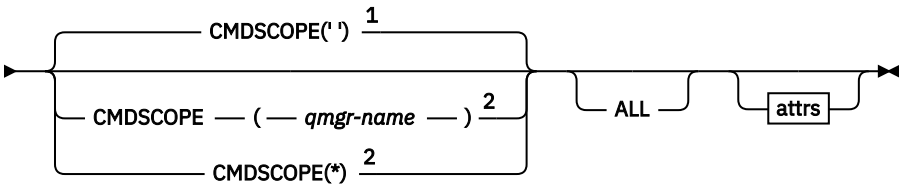
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for DISPLAY PUBSUB” on page 1118
- “Returned parameters” on page 1119

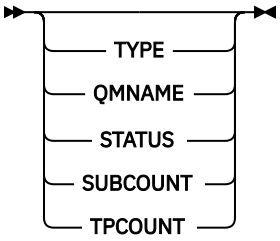
Synonym: None

DISPLAY PUBSUB





Attrs



Notes:

[1] Valid only on z/OS.
[2] 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 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.

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.

z/OS 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.

z/OS 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, z/OS 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 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

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 Performing local administration tasks using MQSC commands.

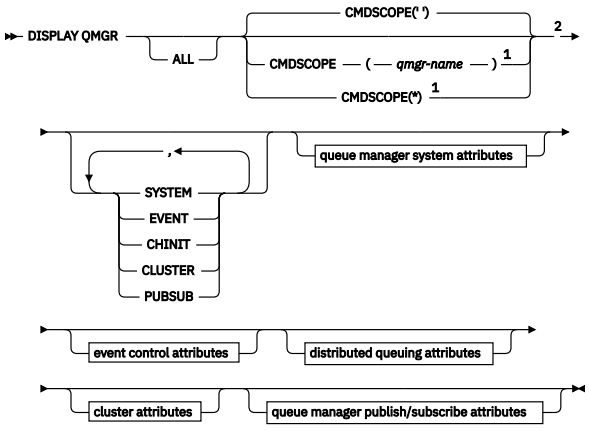
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

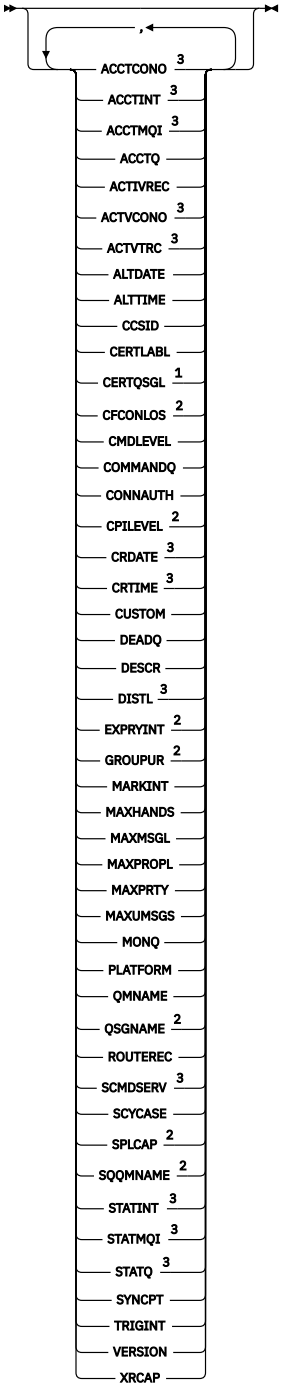
- Syntax diagram
- “Parameter descriptions for DISPLAY QMGR” on page 1124
- “Requested parameters” on page 1125

Synonym: DIS QMGR

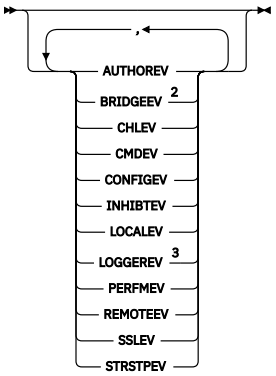
DISPLAY QMGR



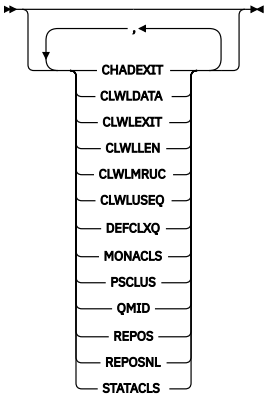
Queue manager system attributes



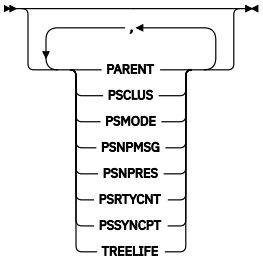
Event control attributes



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.

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 AIX, HP-UX, Linux, IBM i, Solaris, and Windows, this parameter is the default if you do not request any specific parameters.

z/OS **CMDSCOPE**
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 command 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 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 1125 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 1125 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 1125 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 1125 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 1125 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:

ACCTCONO
Whether the settings of the ACCTQMQI and ACCTQ queue manager parameters can be overridden. This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

ACCTINT
The interval at which intermediate accounting records are written. This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

ACCTMQI
Whether accounting information is to be collected for MQI data. This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

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.

ACTVCONO
Whether the settings of the ACTVTRC queue manager parameter can be overridden. This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

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 IBM i , UNIX, Linux, and Windows systems.

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.

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`.

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 **CERTQSGL**
Specifies the queue sharing group (QSG) certificate label.

This parameter is valid only on z/OS.

CERTVPOL
Specifies which SSL/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 on only UNIX, 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.

CHAD
Whether auto-definition of receiver and server-connection channels is enabled.

z/OS This parameter is not valid on z/OS.

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.

CHIADAPS
The number of adapter subtasks to use to process IBM MQ calls.

z/OS 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.

CHISERVP
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.

CLWLLEN
The maximum number of bytes of message data that is passed to the cluster workload exit.

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. This behavior is implicit in versions of IBM WebSphere MQ, earlier than Version 7.5. In earlier versions, the queue manager attribute 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.

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. Refer to WLM/DNS no longer supported. This parameter is valid only on z/OS

z/OS **DNSWLM**
This parameter is no longer used. Refer to 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.

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.

IPADDRV
Whether to use an IPv4 or IPv6 IP address for a channel connection in ambiguous cases.

LOCALEV
Whether local error events are generated.

LOGGEREV
Whether recovery log events are generated. This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

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.

This parameter is valid only on IBM i, z/OS, UNIX, Linux, and Windows systems.

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 IBM MQ Telemetry. IBM 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, V 8.0.0.2 APPLIANCE, UNIX, or 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.

z/OS **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 IBM MQ for z/OS.

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.

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.

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.

z/OS 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 IBM MQ Advanced Message Security (IBM MQ AMS) capabilities are available to the queue manager. If the IBM MQ AMS component is installed for the version of IBM MQ that the queue manager is running under, the attribute has a value `ENABLED (MQCAP_SUPPORTED)`. If the IBM MQ AMS component is not installed, the value is `DISABLED (MQCAP_NOT_SUPPORTED)`.

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.

SSLCRLNL
Indicates the namelist of AUTHINFO objects being used for the queue manager for certificate revocation checking.

Only authentication information objects with types of LDAPCRL 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
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 UNIX, Linux, and Windows systems.

SSLEV
Whether SSL events are generated.

SSLFIPS
Whether only FIPS-certified algorithms are to be used if cryptography is executed 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 SSL conversation before the secret key is renegotiated.

z/OS **SSLTASKS**
On z/OS only, indicates the number of server subtasks to use for processing SSL 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.

STATINT
The interval at which statistics monitoring data is written to the monitoring queue. This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

STATMQI
Whether statistics monitoring data is to be collected for the queue manager. This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

STATQ
Whether statistics data is to be collected for queues. This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

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 SSL and 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 IBM MQ Telemetry capability is supported by the queue manager.

For more details of these parameters, see “ALTER QMGR” on page 724.

DISPLAY QMSTATUS

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 Performing local administration tasks using MQSC commands.

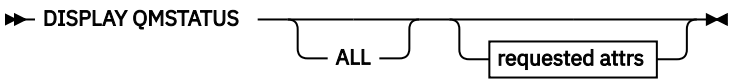
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram

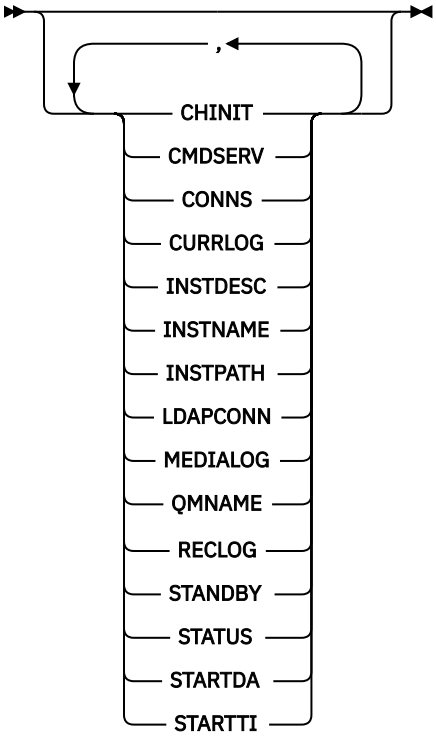
- “Parameter descriptions for DISPLAY QMSTATUS” on page 1135
- “Requested parameters” on page 1135

Synonym: DIS QMSTATUS

DISPLAY QMSTATUS



Requested attrs



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.

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.

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.

INSTDESC
Description of the installation associated with the queue manager. This parameter is not valid on IBM i.

INSTNAME
Name of the installation associated with the queue manager. This parameter is not valid on IBM i.

INSTPATH
Path of the installation associated with the queue manager. This parameter is not valid on IBM i.

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.

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.

QMNAME
The name of the queue manager. This parameter is always returned.

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.

STANDBY
Whether a standby instance is permitted. It is one of the following:

NOPERMIT
Standy instances are not permitted.

PERMIT
Standy instances are permitted.

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.

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).

DISPLAY QSTATUS

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 Performing local administration tasks using MQSC commands.

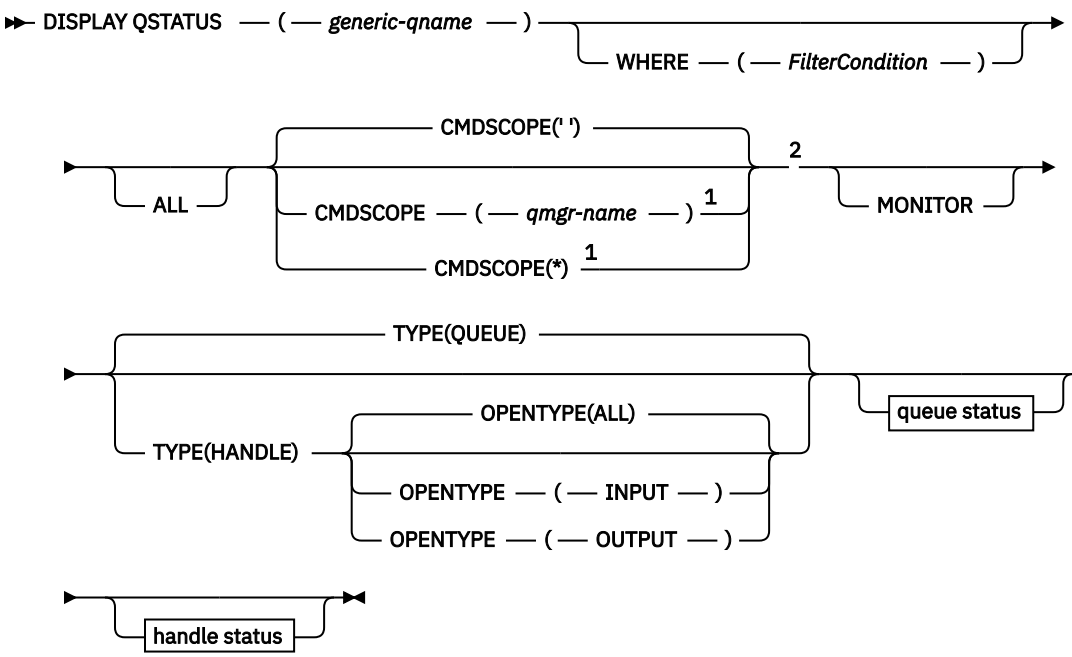
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

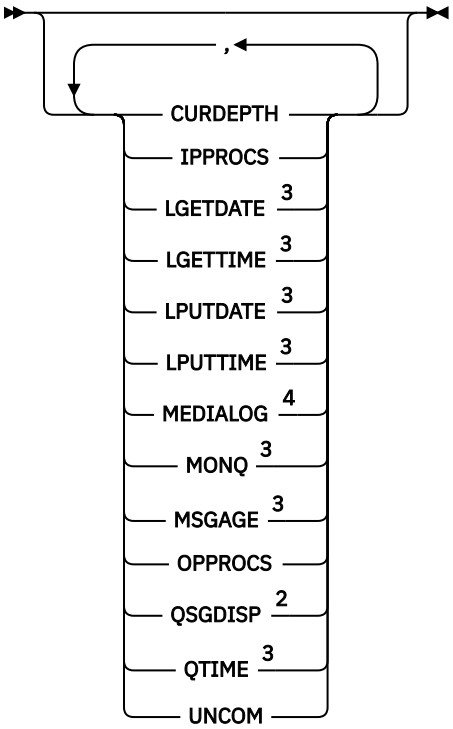
- Syntax diagram
- “Usage notes for DISPLAY QSTATUS” on page 1139
- “Parameter descriptions for DISPLAY QSTATUS” on page 1140
- “Queue status” on page 1142
- “Handle status” on page 1144

Synonym: DIS QS

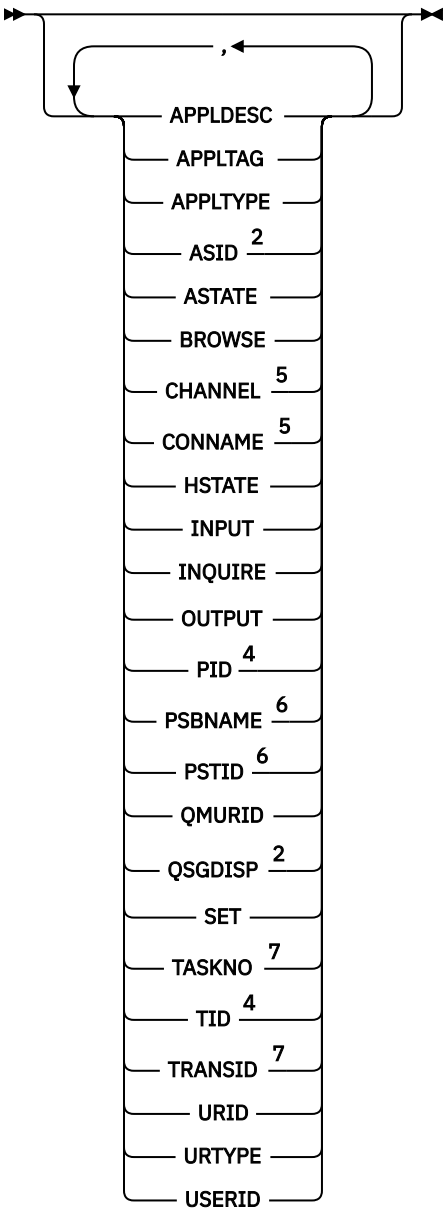
DISPLAY QSTATUS



Queue status



Handle status



Notes:

[1] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[2] Valid on z/OS only.
[3] Also displayed by selection of the MONITOR parameter.
[4] Not valid on z/OS.
[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 is executed when the queue manager is a member of a queue-sharing group. It is valid on z/OS only.

' '
The command is executed on the queue manager on which it was entered. This value is the default.

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 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) z/OS on platforms other than z/OS
- z/OS 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.

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.

z/OS 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.

z/OS 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.

z/OS 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.

z/OS 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 on AIX, HP-UX, Linux, IBM i, Solaris, and Windows.

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
On platforms other than z/OS, 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)

- On platforms other than z/OS, user identifier (USERID parameter) - not returned for APPLTYPE(SYSTEM)
- On platforms other than z/OS, process ID (PID parameter)
- On platforms other than z/OS, thread ID (TID parameter)
- On platforms other than z/OS, application tag (APPLTAG parameter)
- Application type (APPLTYPE parameter)
- On platforms other than z/OS, whether the handle provides input access (INPUT parameter)
- On platforms other than z/OS, whether the handle provides output access (OUTPUT parameter)
- On platforms other than z/OS, whether the handle provides browse access (BROWSE parameter)
- On platforms other than z/OS, whether the handle provides inquire access (INQUIRE parameter)
- On platforms other than z/OS, whether the handle provides set access (SET 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 job name
- UNIX process

  Note: On HP-UX if the process name exceeds 14 characters, only the first 14 characters are shown. On all other platforms if the process name exceeds 28 characters, only the first 28 characters are shown.
- 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.

CONNAME
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.

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

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 Performing local administration tasks using MQSC commands.

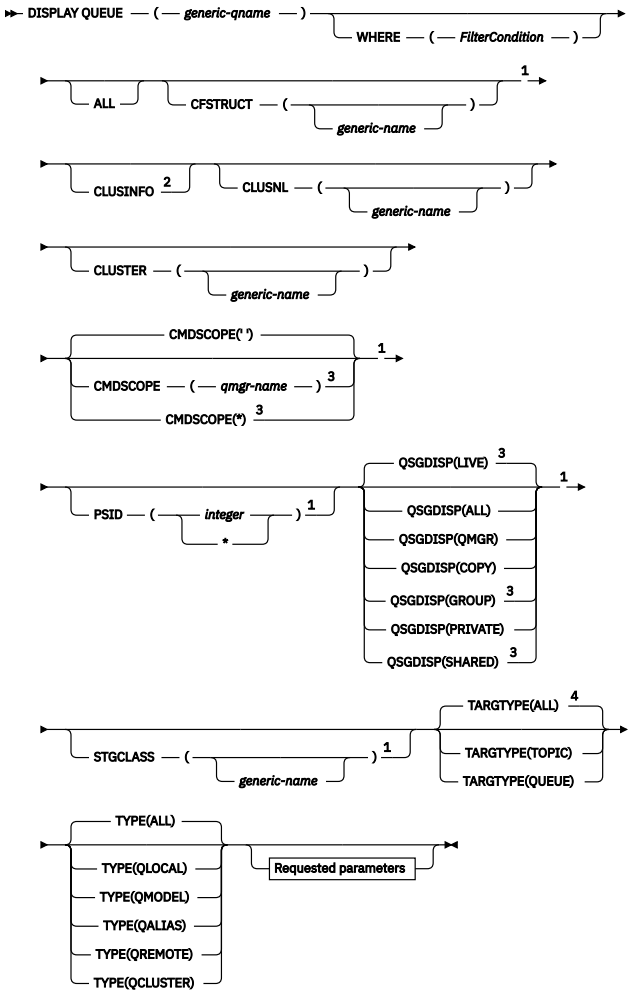
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

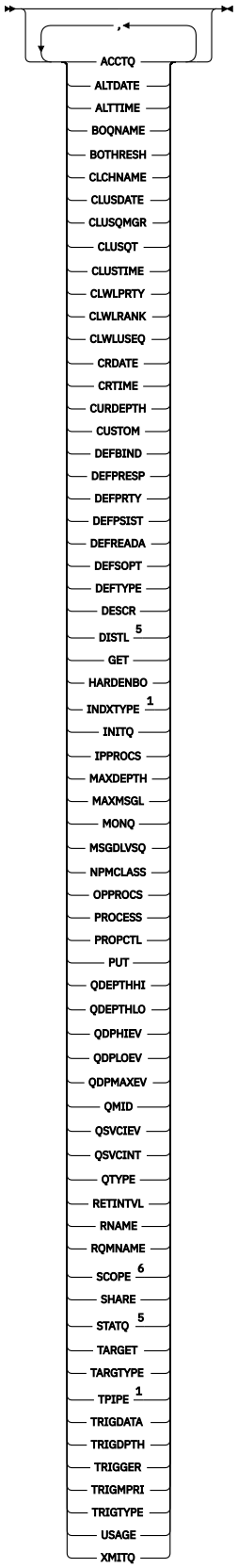
- Syntax diagram
- “Usage notes” on page 1152
- “Parameter descriptions for DISPLAY QUEUE” on page 1152
- “Requested parameters” on page 1156

Synonym: **DIS Q**

DISPLAY QUEUE



Requested parameters



Notes:

[1] Valid only on z/OS.

[2] On z/OS, you cannot issue this from CSQINP2.

[3] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.
[4] Valid only on an alias queue.
[5] Not valid on z/OS.
[6] 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 QSVCIEV parameters as filter keywords. You cannot use z/OS CFSTRUCT, CLUSTER, z/OS PSID, z/OS STGCLASS,, or CLUSNLif 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 AIX, HP-UX, Linux, IBM i, Solaris, and Windows z/OS, and z/OS, this is the default if you do not specify a generic name and do not request any specific attributes.

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 **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.

This parameter is valid only on AIX, HP-UX, Linux, IBM i, Solaris, and Windows z/OS, and z/OS.

z/OS 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.

This parameter is valid only on AIX, HP-UX, Linux, IBM i, Solaris, Windows, and z/OS, and z/OS

Note: z/OS 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.

This parameter is valid only on AIX, HP-UX, Linux, IBM i, Solaris, Windows z/OS, and z/OS.

z/OS **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 or SHARED.

' '
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.

You cannot use **CMDSCOPE** as a filter keyword.

z/OS **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.

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). 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.

TARGTYPE (*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.

On platforms other than z/OS, 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.

*Table 74. Parameters that can be returned by the **DISPLAY QUEUE** command.*

Cross-tabulation of queue parameters and queue types. If the parameter applies to the queue type, the cell contains a check mark.

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
ACCTQ	✓	✓	N/A	N/A	N/A
ALTDATE	✓	✓	✓	✓	✓
ALTTIME	✓	✓	✓	✓	✓
BOQNAME	✓	✓	N/A	N/A	N/A
BOTHRESH	✓	✓	N/A	N/A	N/A
CFSTRUCT	✓	✓	N/A	N/A	N/A
CLCHNAME	✓	✓	N/A	N/A	N/A
CLUSDATE	N/A	N/A	N/A	N/A	✓
CLUSNL	✓	N/A	✓	✓	N/A
CLUSQMGR	N/A	N/A	N/A	N/A	✓
CLUSQT	N/A	N/A	N/A	N/A	✓
CLUSTER	✓	N/A	✓	✓	✓
CLUSTIME	N/A	N/A	N/A	N/A	✓
CLWLPRTY	✓	N/A	✓	✓	✓
CLWLRANK	✓	N/A	✓	✓	✓
CLWLUSEQ	✓	N/A	N/A	N/A	N/A
CRDATE	✓	✓	N/A	N/A	N/A
CRTIME	✓	✓	N/A	N/A	N/A
CURDEPTH	✓	N/A	N/A	N/A	N/A
CUSTOM	✓	✓	✓	✓	✓
DEFBIND	✓	N/A	✓	✓	✓
DEFPRESP	✓	✓	✓	✓	✓
DEFPRTY	✓	✓	✓	✓	✓
DEFPSIST	✓	✓	✓	✓	✓
DEFREADA	✓	✓	✓	N/A	N/A
DEFSOPT	✓	✓	N/A	N/A	N/A

*Table 74. Parameters that can be returned by the **DISPLAY QUEUE** command.*

Cross-tabulation of queue parameters and queue types. If the parameter applies to the queue type, the cell contains a check mark.

(continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
DEFTYPE	✓	✓	N/A	N/A	N/A
DESCR	✓	✓	✓	✓	✓
DISTL	✓	✓	N/A	N/A	N/A
GET	✓	✓	✓	N/A	N/A
HARDENBO	✓	✓	N/A	N/A	N/A
INDXTYPE	✓	✓	N/A	N/A	N/A
INITQ	✓	✓	N/A	N/A	N/A
IPPROCS	✓	N/A	N/A	N/A	N/A
MAXDEPTH	✓	✓	N/A	N/A	N/A
MAXMSGL	✓	✓	N/A	N/A	N/A
MONQ	✓	✓	N/A	N/A	N/A
MSGDLVSQ	✓	✓	N/A	N/A	N/A
NPMCLASS	✓	✓	N/A	N/A	N/A
OPPROCS	✓		N/A	N/A	N/A
PROCESS	✓	✓	N/A	N/A	N/A
PROPCTL	✓	✓	✓	N/A	N/A
PSID	✓	N/A	N/A	N/A	N/A
PUT	✓	✓	✓	✓	✓
QDEPTHHI	✓	✓	N/A	N/A	N/A
QDEPTHLO	✓	✓	N/A	N/A	N/A
QDPHIEV	✓	✓	N/A	N/A	N/A
QDPLOEV	✓	✓	N/A	N/A	N/A
QDPMAXEV	✓	✓	N/A	N/A	N/A
QMID	N/A	N/A	N/A	N/A	✓
QSGDISP	✓	✓	✓	✓	✓
QSVCIEV	✓	✓	N/A	N/A	N/A

*Table 74. Parameters that can be returned by the **DISPLAY QUEUE** command.*

Cross-tabulation of queue parameters and queue types. If the parameter applies to the queue type, the cell contains a check mark.

(continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
QSVCINT	✓	✓	N/A	N/A	N/A
QTYPE	✓	✓	✓	✓	✓
RETINTVL	✓	✓	N/A	N/A	N/A
RNAME	N/A	N/A	N/A	✓	N/A
RQMNAME	N/A	N/A	N/A	✓	N/A
SCOPE	✓	N/A	✓	✓	N/A
SHARE	✓	✓	N/A	N/A	N/A
STATQ	✓	✓	N/A	N/A	N/A
STGCLASS	✓	✓	N/A	N/A	N/A
TARGET	N/A	N/A	✓	N/A	N/A
TARGTYPE	N/A	N/A	✓	N/A	N/A
TPIPE	✓	N/A	N/A	N/A	N/A
TRIGDATA	✓	✓	N/A	N/A	N/A
TRIGDPTH	✓	✓	N/A	N/A	N/A
TRIGGER	✓	✓	N/A	N/A	N/A
TRIGMPRI	✓	✓	N/A	N/A	N/A
TRIGTYPE	✓	✓	N/A	N/A	N/A
USAGE	✓	✓	N/A	N/A	N/A
XMITQ	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.

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.

CLUSQMGR
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 AIX, HP-UX, Linux, IBM i, Solaris, and Windows.)

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.

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.

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.

QSVCINT
Service interval event generation threshold.

QTYPE
Queue type.

On AIX, HP-UX, Linux, IBM i, Solaris, Windows, and z/OS, the queue type is always displayed.

On AIX, HP-UX, Linux, IBM i, Solaris, and Windows, 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.

TARGET
This parameter requests that the base object name of an aliased queue is displayed.

TARGTYPE
This parameter requests that the target (base) type of an aliased queue is displayed.

TPIPE
The TPIPE names used for communication with OTMA via the IBM MQ - IMS bridge if the bridge is active. This parameter is supported only on z/OS.

z/OS For more information about TPIPEs, see Controlling the IMS bridge.

TRIGDATA
Trigger data.

TRIGDPTH
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 913.

DISPLAY SBSTATUS

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 Performing local administration tasks using MQSC commands.

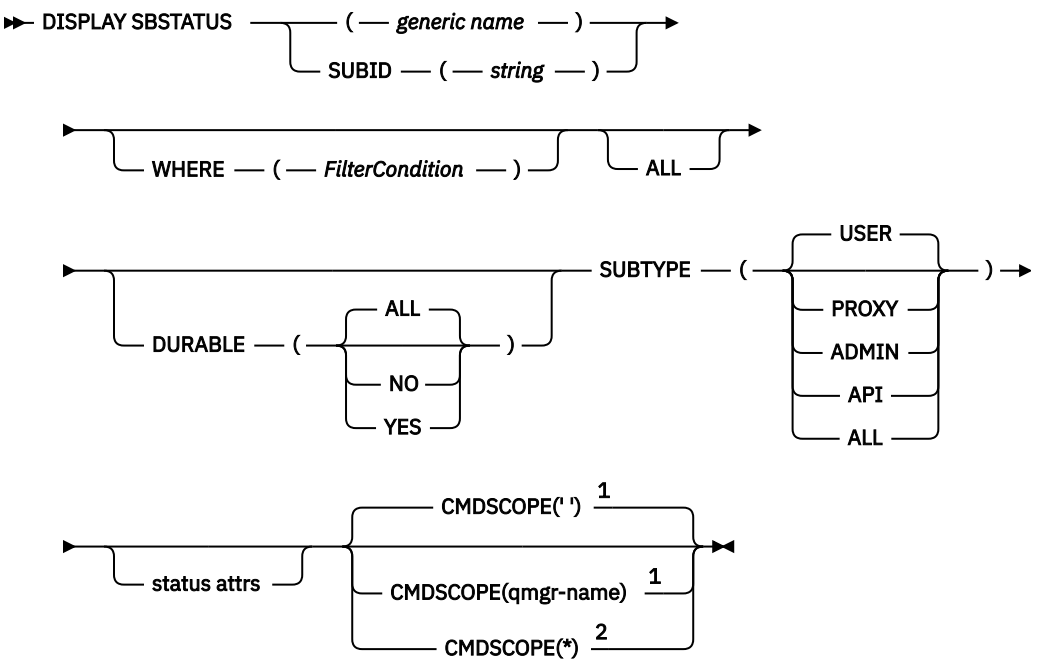
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

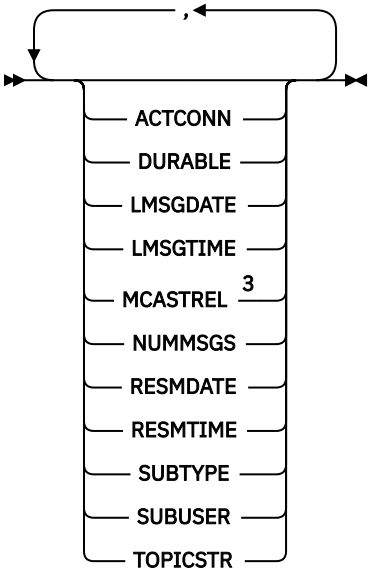
- Syntax diagram
- "Parameter descriptions for DISPLAY SBSTATUS" on page 1165
- "Requested parameters" on page 1167

Synonym: DIS SBSTATUS

DISPLAY SBSTATUS



Status attributes



Notes:

1 Valid only on z/OS.
2 Valid only on z/OS when the queue manager is a member of a queue-sharing group.
3 Not valid on z/OS.

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.

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 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 processed on the queue manager on which it was entered. This is the default value.

qmgr-name

The command is processed 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 processed 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 the user can switch on 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 owing 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" on page 952.

z/OS DISPLAY SECURITY

Use the MQSC command DISPLAY SECURITY to display the current settings for the security parameters.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			CR

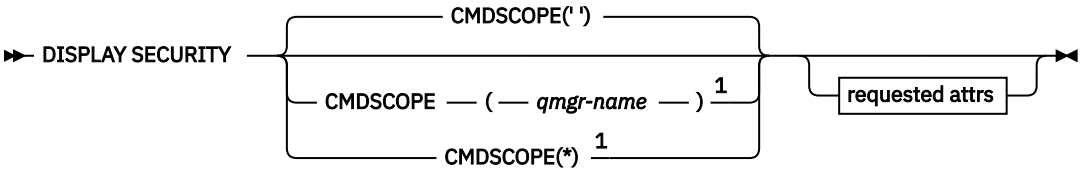
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Parameter descriptions for DISPLAY SECURITY" on page 1169

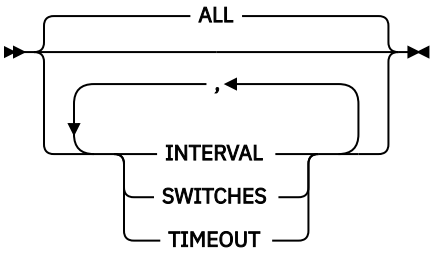
Note: From IBM WebSphere MQ 7.0 onwards, this command is no longer allowed to be issued from CSQINP1 or CSQINP2 on z/OS.

Synonym: DIS SEC

DISPLAY SECURITY



Requested attrs



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

Parameter descriptions for DISPLAY SECURITY

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 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.

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" on page 785 for details of the TIMEOUT and INTERVAL parameters.

Related tasks

Displaying security status

DISPLAY SERVICE

Use the MQSC command DISPLAY SERVICE to display information about a service.

Using MQSC commands

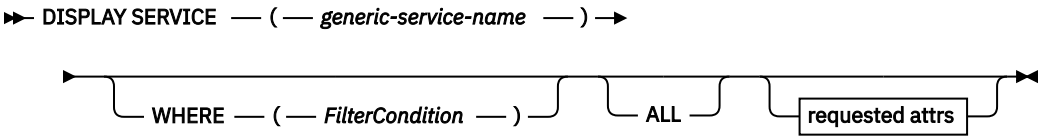
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

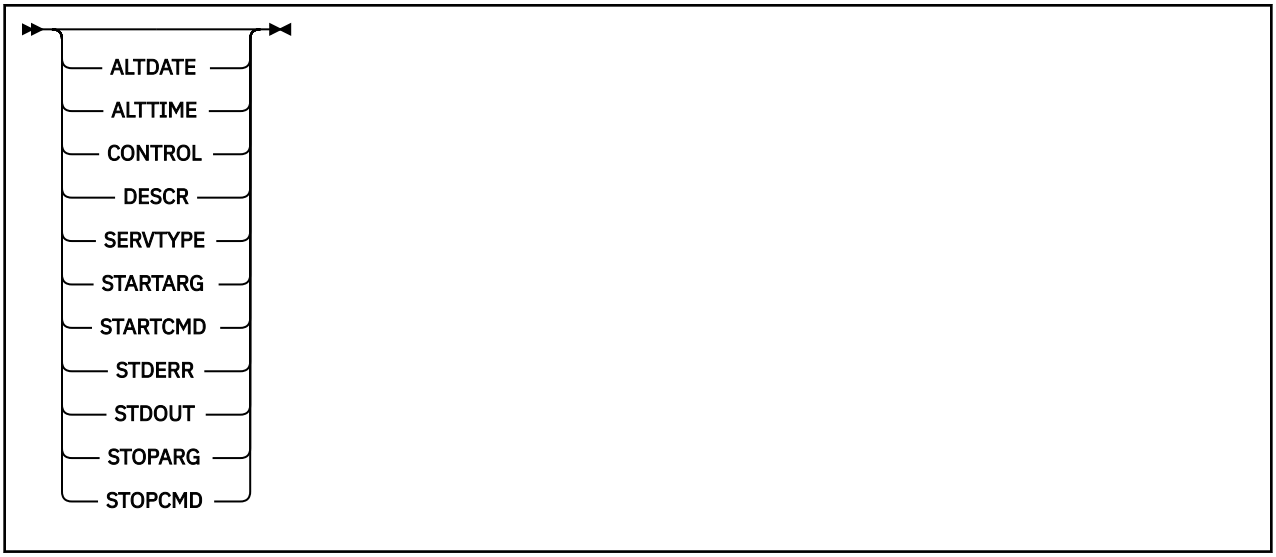
- Syntax diagram
- "Keyword and parameter descriptions for DISPLAY SERVICE" on page 1171
- "Requested parameters" on page 1172

Synonym:

DISPLAY SERVICE



Requested attrs



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" on page 945.

z/OS DISPLAY SMDS

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

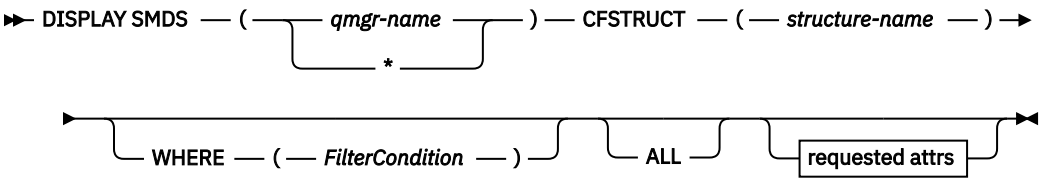
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

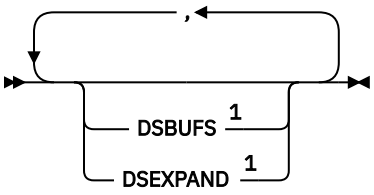
- Syntax diagram
- "Parameter descriptions for DISPLAY SMDS" on page 1174
- "Usage notes for DISPLAY SMDSCONN" on page 1177

Synonym:

DISPLAY SMDS



Requested attrs



Notes:

[1] For more information about this parameter, see Planning your coupling facility and offload storage environment.

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

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

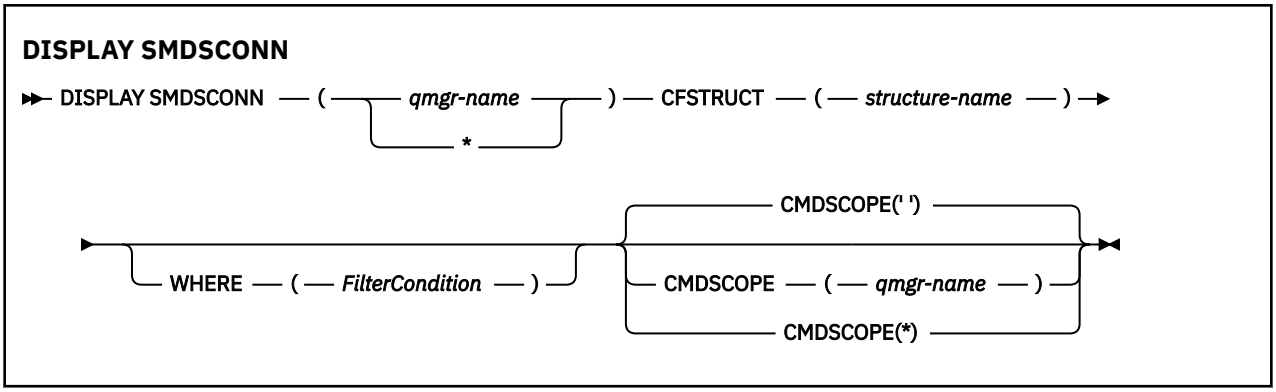
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for DISPLAY SMDSCONN” on page 1176
- “Usage notes for DISPLAY SMDSCONN” on page 1177

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 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.

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" on page 1300
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" on page 1320
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.

z/OS DISPLAY STGCLASS

Use the MQSC command DISPLAY STGCLASS to display information about storage classes.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

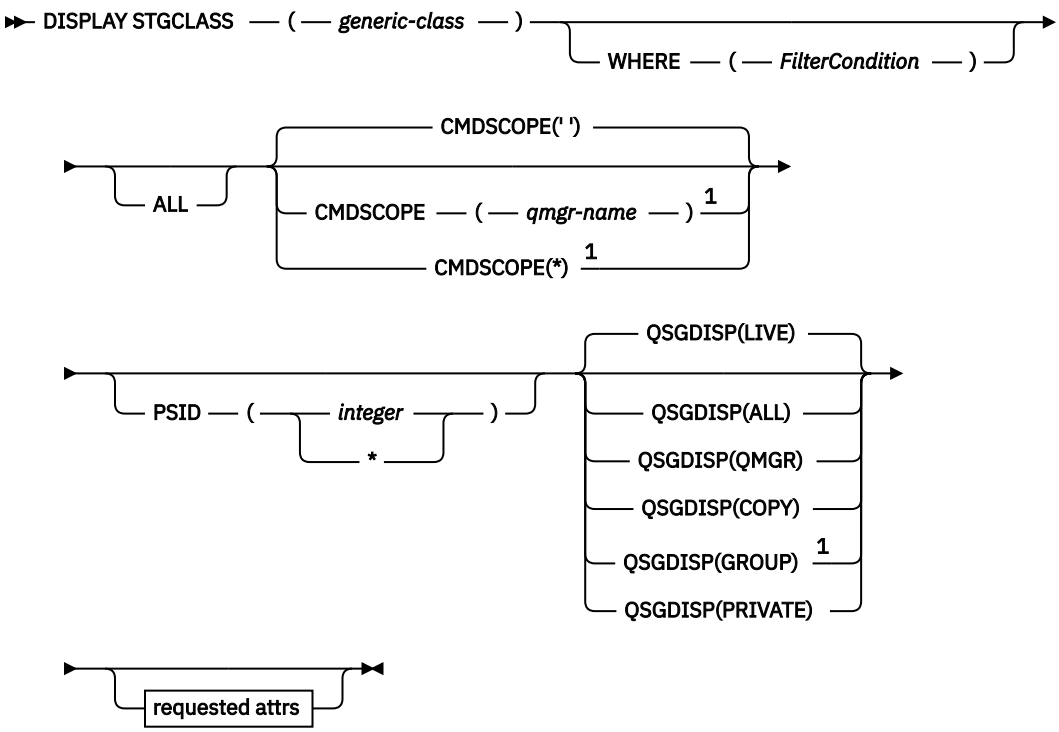
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

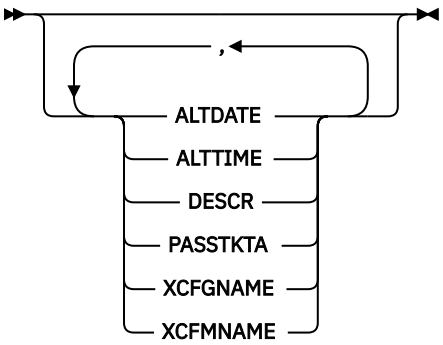
- Syntax diagram
- "Parameter descriptions for DISPLAY STGCLASS" on page 1180
- "Requested parameters" on page 1182

Synonym: DIS STC

DISPLAY STGCLASS



Requested attrs



Notes:

[1] 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 is executed 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 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.

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” on page 911.

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`.

ALTTIME
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” on page 948.

DISPLAY SUB

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 Performing local administration tasks using MQSC commands.

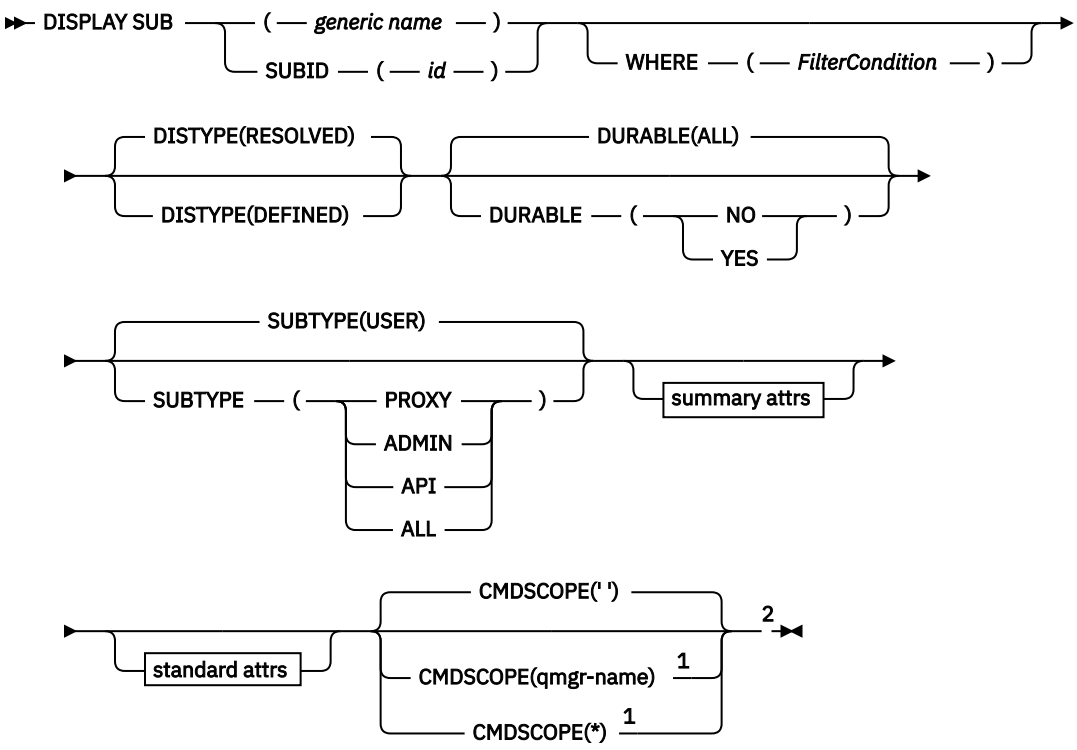
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

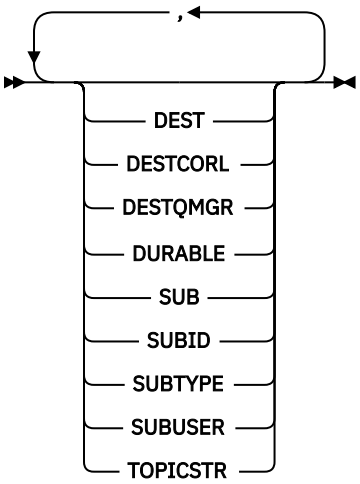
- Syntax diagram
- “Usage notes for DISPLAY SUB” on page 1184
- “Parameter descriptions for DISPLAY SUB” on page 1185

Synonym: DIS SUB

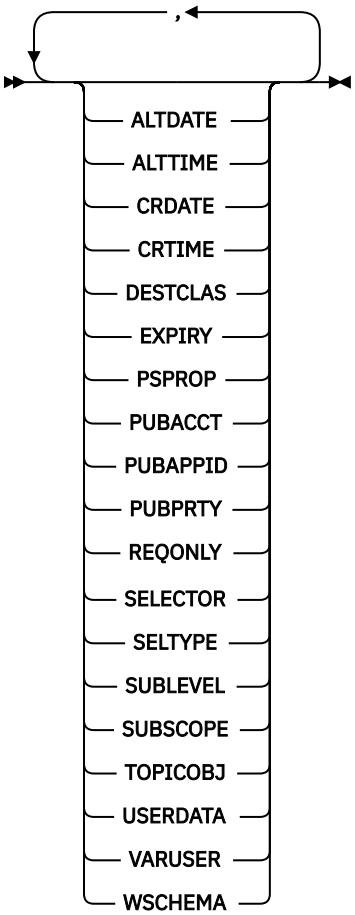
DISPLAY SUB



summary attributes



standard 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.

Usage notes for DISPLAY SUB

1. 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 distributed platforms using runmqsc, these non-printable characters will be 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 are valid forms:

```
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: 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.

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.

On AIX, HP-UX, Linux, IBM i, Solaris, Windows, and z/OS, 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.

z/OS **CMDSCOPE**
This parameter applies to z/OS only and specifies how the command is processed when the queue manager is a member of a queue-sharing group.

' '
The command is processed on the queue manager on which it was entered. This is the default value.

qmgr-name
The command is processed 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 processed 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 ' 00000000000000000000000000000000000000000000000000000000 ' (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.

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 recreate 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.

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.

V 8.0.0.6 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 will either be 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.

V 8.0.0.7 From IBM MQ 8.0.0, Fix Pack 7, 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` in the JmsConstants interface with the method `javax.jms.Message.getStringProperty(java.lang.String)`. 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.

WSCHEMA
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.

DISPLAY SVSTATUS

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

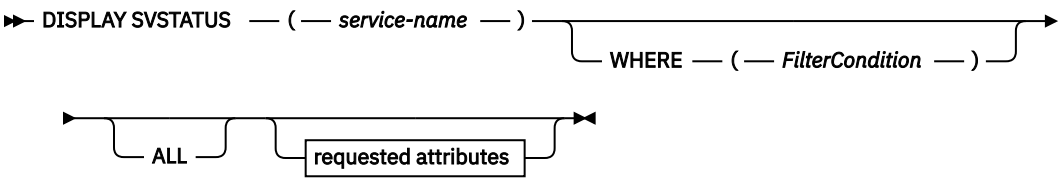
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

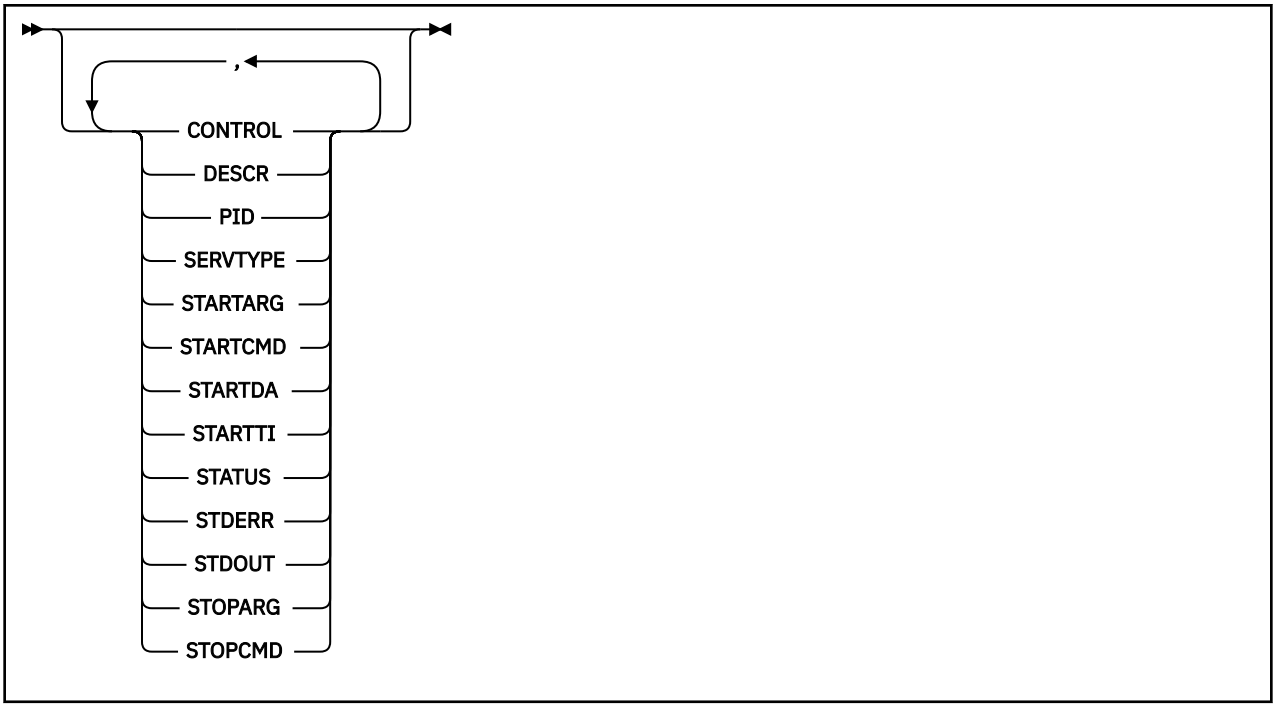
- Syntax diagram
- “Keyword and parameter descriptions for DISPLAY SVSTATUS” on page 1191
- “Requested parameters” on page 1192

Synonym:

DISPLAY SVSTATUS



Requested attributes



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 details of these parameters, see "DEFINE SERVICE" on page 945.

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

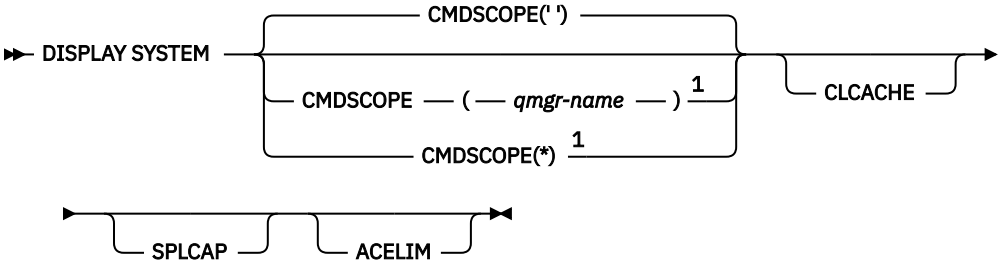
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

You can issue this command from sources 12CR. For an explanation of the source symbols, see Sources from which you can issue MQSC commands on z/OS.

- Syntax diagram
- "Usage notes for DISPLAY SYSTEM" on page 1193
- "Parameter descriptions for DISPLAY SYSTEM" on page 1194

Synonym: DIS SYSTEM

DISPLAY SYSTEM



Notes:

1 Valid only when the queue manager is a member of a queue sharing group.

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 (QINDXBLD).
- 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 (ROUTCDE).
- Whether SMF accounting data is collected when IBM MQ is started (SMFACCT).
- Whether SMF statistics are collected when IBM MQ is started (SMFSTAT).
- The time, in minutes, between each gathering of statistics data (STATIME).
- 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.
- Whether batch jobs can currently be swapped out during some MQ API calls or not (CONNSWAP).

  Attention: From IBM MQ 9.0, this keyword has no effect.
- 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

Use the MQSC command DISPLAY TCLUSTER to display the attributes of the IBM MQ cluster topic object.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

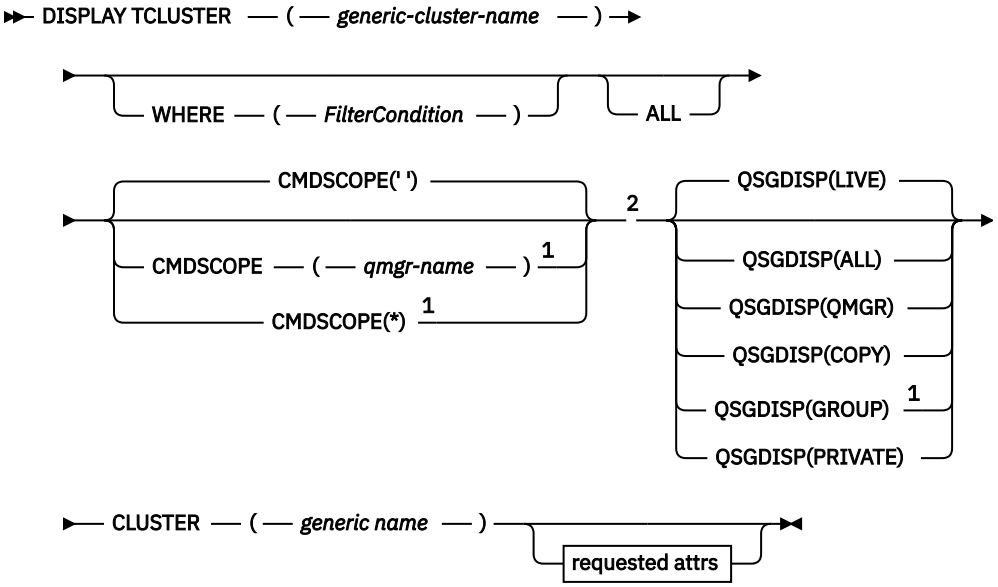
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

The DISPLAY TCLUSTER command produces the same output as the DISPLAY TOPIC TYPE(CLUSTER) command.

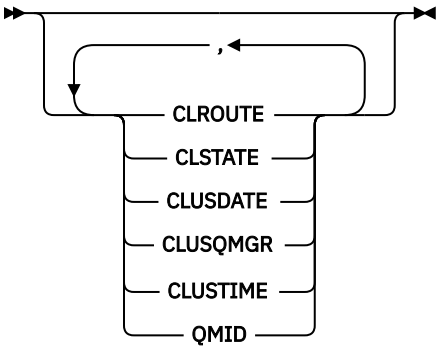
See "DISPLAY TOPIC" on page 1201 for further information.

Synonym: DIS TOPIC

DISPLAY TCLUSTER



Requested attrs



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.

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-cluster-name)
The name of the administrative cluster 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 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. This value 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 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 distributed platforms using the **runmqsc** command, these non-printable characters are displayed as dots.

Related reference
"DISPLAY TPSTATUS" on page 1210
Use the MQSC command DISPLAY TPSTATUS to display the status of one or more topics in a topic tree.

"DISPLAY TOPIC" on page 1201
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

Use the MQSC command DISPLAY THREAD to display information about active and in-doubt threads.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

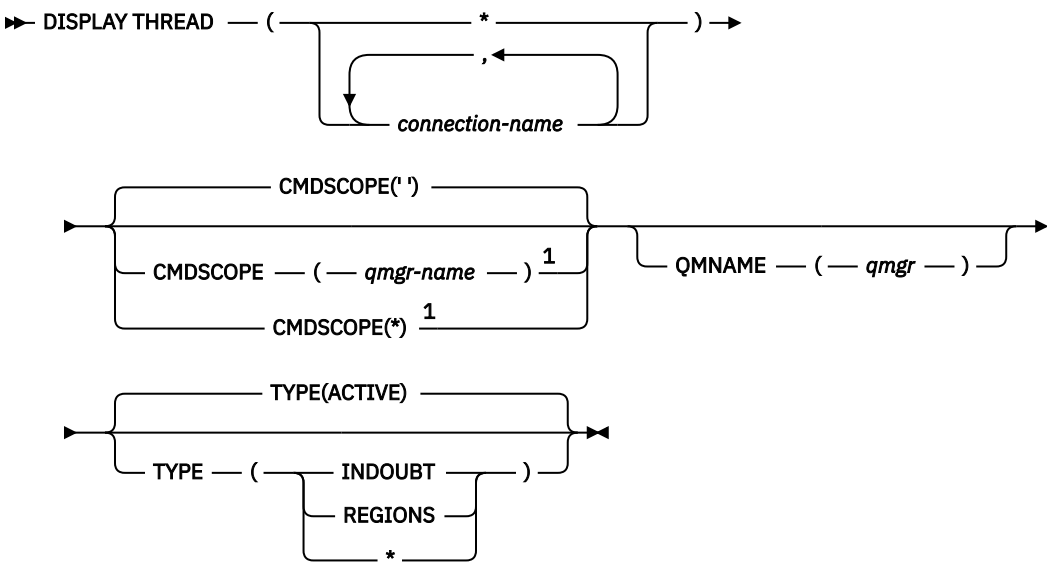
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 1200
- "Parameter descriptions for DISPLAY THREAD" on page 1200

Synonym: DIS THD

DISPLAY THREAD



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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 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.

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

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 Performing local administration tasks using MQSC commands.

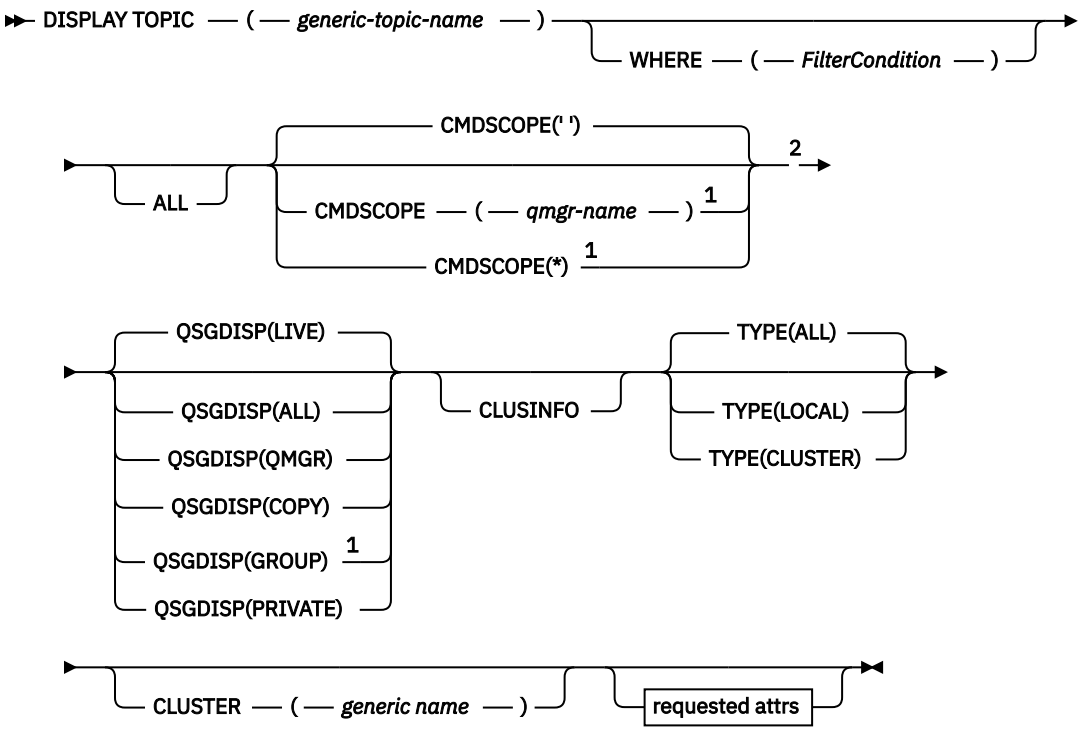
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

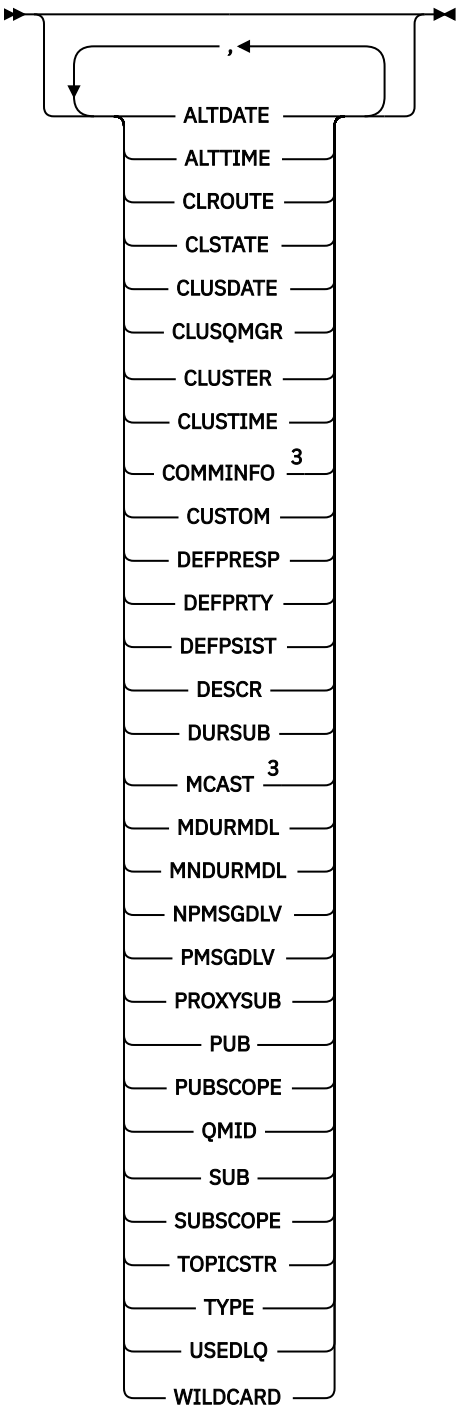
- Syntax diagram
- “Usage notes for DISPLAY TOPIC” on page 1203
- “Parameter descriptions for DISPLAY TOPIC” on page 1204
- “Requested parameters” on page 1207

Synonym: DIS TOPIC

DISPLAY TOPIC



Requested attrs



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.

Usage notes for DISPLAY TOPIC

1. z/OS 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. On distributed platforms 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 DISPLAY TOPIC TYPE(CLUSTER) command. 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: 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 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. This value 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 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.

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.

z/OS 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.

z/OS 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" on page 958.

Table 75. Parameters that can be returned by the DISPLAY TOPIC command

	Local topic	Cluster topic
ALTDATE	✓	✓
ALTTIME	✓	✓
CLROUTE	✓	✓
CLSTATE		✓
CLUSDATE		✓
CLUSQMGR		✓
CLUSTER	✓	✓
CLUSTIME		✓
COMMINFO	✓	
CUSTOM	✓	✓
DEFPRTY	✓	✓
DEFPSIST	✓	✓
DEFPRESP	✓	✓
DESCR	✓	✓
DURSUB	✓	✓
MCAST	✓	
MDURMDL	✓	✓
MNDURMDL	✓	✓
NPMSGDLV	✓	✓

Table 75. Parameters that can be returned by the DISPLAY TOPIC command (continued)		
	Local topic	**Cluster topic**
PMSGDLV	✓	✓
PROXYSUB	✓	✓
PUB	✓	✓
PUBSCOPE	✓	✓
QMID		✓
SUB	✓	✓
SUBSCOPE	✓	✓
TOPICSTR	✓	✓
TYPE	✓	✓
USEDLQ	✓	
WILDCARD	✓	✓

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`.

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" on page 958.

Related reference

"DISPLAY TPSTATUS" on page 1210
Use the MQSC command DISPLAY TPSTATUS to display the status of one or more topics in a topic tree.

DISPLAY TPSTATUS

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 Performing local administration tasks using MQSC commands.

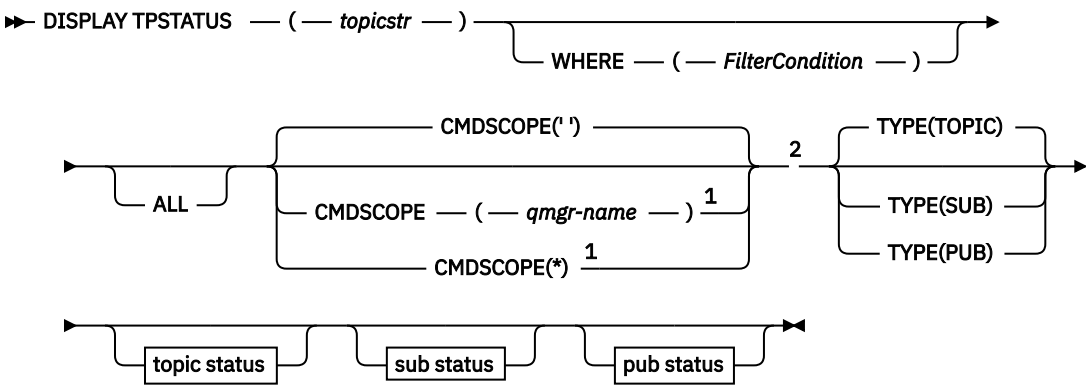
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

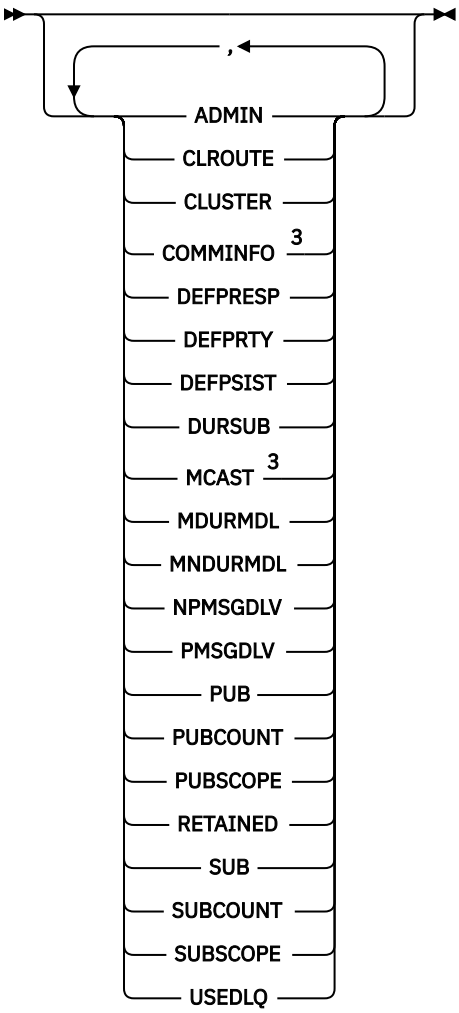
- Syntax diagram
- "Usage notes for DISPLAY TPSTATUS" on page 1212
- "Parameter descriptions for DISPLAY TPSTATUS" on page 1212
- "Topic status parameters" on page 1214
- "Sub status parameters" on page 1216
- "Pub status parameters" on page 1217

Synonym: DIS TPS

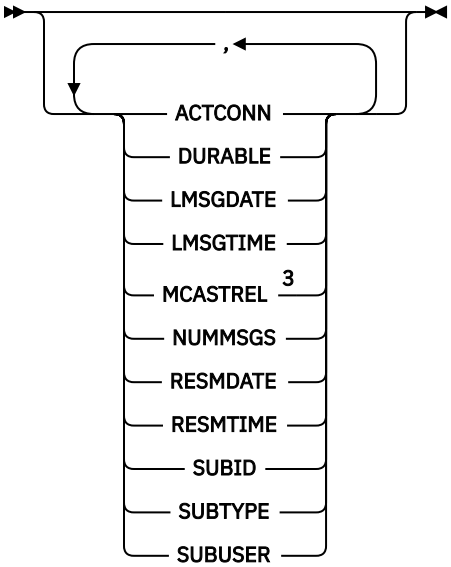
DISPLAY TPSTATUS



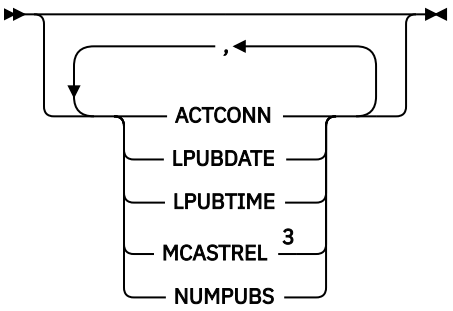
Topic status



Sub status



Pub status



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.

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. On distributed platforms 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.

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 ths 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 the user can switch on 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 the user can switch on 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.

z/OS DISPLAY TRACE

Use the MQSC command DISPLAY TRACE to display a list of active traces.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

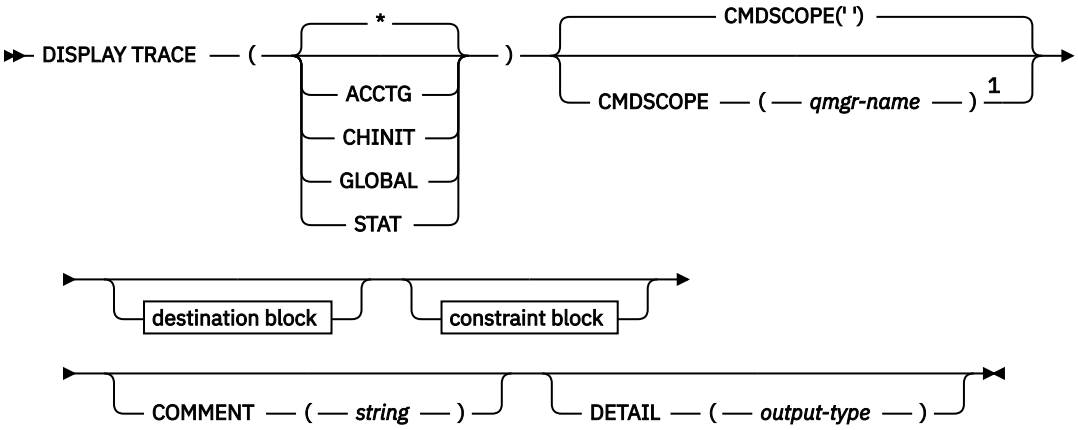
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

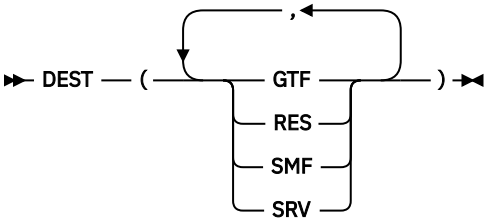
- Syntax diagram
- “Parameter descriptions for DISPLAY TRACE” on page 1218
- “Destination block” on page 1219
- “Constraint block” on page 1219

Synonym: DIS TRACE

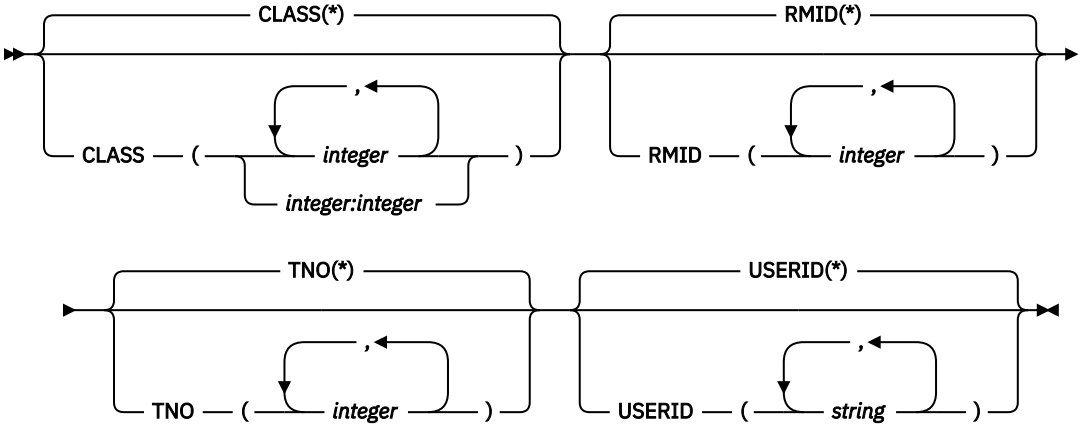
DISPLAY TRACE



Destination block



Constraint block



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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 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 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.

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” on page 1301 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” on page 1301 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

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

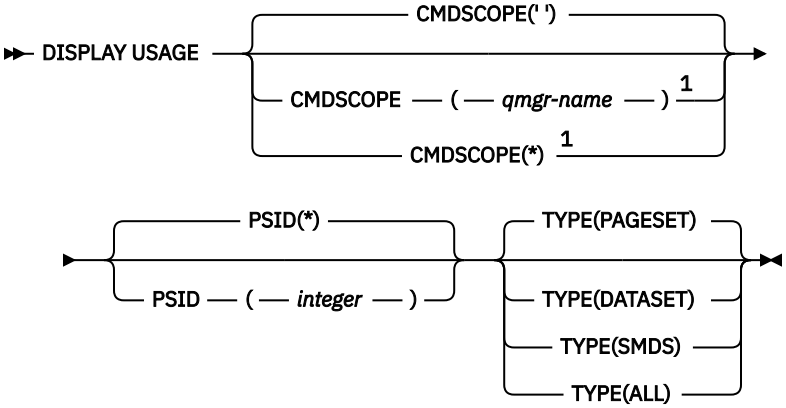
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

Attention: From IBM MQ 8.0, the output of the DISPLAY USAGE command includes message CSQI065I `+MP11 Buffer pool attributes`, instead of message CSQP001I `+MG11 Buffer pool 0 has 25000 buffers`.

- Syntax diagram
- “Parameter descriptions for DISPLAY USAGE” on page 1221

Synonym: DIS USAGE

DISPLAY USAGE



Notes:

1 Valid only when the queue manager is a member of a queue-sharing group.

Parameter descriptions for DISPLAY USAGE

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.

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 pageset 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.

z/OS MOVE QLOCAL

Use the MQSC command MOVE QLOCAL to move all the messages from one local queue to another.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

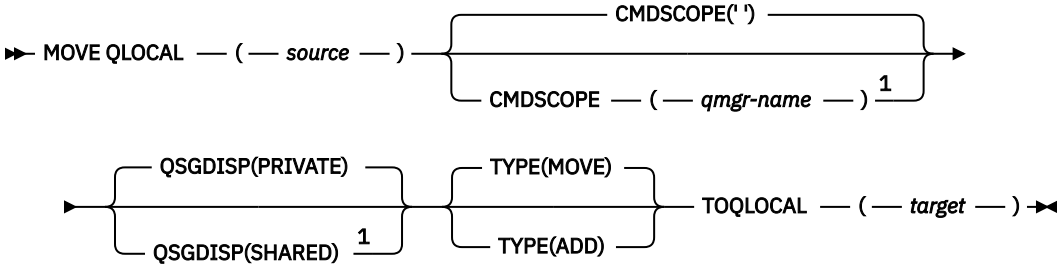
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for MOVE QLOCAL" on page 1222
- "Parameter descriptions for MOVE QLOCAL" on page 1223

Synonym: MOVE QL

MOVE QLOCAL



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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.

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 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.

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

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 Performing local administration tasks using MQSC commands.

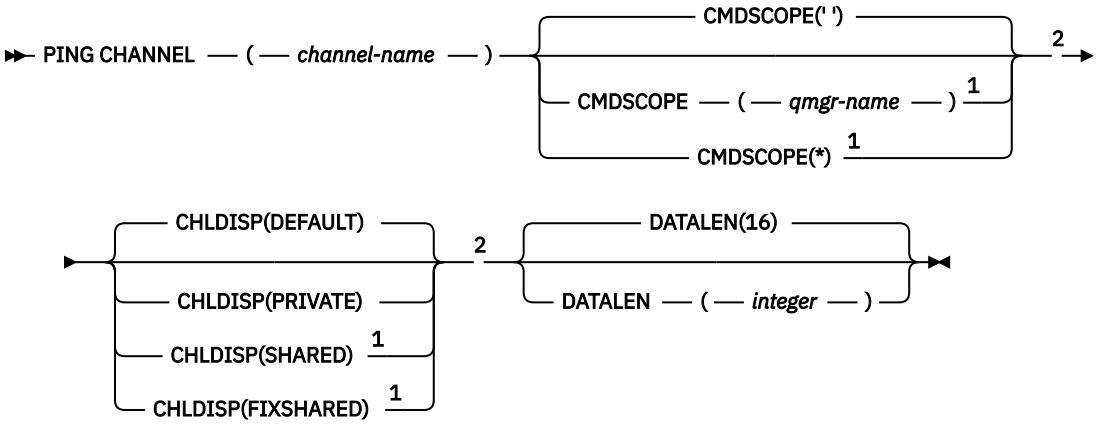
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 1225
- "Parameter descriptions for PING CHANNEL" on page 1225

Synonym: PING CHL

PING CHANNEL



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.
[2] Valid only on z/OS.

Usage notes

1. z/OS 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 (CLUSSDR) 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.

z/OS **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.

If CHLDISP is set to SHARED, CMDSCOPE must be blank or the local queue manager.

' '
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 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.

Note: The '*' option is not permitted if CHLDISP is FIXSHARED.

z/OS **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.

Table 76. CHLDISP and CMDSCOPE for PING CHANNEL

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)	CMDSCOPE(*)
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

Table 76. CHLDISP and CMDSCOPE for PING CHANNEL (continued)			
CHLDISP	**CMDSCOPE( ) or CMDSCOPE (local-qmgr)**	**CMDSCOPE (qmgr-name)**	**CMDSCOPE(*)**
SHARED	Ping a shared channel on the most suitable queue manager in the group This might automatically generate a command using CMDSCOPE 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	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.

PING QMGR

Use the MQSC command PING QMGR to test whether the queue manager is responsive to commands.

Using MQSC commands

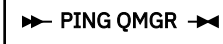
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- "Usage notes" on page 1228

Synonym: 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.

PURGE CHANNEL

V 8.0.0.4 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

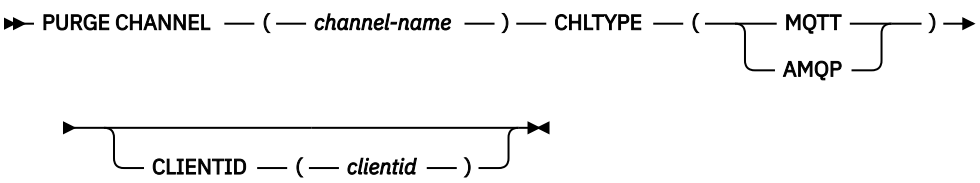
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	AIX and Linux	Windows	z/OS
	✓	✓	

- Syntax diagram
- “Parameter descriptions for PURGE CHANNEL” on page 1228

Synonym: None

PURGE CHANNEL



Parameter descriptions for PURGE CHANNEL

(channel name)

V 8.0.0.4 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.

V 8.0.0.4 The value must be either MQTT or AMQP.

CLIENTID ***(string)***

Client identifier. V 8.0.0.4 The client identifier is a 23 byte string that identifies an IBM 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 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

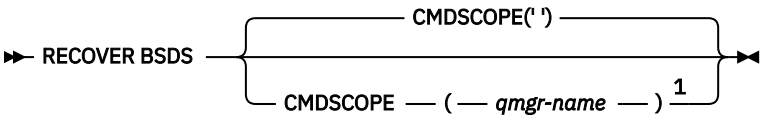
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

You can issue this command from sources CR. For an explanation of the source symbols, see Sources from which you can issue MQSC commands on z/OS.

- Syntax diagram
- "Usage notes for RECOVER BSDS" on page 1229
- "Keyword and parameter descriptions for RECOVER BSDS" on page 1229

Synonym: REC BSDS

RECOVER BSDS



Notes:

1 Valid only when the queue manager is a member of a queue-sharing group.

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

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

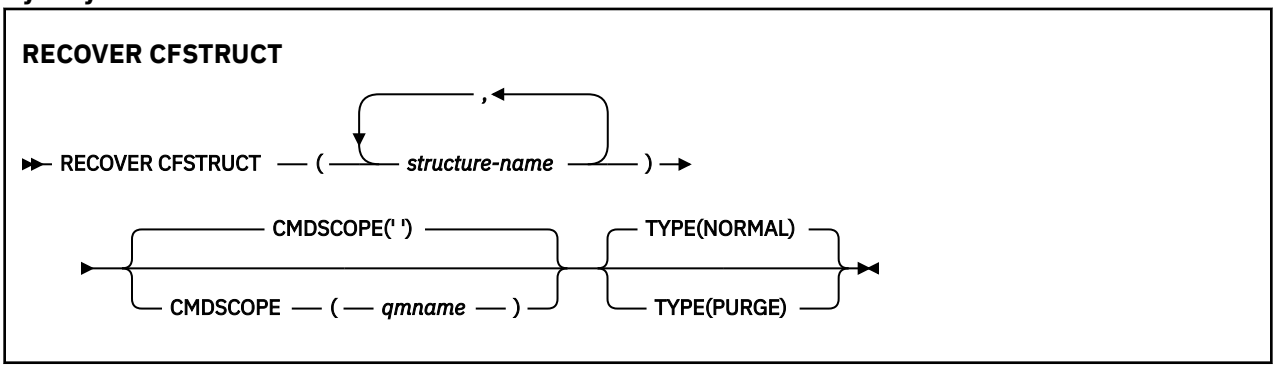
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for RECOVER CFSTRUCT” on page 1230
- “Keyword and parameter descriptions for RECOVER CFSTRUCT” on page 1231

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 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.

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

Use the MQSC command **REFRESH CLUSTER** to discard all locally held cluster information and force it to be rebuilt. The command also processes any auto-defined 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 Performing local administration tasks using MQSC commands.

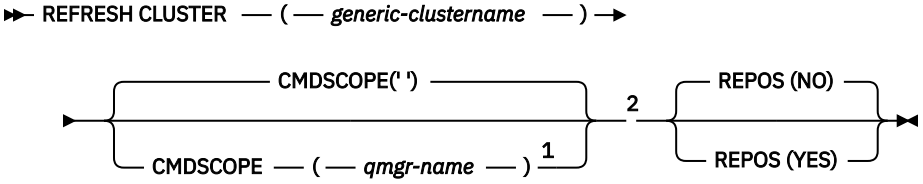
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for REFRESH CLUSTER" on page 1232
- "Parameter descriptions for REFRESH CLUSTER" on page 1234

Synonym: REF CLUSTER

REFRESH CLUSTER



Notes:

1 Valid only when the queue manager is a member of a queue-sharing group.
2 Valid only on z/OS.

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 recreated.
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 UNIX systems, the command is valid only on AIX, HP-UX, Linux, and Solaris.
11. z/OS On z/OS, the command fails if the channel initiator is not started.

12. 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.

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 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. ' ' 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. 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
Application issues seen when running REFRESH CLUSTER
REFRESH CLUSTER considerations for publish/subscribe clusters
Clustering: Using REFRESH CLUSTER best practices

REFRESH QMGR

Use the MQSC command REFRESH QMGR to perform special operations on queue managers.

Using MQSC commands

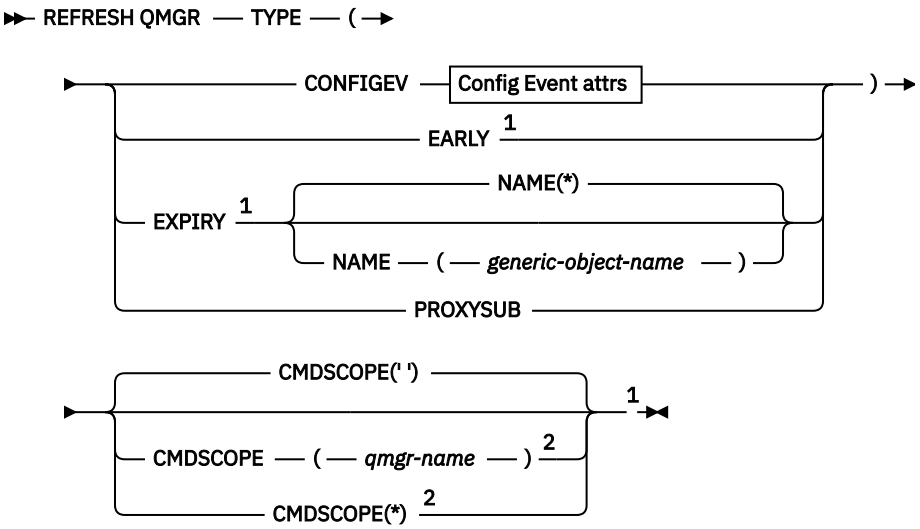
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	See "Using REFRESH QMGR on z/OS" on page 1236

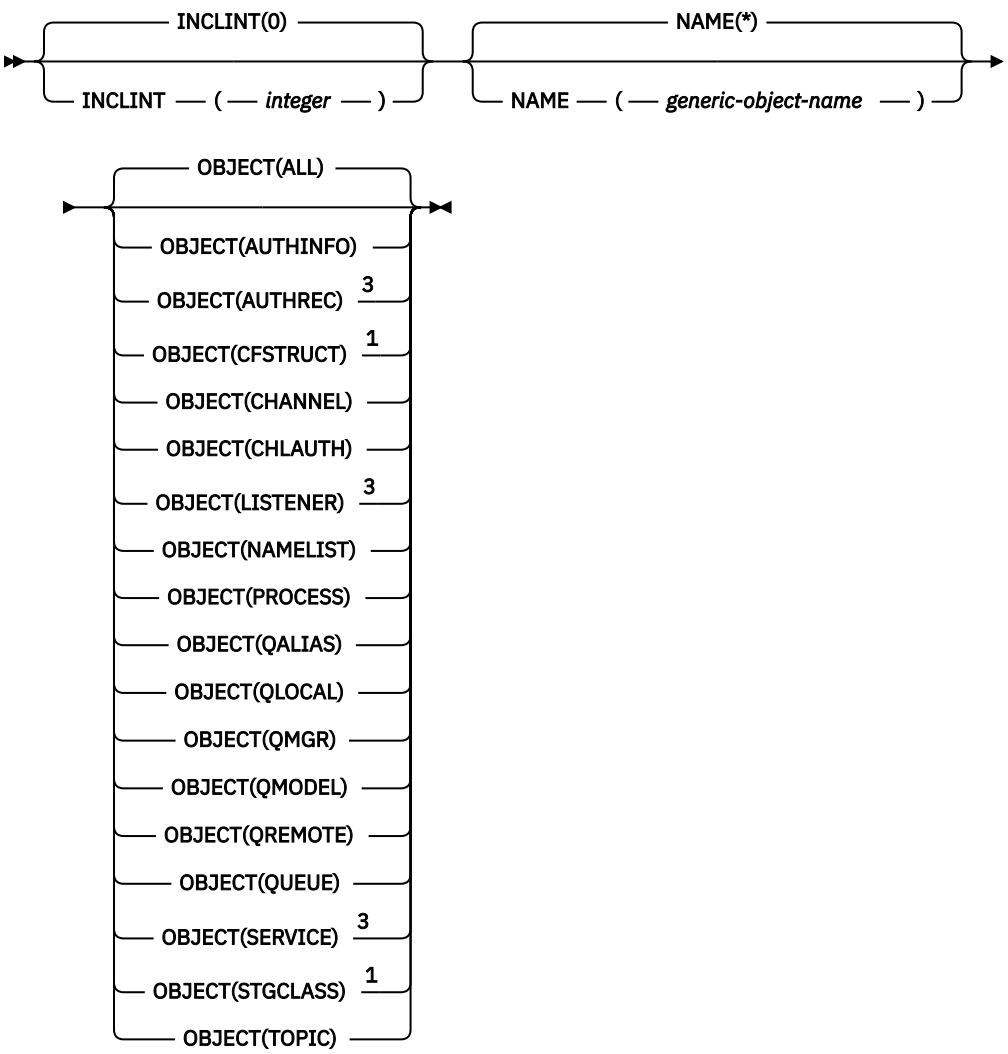
- Syntax diagram
- z/OS "Using REFRESH QMGR on z/OS" on page 1236
- "Usage Notes for REFRESH QMGR" on page 1237
- "Parameter descriptions for REFRESH QMGR" on page 1237

Synonym: None

REFRESH QMGR



Config Event attrs



Notes:

[1] Valid only on z/OS.
[2] Valid only when the queue manager is a member of a queue-sharing group.
[3] Not valid on z/OS.

Using REFRESH QMGR on z/OS

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 “Using commands on z/OS” on page 629.

Command	Command Sources	Notes
REFRESH QMGR TYPE(CONFIGEV)	2CR	
REFRESH QMGR TYPE(EARLY)	C	Queue manager must not be active.
REFRESH QMGR TYPE(EXPIRY)	2CR	

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. z/OS 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. z/OS 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. V 8.0.0.4 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

z/OS **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.

' '

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.

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 ALTDATE 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.

z/OS See Task 3: 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 EXPRYINT 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

Use the MQSC command REFRESH SECURITY to perform a security refresh.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

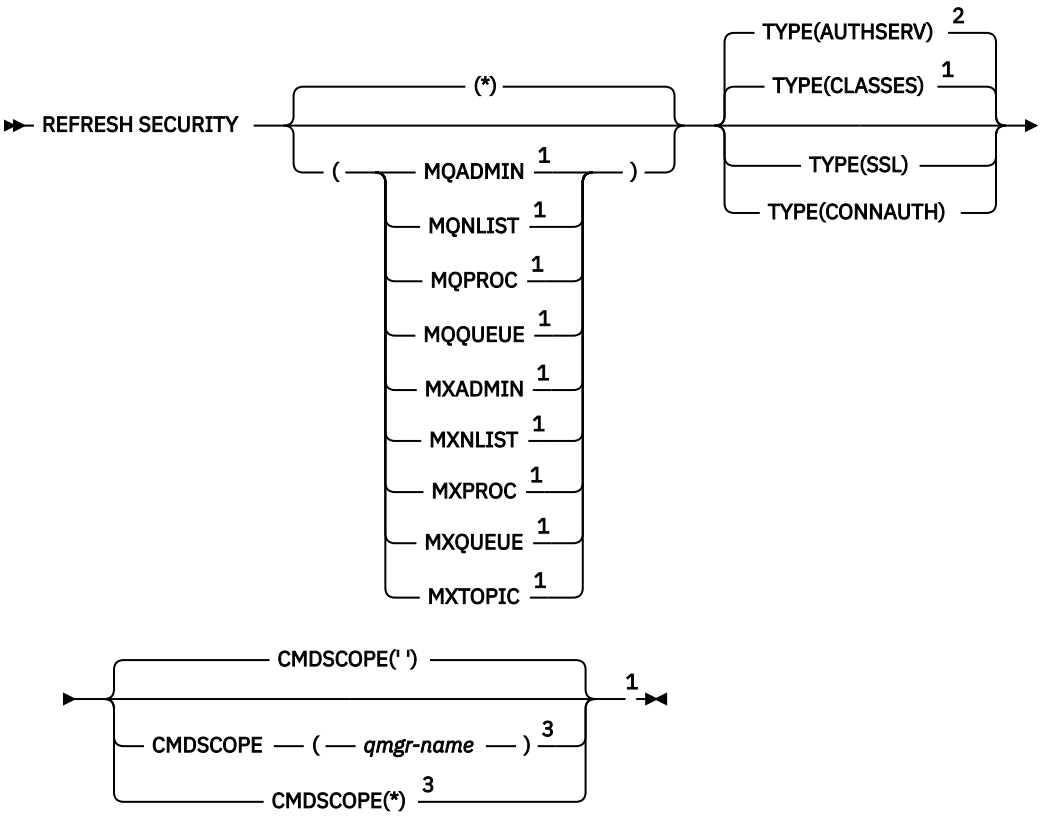
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	See "Using REFRESH SECURITY on z/OS" on page 1239

- Syntax diagram
- z/OS "Using REFRESH SECURITY on z/OS" on page 1239
- "Usage notes for REFRESH SECURITY" on page 1240
- "Parameter descriptions for REFRESH SECURITY" on page 1241

Synonym: REF SEC

REBUILD SECURITY is another synonym for REFRESH SECURITY.

REFRESH SECURITY



Notes:

[1] Valid only on z/OS.
[2] Not valid on z/OS.
[3] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.

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 "Using commands on z/OS" on page 629.

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 SSL channels are stopped and restarted. Sometimes SSL 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 an SSL 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 SSL 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 changed

   If no refresh is needed, the command completes successfully and the channels are unaffected.
3. z/OS (Platforms other than z/OS) The command updates all SSL channels regardless of whether a security refresh is needed.
4. If a refresh is to be performed, the command updates all SSL channels currently running, as follows:
   - Sender, server and cluster-sender channels using SSL are allowed to complete the current batch. In general they then run the SSL handshake again with the refreshed view of the SSL key repository. However, you must manually restart a requester-server channel on which the server definition has no CONNAME parameter.
   - All other channel types using SSL 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 SSL handshake uses the refreshed view of the contents of the SSL 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 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 z/OS for non-z/OS systems.

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.

TYPE
Specifies the type of refresh that is to be performed.

IBM i distributed **AUTHSERV**
The list of authorizations held internally by the authorization services component is refreshed.

This is the default value.

z/OS **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.

You must refresh the configuration before the queue manager recognizes the changes.

Windows IBM i Linux UNIX On IBM i IBM i, UNIX, Linux, and Windows platforms, 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" on page 1235 command.

Related tasks

z/OS Refreshing queue manager security on z/OS

RESET CFSTRUCT

Use the MQSC command RESET CFSTRUCT to modify the status of a specific application structure.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Notes:” on page 1243
- “Parameter descriptions for RESET CFSTRUCT” on page 1243

Synonym: None.

RESET CFSTRUCT

Notes:

1. z/OS Valid only when the queue manager is a member of a queue-sharing group.
2. RESET CFSTRUCT requires 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

RESET 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 Performing local administration tasks using MQSC commands.

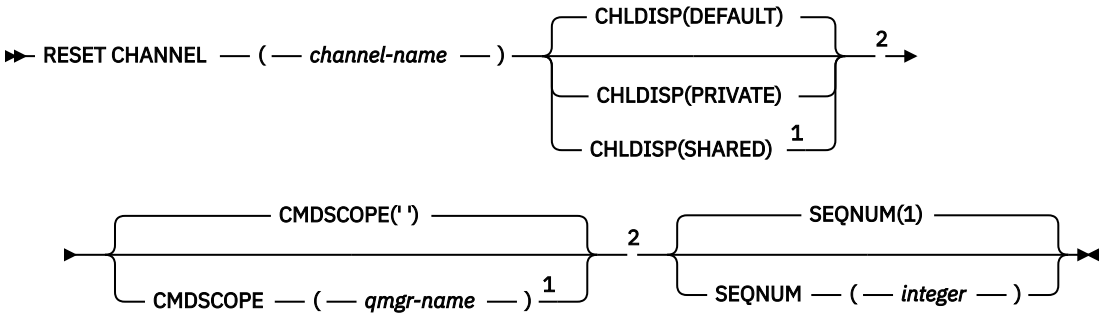
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 1244
- "Parameter descriptions for RESET CHANNEL" on page 1244

Synonym: RESET CHL

RESET CHANNEL



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.
[2] Valid only on z/OS.

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:

Table 77. CHLDISP and CMDSCOPE for RESET CHANNEL

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)
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 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. 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 is executed 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 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 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.

RESET 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 Performing local administration tasks using MQSC commands.

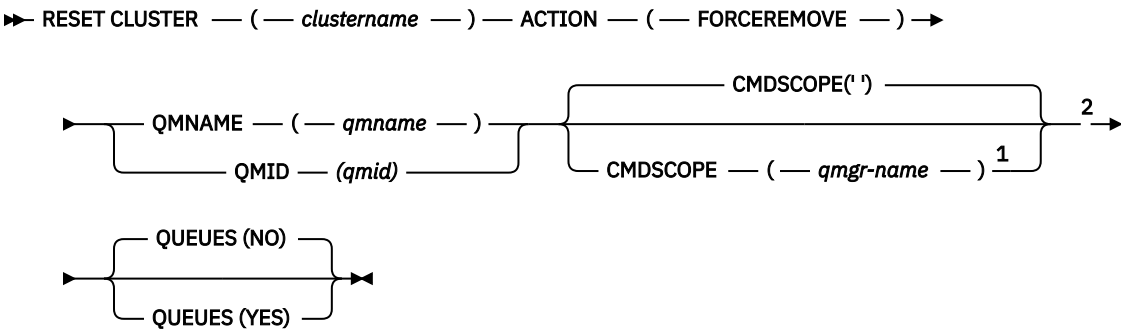
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for RESET CLUSTER" on page 1246
- "Parameter descriptions for RESET CLUSTER" on page 1247

Synonym: None

RESET CLUSTER



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.

Usage notes for RESET CLUSTER

- On UNIX systems, the command is valid only on AIX, HP-UX, Linux, and Solaris.
- 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.

z/OS **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.

' '
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.

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.

z/OS 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

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 Performing local administration tasks using MQSC commands.

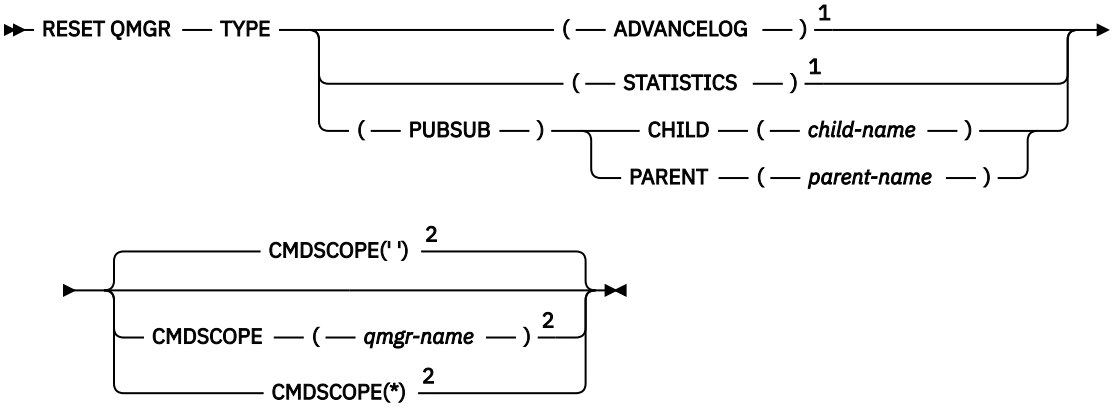
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for RESET QMGR" on page 1248
- "Parameter descriptions for RESET QMGR" on page 1249

Synonym: None

RESET QMGR



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

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.

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.

STATISTICS
Requests that the queue manager ends the current statistics collection period and writes the collected statistics.

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.

z/OS **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. This value 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 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 setting this value is the same as entering the command on every queue manager in the queue-sharing group.

z/OS RESET QSTATS

Use the MQSC command RESET QSTATS to report performance data for a queue and then to reset that data.

Using MQSC commands

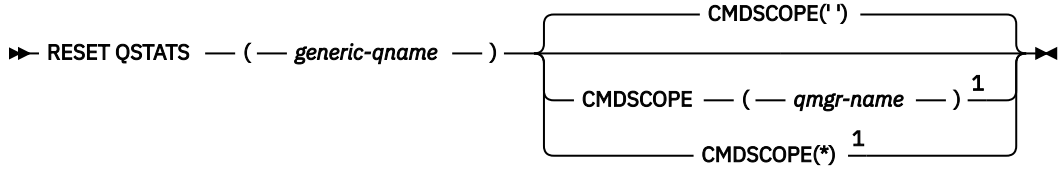
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for RESET QSTATS” on page 1250
- “Parameter descriptions for RESET QSTATS” on page 1251

Synonym: None



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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. z/OS 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

z/OS **QSGDISP**

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.

MSGSIN

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.

MSGSOUT

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.

z/OS **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.

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

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

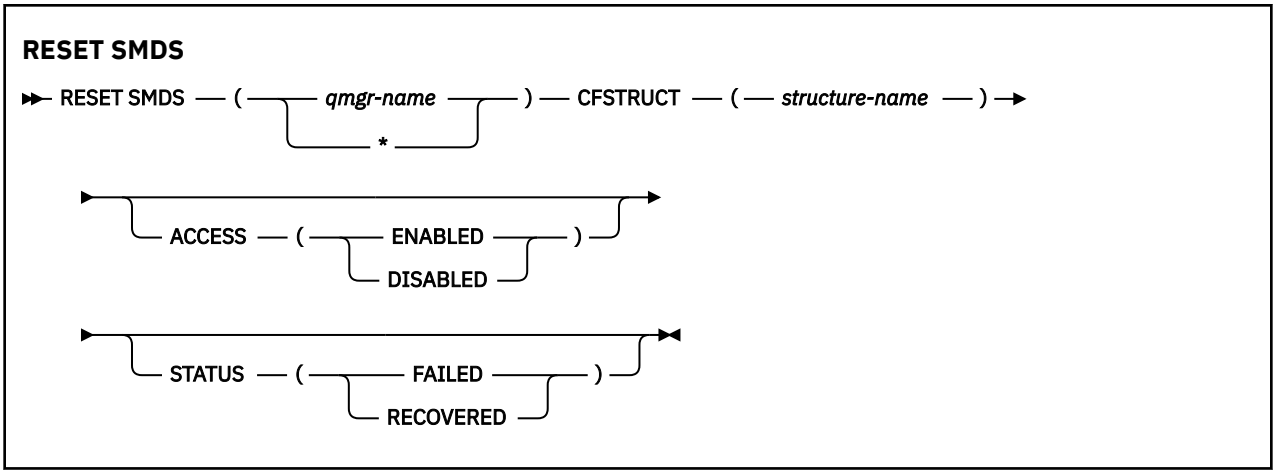
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for RESET SMDS” on page 1253

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

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

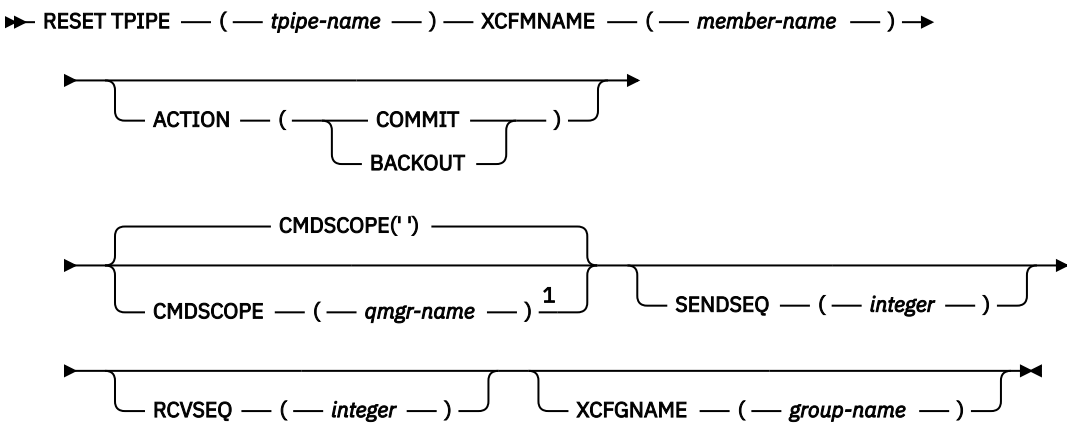
IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 1255
- "Parameter descriptions for RESET TPIPE" on page 1255

Synonym: There is no synonym for this command.

RESET TPIPE



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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 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.

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

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 Performing local administration tasks using MQSC commands.

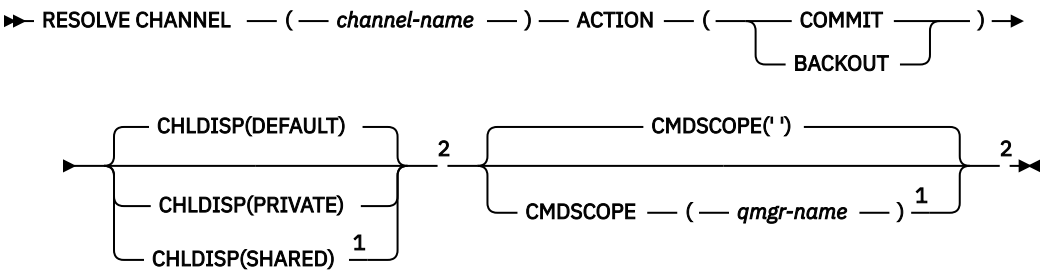
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for RESOLVE CHANNEL" on page 1256
- "Parameter descriptions for RESOLVE CHANNEL" on page 1257

Synonym: RESOLVE CHL (RES CHL on z/OS)

RESOLVE CHANNEL



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.

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 (CLUSSDR) 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:

Table 78. CHLDISP and CMDSCOPE for RESOLVE CHANNEL

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)
PRIVATE	Resolve private channel on the local queue manager	Resolve private channel on the named queue manager

Table 78. CHLDISP and CMDSCOPE for RESOLVE CHANNEL (continued)

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)
SHARED	Resolve a shared channel on all active queue managers. This might automatically generate a command using CMDSCOPE 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 is executed 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 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 only if you are using a queue-sharing group environment and if the command server is enabled.

z/OS RESOLVE INDOUBT

Use the MQSC command RESOLVE INDOUBT to resolve threads left in doubt because IBM MQ or a transaction manager could not resolve them automatically.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

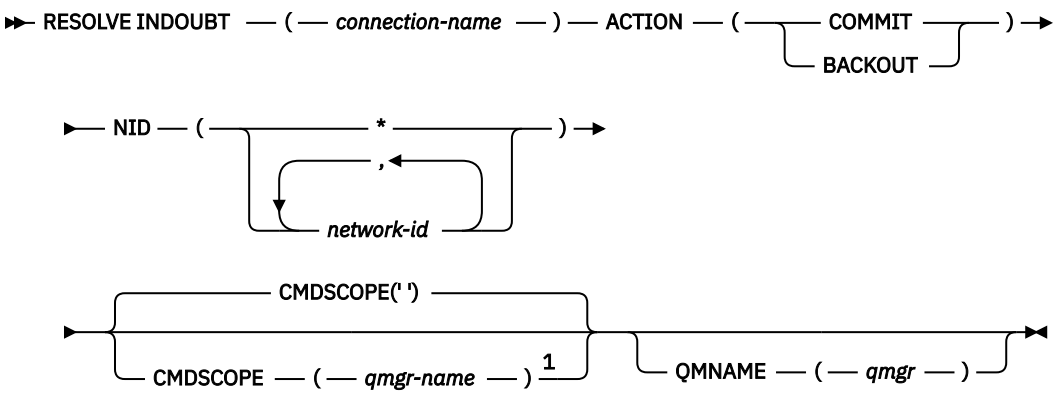
IBM i	UNIX and Linux	Windows	z/OS
			2CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 1259
- "Parameter descriptions for RESOLVE INDOUBT" on page 1259

Synonym: RES IND

RESOLVE INDOUBT



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

Usage notes

This command does not apply to units of recovery associated with batch or TSO applications, unless you are using the RRS adapter.

Parameter descriptions for RESOLVE INDOUBT

(connection-name)

1 through 8 character connection name.

- For a CICS connection it is the CICS applid.
- For an IMS adapter connection, it is the IMS control region job name.
- For an IMS bridge connection, it is the IBM MQ queue manager name.
- For an RRS connection, it is RRSBATCH.
- For a CHIN connection, it is the IBM MQ channel initiator name.

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 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.

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*.

(*)

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

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

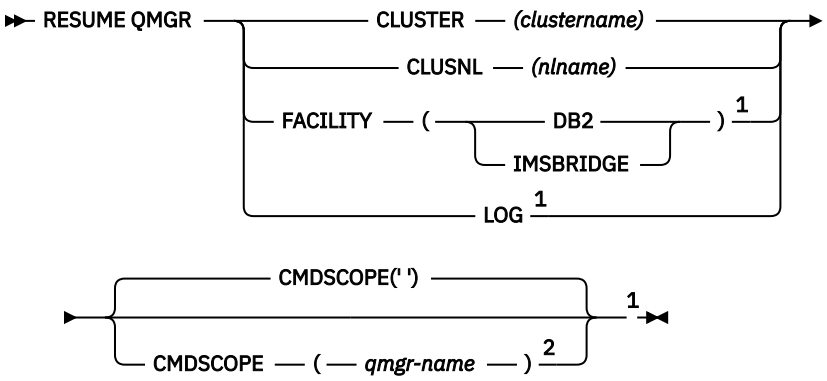
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	See “Using RESUME QMGR on z/OS” on page 1261

- Syntax diagram
- z/OS “Using RESUME QMGR on z/OS” on page 1261
- “Usage notes” on page 1261
- “Parameter descriptions for RESUME QMGR” on page 1261

Synonym: None

RESUME QMGR



Notes:

[1] Valid only on z/OS.

[2] Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.

Using RESUME QMGR on z/OS

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 "Using commands on z/OS" on page 629.

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. On UNIX systems, the command is valid only on AIX, HP-UX, Linux, and Solaris.
2. 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. On z/OS, you cannot issue RESUME QMGR CLUSTER *(clustername)* or RESUME QMGR FACILITY commands from CSQINP2.
4. 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. 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 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.

z/OS RVERIFY SECURITY

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

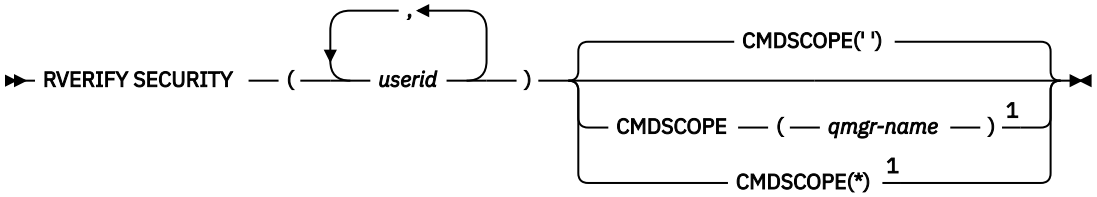
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for RVERIFY SECURITY” on page 1263

Synonym: REV SEC

REVERIFY SECURITY is another synonym for RVERIFY SECURITY

RVERIFY SECURITY



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

Parameter descriptions for RVERIFY SECURITY

(userids...)
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 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 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 SET ARCHIVE

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			12CR

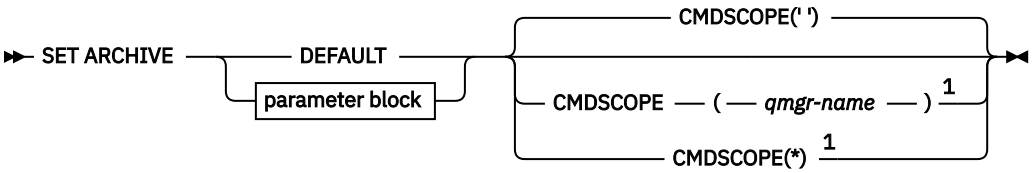
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for SET ARCHIVE" on page 1264
- "Parameter descriptions for SET ARCHIVE" on page 1265

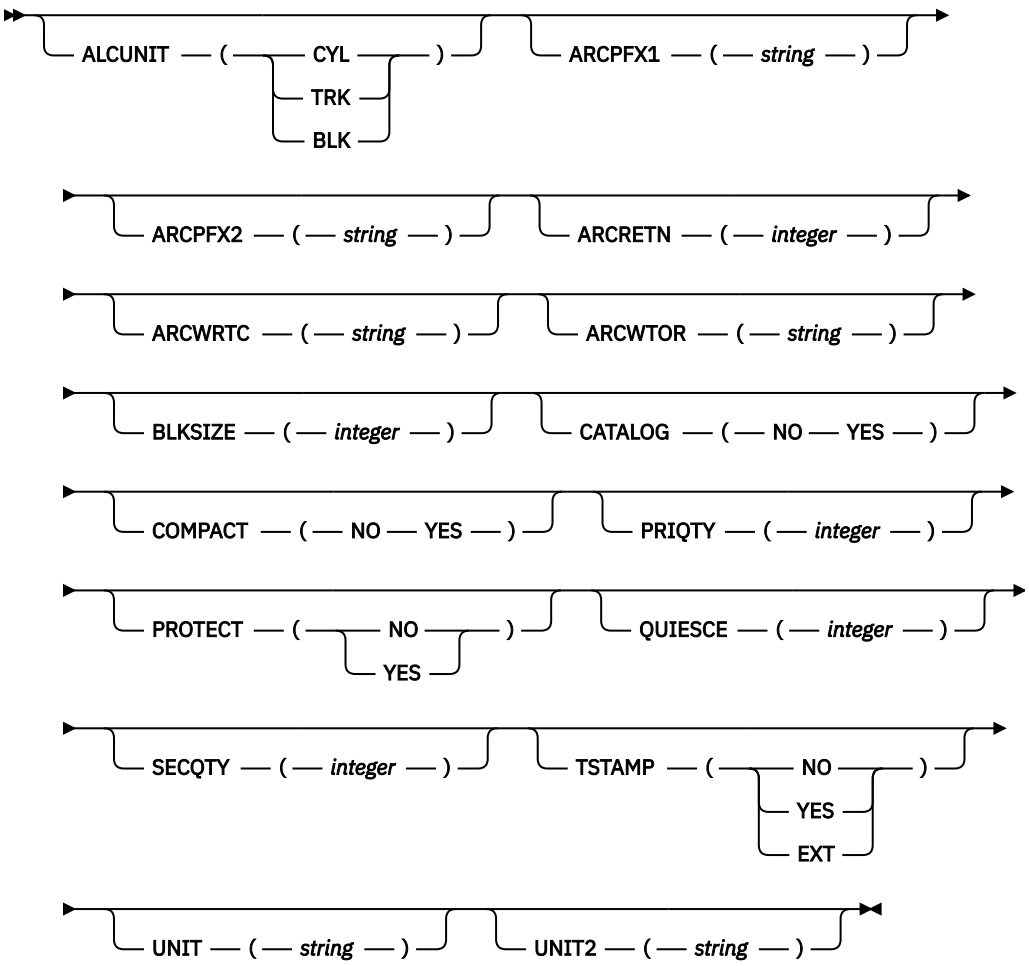
- "Parameter block" on page 1265

Synonym: SET ARC

SET ARCHIVE



Parameter Block



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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 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 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 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 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.

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

z/OS 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.

z/OS 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 the *MVS Routing and Descriptor Codes* manual.

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:

```
arcpfxi.A nnnnnnn
```

Where *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 35 characters.

YES
Names include a time stamp. (The synonym is **Y**.) The archive log data sets are named:

```
arcpfxi.cyyddd.T hhmmsst.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 ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 19 characters.

EXT
Names include a time stamp. The archive log data sets are named:

```
arcpfxi.D yyyyddd.T hhmmsst.A nnnnnnn
```

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.

SET AUTHREC

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

Using MQSC commands

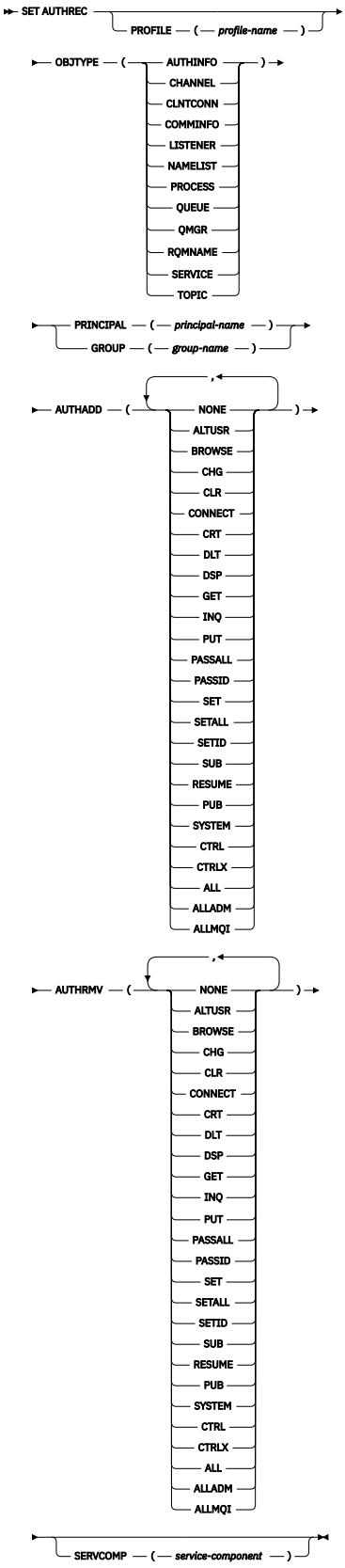
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Parameter descriptions” on page 1270
- Usage notes for SET AUTHREC

See “setmqaut” on page 123 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 UNIX, Linux, and Windows systems 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.

SET CHLAUTH

Use the MQSC command SET CHLAUTH to create or modify a channel authentication record.

Using MQSC commands

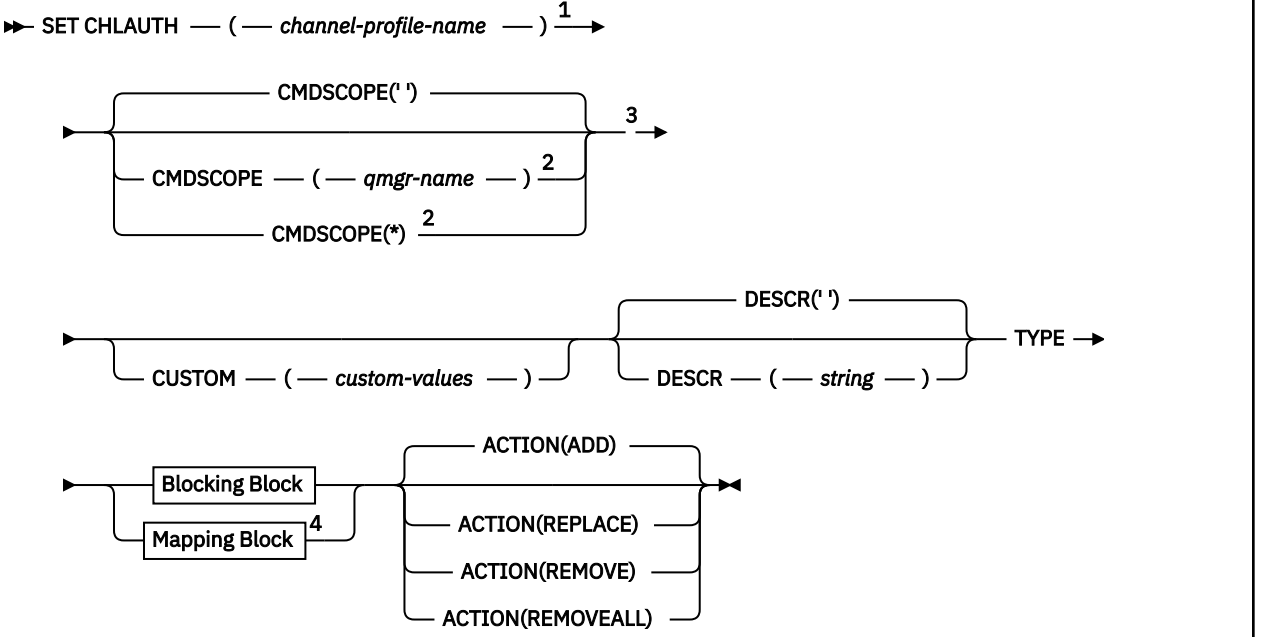
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	2CR

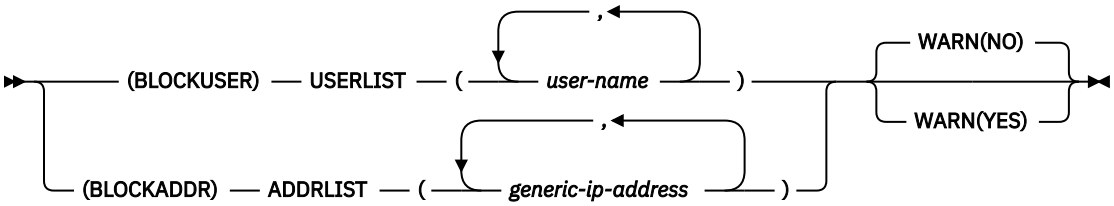
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- Usage notes
- Parameters

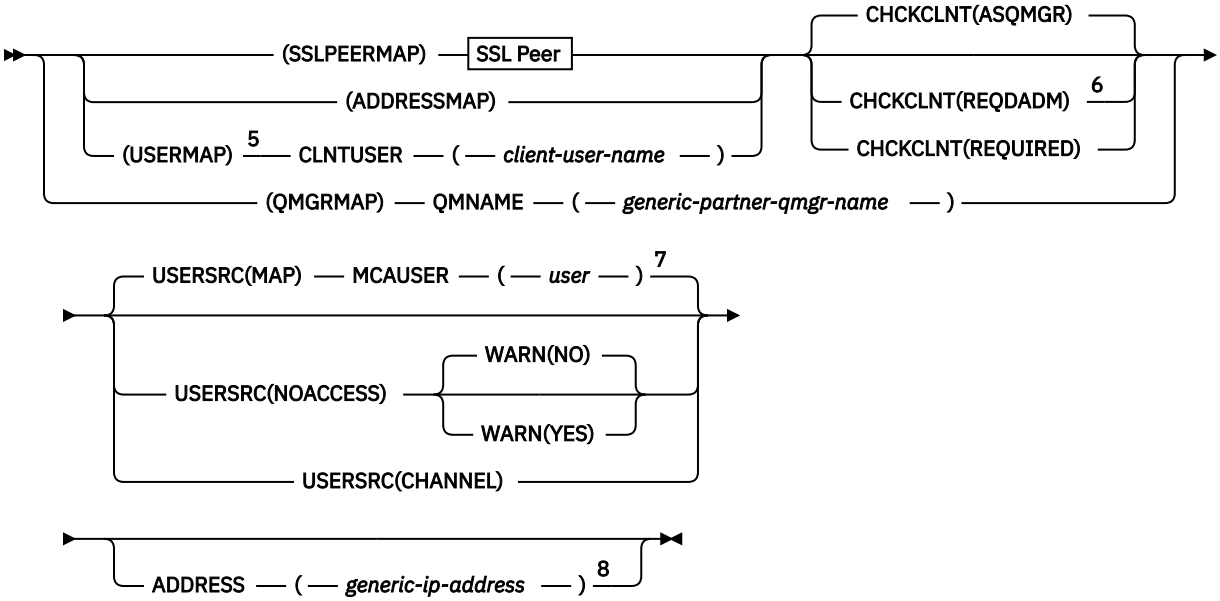
SET CHLAUTH



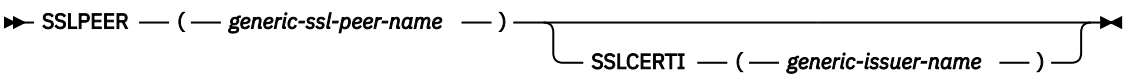
Blocking Block



Mapping Block



SSL Peer



Notes:

[1] The channel profile name must be '*' when TYPE is BLOCKADDR.
[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] Select the appropriate value for TYPE, depending upon the option that you select from the two types of block.
[5] USERMAP rules apply only to server connection channels.
[6] Not valid on z/OS.
[7] If you allow USERSRC to default to MAP, you must set a value for the parameter MCAUSER.
[8] Mandatory when TYPE is ADDRESSMAP.

Usage notes

The following table shows which parameters are valid for each value of **ACTION**:

	Action		
Parameter	**ADD or REPLACE**	**REMOVE**	**REMOVEALL**
CHLAUTH	✓	✓	✓
TYPE	✓	✓	✓
z/OS z/OS CMDSCOPE	✓	✓	✓
ACTION	✓	✓	✓
ADDRESS	✓	✓	
ADDRLIST	✓	✓	
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. The BLOCKUSER parameter must be accompanied by a USERLIST.

BLOCKADDR
This channel authentication record prevents connections from a specified IP address or addresses. The BLOCKADDR parameter must be accompanied by an ADDRLIST. BLOCKADDR operates at the listener before the channel name is known.

SSLPEERMAP
This channel authentication record maps SSL or TLS Distinguished Names (DNs) to MCAUSER values. The SSLPEERMAP parameter must be accompanied by an SSLPEER.

ADDRESSMAP
This channel authentication record maps IP addresses to MCAUSER values. The ADDRESSMAP parameter must be accompanied by an ADDRESS. ADDRESSMAP operates at the channel.

USERMAP
This channel authentication record maps asserted user IDs to MCAUSER values. The USERMAP parameter must be accompanied by a CLNTUSER.

QMGRMAP
This channel authentication record maps remote queue manager names to MCAUSER values. The QMGRMAP parameter must be accompanied by a QMNAME.

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 types SSLPEERMAP, ADDRESSMAP, USERMAP and QMGRMAP, if the specified configuration exists, the command fails.

For types BLOCKUSER and BLOCKADDR, the configuration is added to the list.

REPLACE
Replace the current configuration of a channel authentication record.

For types 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 types 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. If the configuration does not exist the command fails. 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

Attention: Host names can be specified in this parameter, only on queue managers that have IBM MQ 8.0 new functions enabled with OPMODE.

The filter to be used to compare with the IP address or hostname 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 hostnames 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 SSL 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” on page 1280 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). See “Generic IP addresses” on page 1280 for more information about filtering IP addresses.

If the address is generic then it must be in quotes.

CHCKCLNT
Specifies whether the connection that matches this rule and is being allowed in with **USERSRC(CHANNEL)** or **USERSRC(MAP)**, must also specify a valid user ID and password. The password cannot contain single quotation marks (').

Attention: This parameter is valid only on queue managers that have IBM MQ 8.0 new functions enabled with OPMODE.

REQDADM
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.

z/OS This option is not valid on z/OS platforms.

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.

ASQMGR
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 REQUIRED or REQDADM z/OS (on platforms other than z/OS), and you have not set the CONNAUTH field on the queue manager, or the value of CHCKCLNT is NONE, the connection fails. You receive message AMQ9793 z/OS on platforms other than z/OS, and message CSQX793E on z/OS.

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 a valid password is supplied:

```
SET CHLAUTH('*.SVRCONN') +
    TYPE(ADDRESSMAP) ADDRESS('192.0.2.*') +
    USERSRC(CHANNEL) CHCKCLNT(REQUIRED)
```

- This rule ensures that SSL authentication must succeed before processing client authentication according to the policy set at 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 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 MQCONNX call using MQCSP.

The maximum length of the string is MQ_CLIENT_USER_ID_LENGTH.

z/OS **CMDSCOPE**
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.

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 SSL or 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

Attention: This parameter is valid only on queue managers that have IBM MQ 8.0 new functions enabled with OPMODE.

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. See IBM MQ rules for SSLPEER values for details.

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:

- On Windows, all members of the mqm group, the Administrators group and SYSTEM.
- On UNIX 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.
- 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 **ChlauthIssueWarn**=*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
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 are all valid values:
  - 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 are all valid values:
  - 2001:DB8:0:0:0:0:0:*
  - 2001:DB8:0:0:0:*
  - 2001:DB8:0:0:0:*: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:*:0:0-8

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

z/OS SET LOG

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

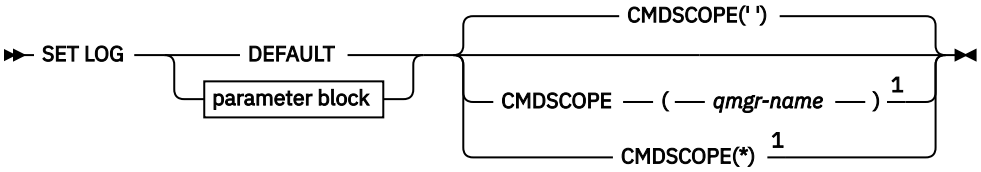
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

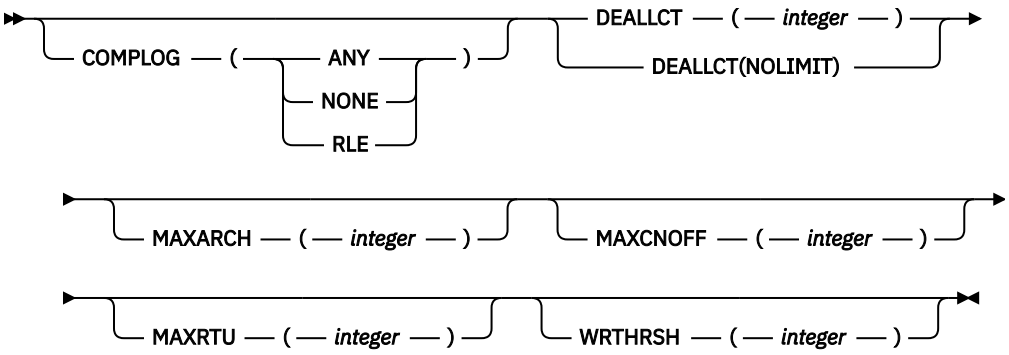
- Syntax diagram
- "Usage notes for SET LOG" on page 1282
- "Parameter descriptions for SET LOG" on page 1283
- "Parameter block" on page 1283

Synonym: SET LOG

SET LOG



Parameter Block



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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 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 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 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.

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

z/OS 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).

z/OS 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.

SET POLICY

Use the MQSC command SET POLICY to set a security policy.

Using MQSC commands

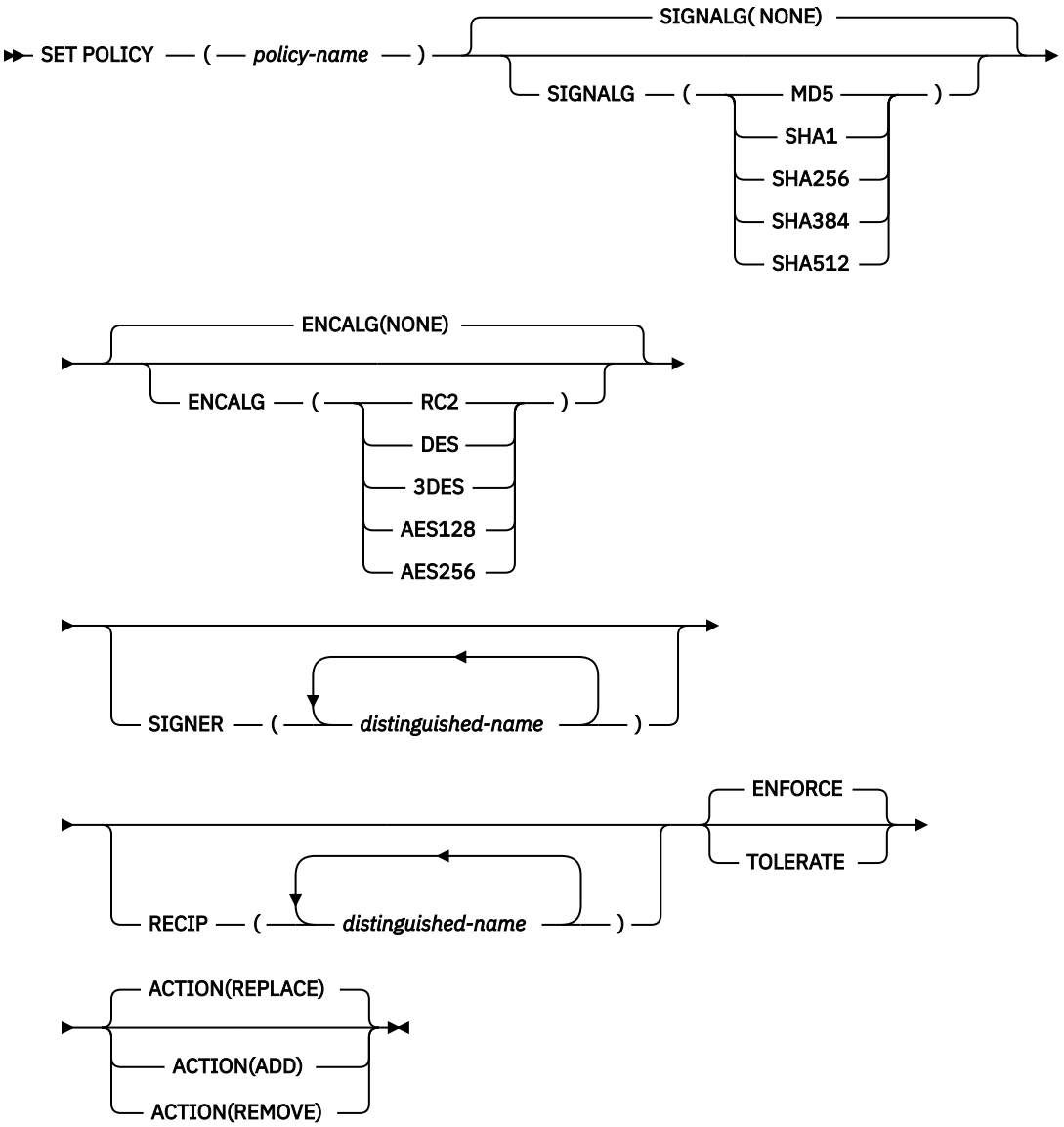
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Parameter descriptions for SET POLICY” on page 1285

Important: You must have an IBM MQ 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
- MD5
- SHA1
- 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.

ENFORCEis 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.

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" on page 1113
Use the MQSC command DISPLAY POLICY to display a security policy.

"setmqspl" on page 144
Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy.

"dspmqspl" on page 73
Use the **dspmqspl** command to display a list of all policies and details of a named policy.

z/OS SET SYSTEM

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

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

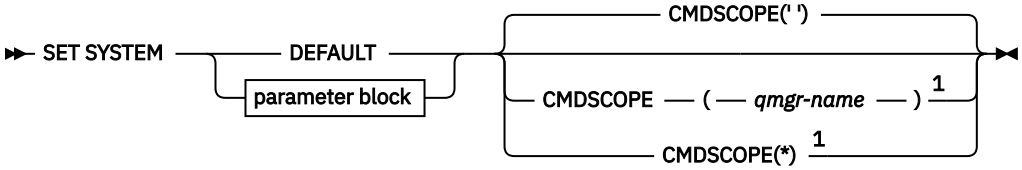
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

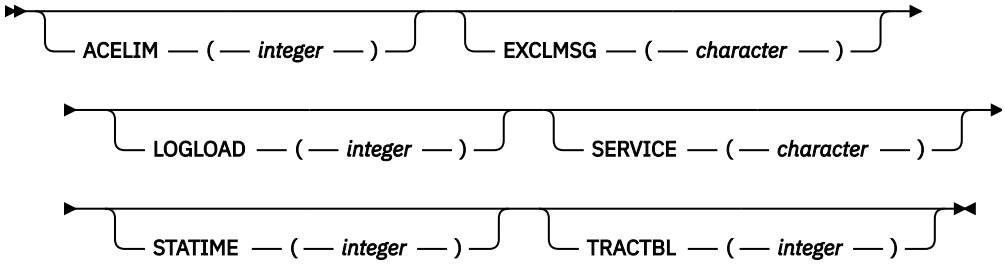
- Syntax diagram
- "Usage notes for SET SYSTEM" on page 1288
- "Parameter descriptions for SET SYSTEM" on page 1288
- "Parameter block" on page 1289

Synonym: None

SET SYSTEM



Parameter Block



Notes:

1 Valid only when the queue manager is a member of a queue-sharing group.

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 and TRACTBL.

Changes to STATIME take effect when the current interval expires, unless the new interval is less than the unexpired portion of the current interval, in which case statistics are 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 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 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 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.

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

z/OS 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:

ACELIM

Specifies the maximum size of the ACE storage pool in 1 KB blocks. The number must be in the range 0-999999. 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; that is the first two elements in QSRSPHBT added together.

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 start up 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
Specifies the interval, in minutes, between consecutive gatherings of statistics.

Specify a number in the range zero through 1440.

If you specify a value of zero, both statistics data and accounting data is collected at the SMF data collection broadcast.

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

Use the MQSC command START CHANNEL to start a channel.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

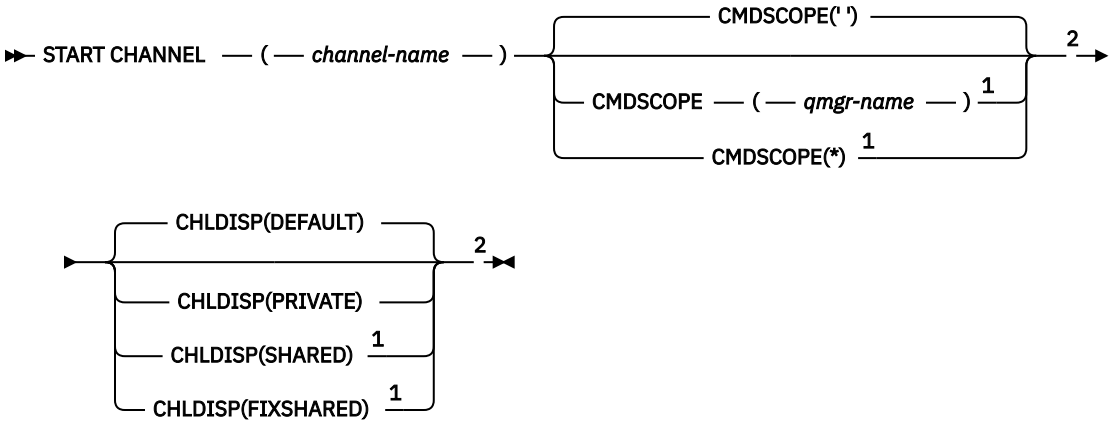
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes” on page 1291
- “Parameter descriptions for START CHANNEL” on page 1291

Synonym: STA CHL

START CHANNEL



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.

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.

z/OS **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:

Table 79. CHLDISP and CMDSCOPE for START CHANNEL

CHLDISP	**CMDSCOPE( ) or CMDSCOPE (local-qmgr)**	**CMDSCOPE (qmgr-name)**	**CMDSCOPE(*)**
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
SHARED	For a shared SDR, RQSTR, and SVR channel, start as a shared channel on the most suitable queue manager in the group. For a shared RCVR and SVRCONN channel, start the channel as a shared channel on all active queue managers. For a shared CLUSSDR or CLUSRCVR channel, this option is not permitted. 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. 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	Not permitted

Table 79. CHLDISP and CMDSCOPE for START CHANNEL (continued)

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)	CMDSCOPE(*)
FIXSHARED	For a shared SDR, RQSTR, and SVR channel, with a nonblank CONNAME, start as a shared channel on the local queue manager. For all other types, this option is not permitted.	For a shared SDR, RQSTR, and SVR with a nonblank CONNAME, start as a shared channel on the named queue manager. For all other types, this option is not permitted.	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.

z/OS 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.

If CHLDISP is set to SHARED, CMDSCOPE must be blank or the local queue manager.

' '

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 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.

This option is not permitted if CHLDISP is FIXSHARED.

START CHANNEL (MQTT)

Use the MQSC command START CHANNEL to start an IBM MQ Telemetry channel.

Using MQSC commands

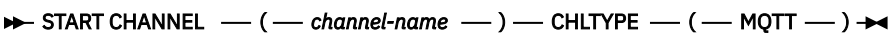
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	AIX and Linux	Windows	z/OS
	✓	✓	

The START CHANNEL (MQTT) command is only valid for IBM MQ Telemetry channels. Supported platforms for IBM MQ Telemetry are AIX, Linux, Windows.

Synonym: STA CHL

START CHANNEL



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

Use the MQSC command START CHINIT to start a channel initiator.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

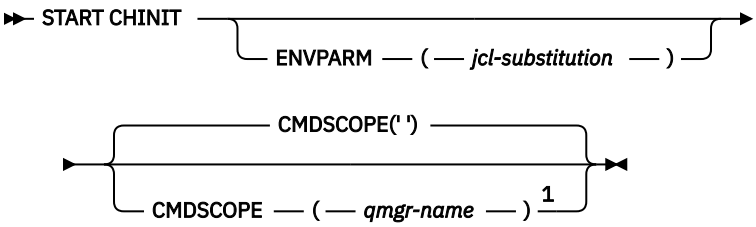
For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 1294
- "Parameter descriptions for START CHINIT" on page 1295

Synonym: STA CHI

Syntax diagram

START CHINIT



Notes:

[1] Valid only on z/OS when the queue manager is a member of a queue-sharing group.

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 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.

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 tasks

Command resource security checking for alias queues and remote queues

z/OS START CMDSERV

Use the MQSC command START CMDSERV to initialize the command server.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			12C

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for START CMDSERV" on page 1296

Synonym: STA CS

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

Use the MQSC command START LISTENER to start a channel listener.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

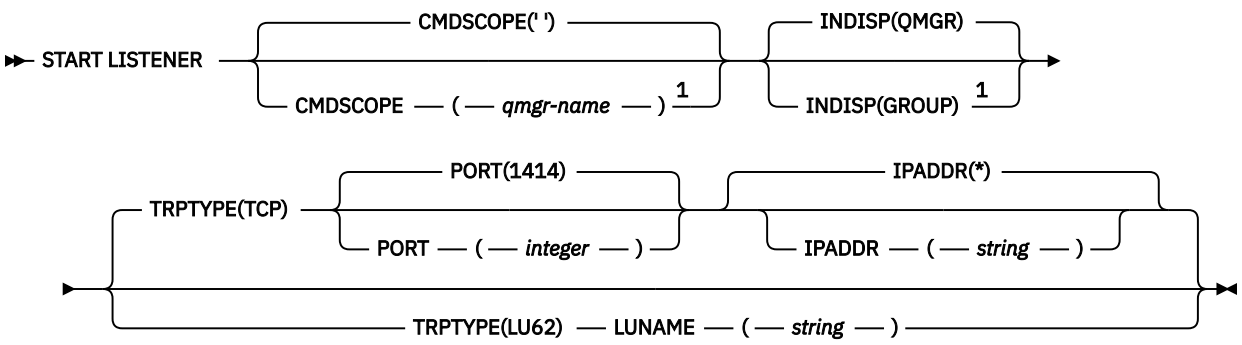
- z/OS Syntax diagram for IBM MQ for z/OS
- Syntax diagram for IBM MQ on other platforms
- “Usage notes” on page 1297
- “Parameter descriptions for START LISTENER” on page 1297

Synonym: STA LSTR

IBM MQ for z/OS

z/OS

START LISTENER

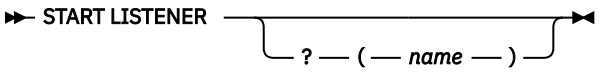


Notes:

[1] Valid only on z/OS when the queue manager is a member of a queue-sharing group.

IBM MQ on other platforms

START LISTENER



Usage notes

1. z/OS 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 IBM i, UNIX systems, and Windows, 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 z/OS (on platforms other than z/OS), the SYSTEM.DEFAULT.LISTENER.TCP is started.

z/OS This parameter is not valid on z/OS.

z/OS **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.

' '

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.

z/OS **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.

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 **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.

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.

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.

z/OS START QMGR

Use the MQSC command START QMGR to initialize the queue manager.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

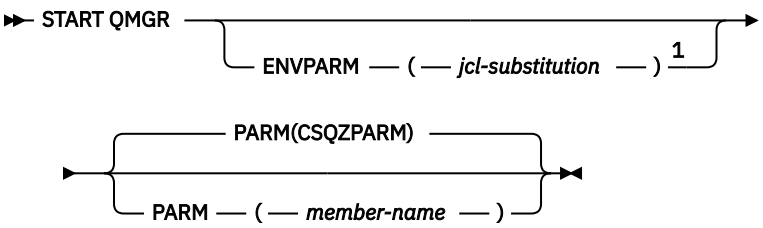
IBM i	UNIX and Linux	Windows	z/OS
			C

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes" on page 1299
- "Parameter descriptions for START QMGR" on page 1299

Synonym: STA QMGR

START QMGR



Notes:

1 MSTR is accepted as a synonym for ENVPARM

Usage notes

When the command has been completed, the queue manager is active and available to CICS, IMS, batch, and TSO applications.

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.

START SERVICE

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

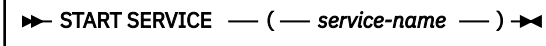
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- “Parameter descriptions for START SERVICE” on page 1300

Synonym:

STA SERVICE



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.

Related concepts
Working with services

z/OS START SMDSCONN

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

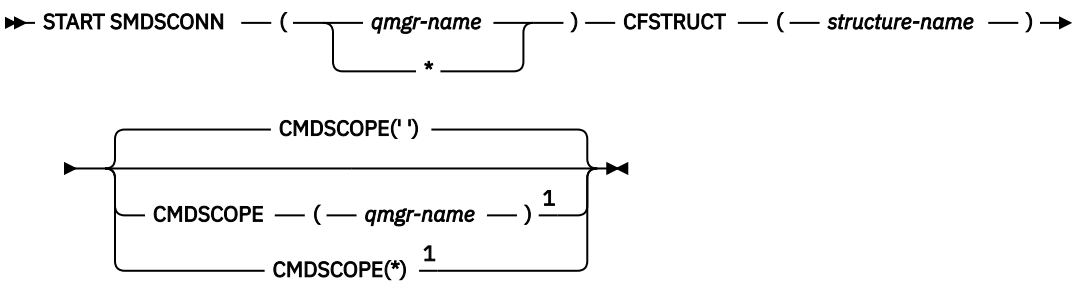
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

- Syntax diagram
- “Parameter descriptions for START SMDSCONN” on page 1301

Synonym:

START SMDSCONN



Notes:

[1] 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 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 START TRACE

Use the MQSC command START TRACE to start traces.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

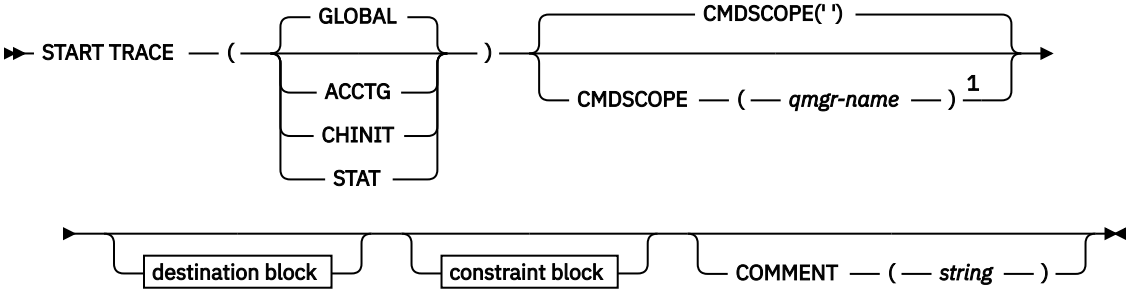
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

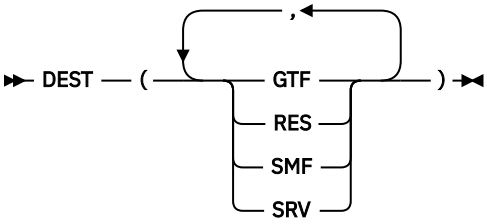
- Syntax diagram
- "Usage notes" on page 1303
- "Parameter descriptions for START TRACE" on page 1303
- "Destination block" on page 1303
- "Constraint block" on page 1304

Synonym: STA TRACE

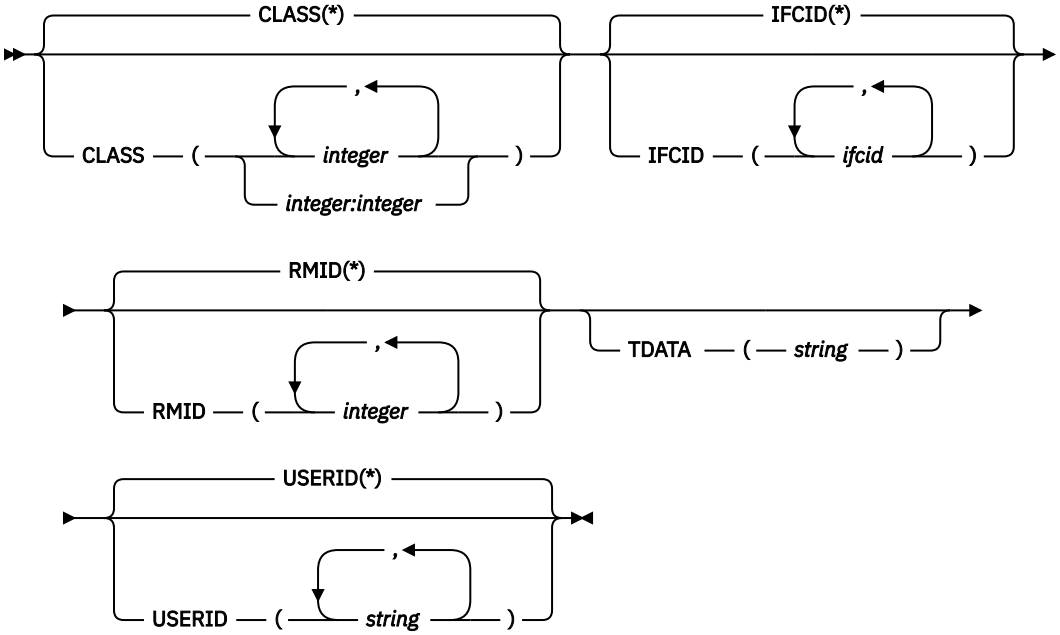
START TRACE



Destination block



Constraint block



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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 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 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.

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 z/OS 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:

Table 80. Destinations allowed for each trace type

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:

Table 81. Constraints allowed for each trace type

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:

(*)
Starts a trace for all classes of data except "CLASS 04 Channel initiator statistics" and "CLASS 04 Channel accounting data".

(*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.

Table 82. Descriptions of trace events and classes.

Table showing the different trace events produced for the various trace classes.

Class	Description
	Global trace
01	Reserved for IBM service
02	User parameter error detected in a control block
03	User parameter error detected on entry to MQI
	User parameter error detected on exit from MQI
	User parameter error detected in a control block
04	Reserved for IBM service
	Statistics trace
01	Subsystem statistics
	Queue manager statistics
02	Queue manager storage summary statistics
03	Queue manager storage detail summary
04	Channel initiator statistics
	Accounting trace
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
	CHINIT trace
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.

Table 83. Resource Manager identifiers that are allowed

RMID	Resource manager
1	Initialization procedures
2	Agent services management

Table 83. Resource Manager identifiers that are allowed (continued)	
RMID	**Resource manager**
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.

STOP CHANNEL

Use the MQSC command STOP CHANNEL to stop a channel.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

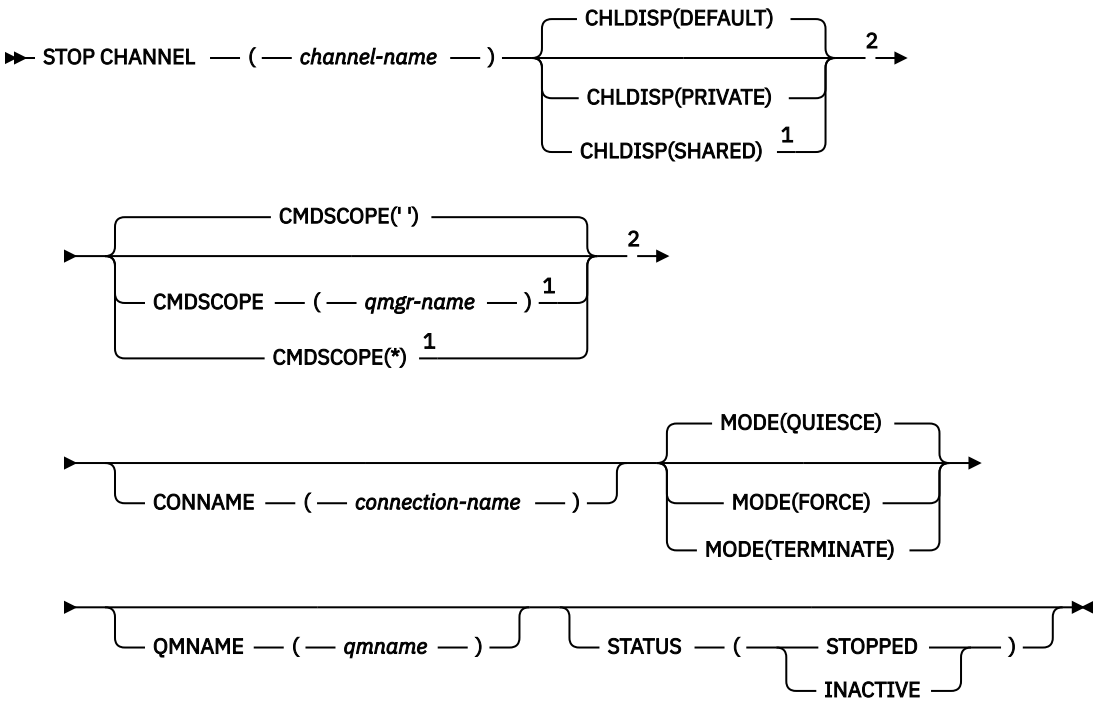
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for STOP CHANNEL” on page 1307
- “Parameter descriptions for STOP CHANNEL” on page 1308

Synonym: STOP CHL

STOP CHANNEL



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.

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. z/OS 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.

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.
- 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:

Table 84. CHLDISP and CMDSCOPE for STOP CHANNEL

CHLDISP	CMDSCOPE( ) or CMDSCOPE (local-qmgr)	CMDSCOPE (qmgr-name)	CMDSCOPE(*)
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 84. CHLDISP and CMDSCOPE for STOP CHANNEL (continued)			
CHLDISP	**CMDSCOPE( ) or CMDSCOPE (local-qmgr)**	**CMDSCOPE (qmgr-name)**	**CMDSCOPE(*)**
SHARED	For RCVR and SVRCONN channels, stop as shared channel on all active queue managers. 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. This might automatically generate a command using CMDSCOPE 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	Not permitted

z/OS **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.

If CHLDISP is set to SHARED, CMDSCOPE must be blank or the local queue manager.

' '

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 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.

CONNAME *(connection-name)*

Connection name. Only channels matching the specified connection name are stopped.

When issuing the STOP CHANNEL command using a **CONNAME** parameter, ensure that the value specified in the **CONNAME** parameter is exactly as shown in "DISPLAY CHSTATUS" on page 1042.

MODE

Specifies whether the current batch is allowed to finish in a controlled manner. This parameter is optional.

QUIESCE

This is the default.

Allows the current batch to finish processing on distributed platforms.

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_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 this is synonymous with FORCE.

On other platforms, this parameter terminates transmission of any current batch. This allows the command to actually terminate the channel thread or process.

For server-connection channels, breaks the current connection, returning MQRC_CONNECTION_BROKEN.

z/OS On IBM MQ for 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 details about channels in STOPPED state, especially SVRCONN channels on IBM MQ for 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.

STOP CHANNEL (MQTT)

Use the MQSC command STOP CHANNEL to stop an IBM MQ Telemetry channel.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

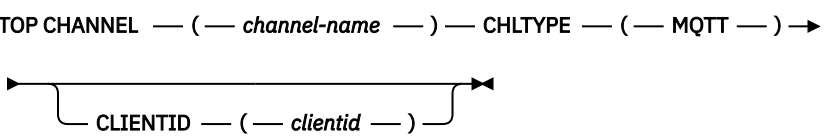
IBM i	AIX and Linux	Windows	z/OS
	✓	✓	

Note: For the telemetry server, AIX is the only supported UNIX platform.

The STOP CHANNEL (MQTT) command is only valid for IBM MQ Telemetry channels.

Synonym: STOP CHL

STOP CHANNEL



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 IBM 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.

z/OS STOP CHINIT

Use the MQSC command STOP CHINIT to stop a channel initiator. The command server must be running.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

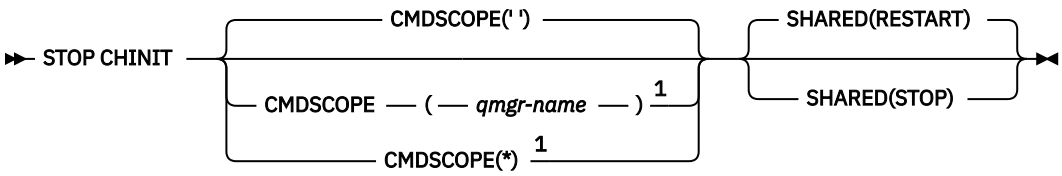
IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- "Usage notes for STOP CHINIT" on page 1312
- "Parameter descriptions for STOP CHINIT" on page 1313

Synonym: STOP CHI

STOP CHINIT



Notes:

[1] 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" on page 1307 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 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.

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.)

z/OS STOP CMDSERV

Use the MQSC command STOP CMDSERV to stop the command server.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			12C

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Usage notes for STOP CMDSERV” on page 1314

Synonym: STOP CS

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.

STOP CONN

Use the MQSC command STOP CONN to break a connection between an application and the queue manager.

Using MQSC commands

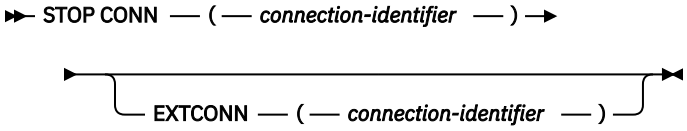
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- "Usage notes" on page 1314
- "Parameter descriptions for STOP CONN" on page 1314

Synonym: STOP CONN

STOP CONN



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" on page 1082
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

Use the MQSC command STOP LISTENER to stop a channel listener.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

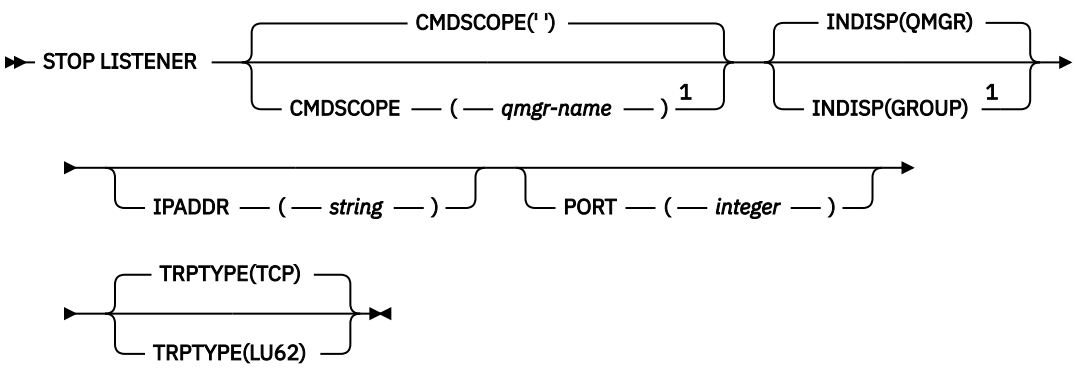
- z/OS Syntax diagram for IBM MQ for z/OS
- Syntax diagram for IBM MQ on other platforms
- z/OS "Usage notes" on page 1316
- "Parameter descriptions for STOP LISTENER" on page 1316

Synonym: STOP LSTR

IBM MQ for z/OS

z/OS

STOP LISTENER

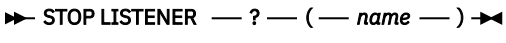


Notes:

1 Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.

IBM MQ on other platforms

STOP LISTENER



Usage notes

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 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.

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.

The listener stops in quiesce mode (it disregards any further requests).

z/OS STOP QMGR

Use the MQSC command STOP QMGR to stop the queue manager.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

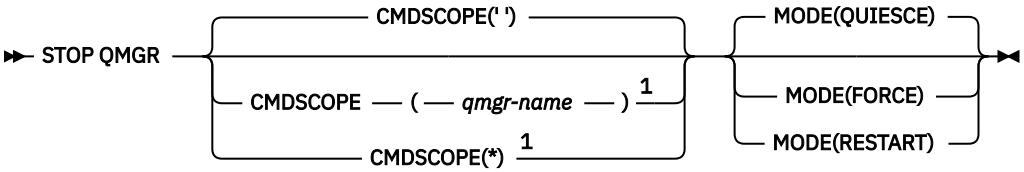
IBM i	UNIX and Linux	Windows	z/OS
			CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- “Parameter descriptions for STOP QMGR” on page 1318

Synonym: There is no synonym for this command.

STOP QMGR



Notes:

[1] 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 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.

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

Use the MQSC command STOP SERVICE to stop a service.

Using MQSC commands

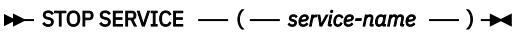
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	

- Syntax diagram
- "Usage notes" on page 1319
- "Parameter descriptions for STOP SERVICE" on page 1319

Synonym:

STOP SERVICE



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.

Related concepts
Working with services
Related reference
"ALTER SERVICE" on page 787
Use the MQSC command ALTER SERVICE to alter the parameters of an existing IBM MQ service definition.
"START SERVICE" on page 1300

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

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

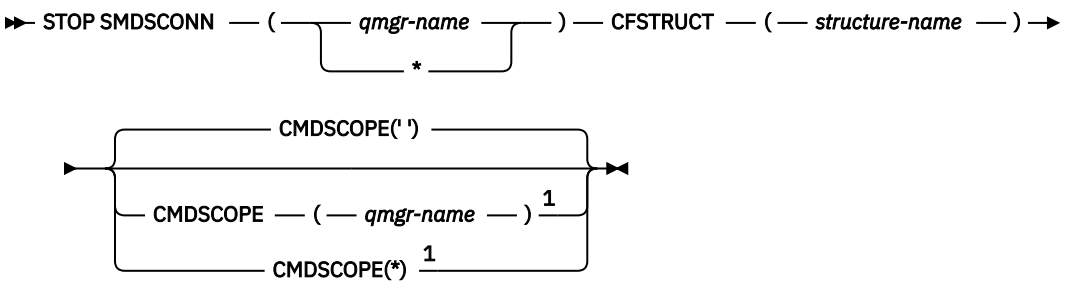
For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

IBM i	UNIX and Linux	Windows	z/OS
			2CR

- "Syntax diagram for STOP SMDSCONN" on page 1320
- "Parameter descriptions for STOP SMDSCONN" on page 1320

Syntax diagram for STOP SMDSCONN

Synonym:



Notes:

[1] Valid only when the queue manager is a member of a queue-sharing group.

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 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 STOP TRACE

Use the MQSC command STOP TRACE to stop tracing.

Using MQSC commands

For information on how you use MQSC commands, see Performing local administration tasks using MQSC commands.

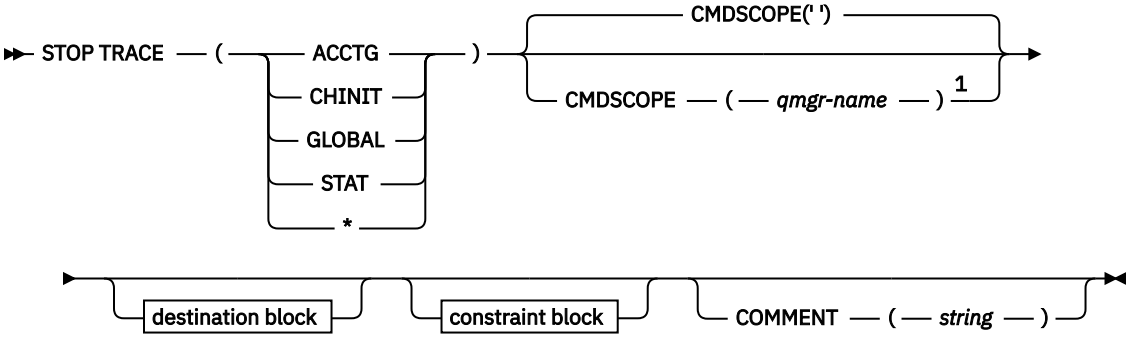
IBM i	UNIX and Linux	Windows	z/OS
			12CR

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

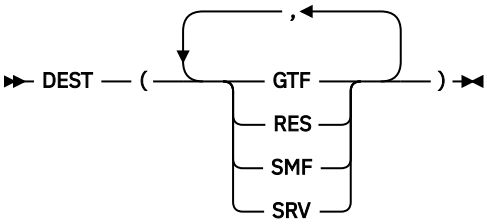
- Syntax diagram
- "Parameter descriptions for STOP TRACE" on page 1322
- "Destination block" on page 1323
- "Constraint block" on page 1323

Synonym: There is no synonym for this command.

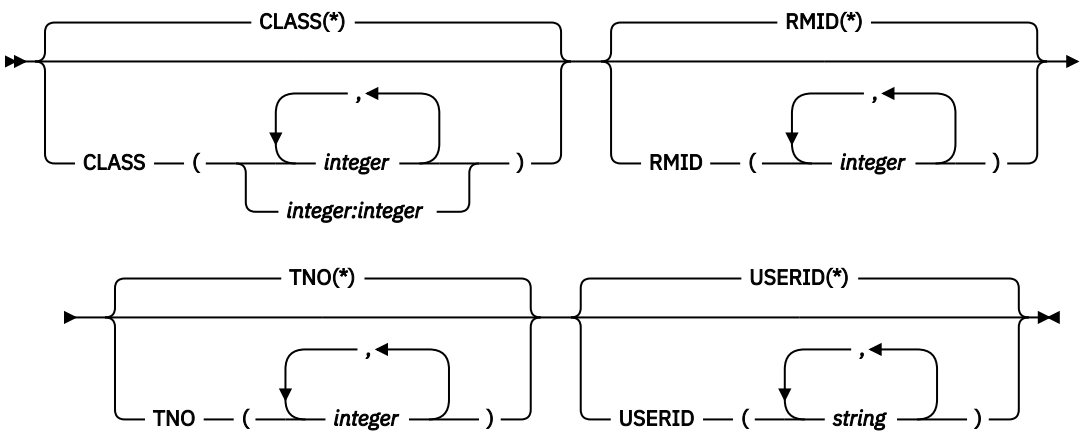
STOP TRACE



Destination block



Constraint block



Notes:

[1] Valid only on z/OS when the queue manager is a member of a queue-sharing group.

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 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 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.

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.

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 be 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

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 Performing local administration tasks 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.

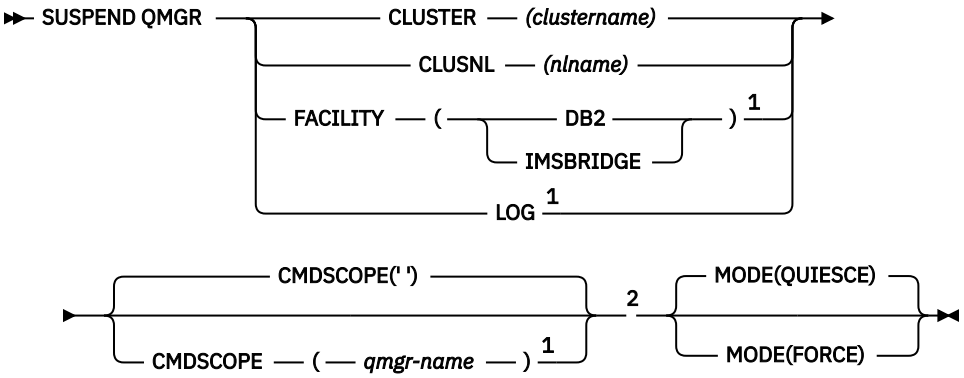
IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	See "Using SUSPEND QMGR on z/OS" on page 1324

For an explanation of the symbols in the z/OS column, see Using commands on z/OS.

- Syntax diagram
- z/OS "Using SUSPEND QMGR on z/OS" on page 1324
- z/OS "Usage notes" on page 1325
- "Parameter descriptions for SUSPEND QMGR" on page 1325

Synonym: None

SUSPEND QMGR



Notes:

1 Valid only on z/OS.
2 Valid only on IBM MQ for z/OS when the queue manager is a member of a queue-sharing group.

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 "Using commands on z/OS" on page 629.

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 is executed 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 See Controlling the IMS bridge for more information about controlling message delivery to shared and non-shared queues.

CLUSTER ***(clustername)***
The name of the cluster for which availability is to be suspended.

CLUSNL ***(nlname)***
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” on page 1260 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" on page 1260 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 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.

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" on page 1260
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 reference

Programmable Command Formats (PCFs) define command and reply messages that can be exchanged between a program and any queue manager (that supports PCFs) in a network. PCFs simplify 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 1327.

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 commands and responses" on page 1836.

For an example PCF, see "PCF example" on page 1862.

Related concepts

"IBM MQ Control commands" on page 8
Find out how to use the IBM MQ control commands.

"MQSC reference" on page 627
Use MQSC commands to manage queue manager objects, including the queue manager itself, queues, process definitions, channels, client connection channels, listeners, services, namelists, clusters, and authentication information objects.

Related reference

"IBM MQ for IBM i CL commands" on page 187
A list of IBM MQ for IBM i CL commands grouped according to command type:

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.

z/OS "Backup CF Structure" on page 1341
"Change, Copy, and Create Authentication Information Object" on page 1342
z/OS "Change, Copy, and Create CF Structure" on page 1351
"Change, Copy, and Create Channel" on page 1356
"Change, Copy, and Create Channel (MQTT)" on page 1389
"Change, Copy, and Create Channel Listener" on page 1394
"Change, Copy, and Create Namelist" on page 1400
"Change, Copy, and Create Process" on page 1403

"Change, Copy, and Create Queue" on page 1407
"Change Queue Manager" on page 1424
"Change Security" on page 1451
z/OS "Change SMDS" on page 1452
"Change, Copy, and Create Service" on page 1453
z/OS "Change, Copy, and Create Storage Class" on page 1455
"Change, Copy, and Create Subscription" on page 1458
"Change, Copy, and Create Topic" on page 1462
"Clear Queue" on page 1471
"Clear Topic String" on page 1472
"Delete Authentication Information Object" on page 1473
"Delete Authority Record" on page 1474
z/OS "Delete CF Structure" on page 1476
"Delete Channel" on page 1476
"Delete Channel (MQTT)" on page 1478
"Delete Channel Listener" on page 1478
"Delete Namelist" on page 1479
"Delete Process" on page 1480
"Delete Queue" on page 1482
"Delete Service" on page 1484
z/OS "Delete Storage Class" on page 1484
"Delete Subscription" on page 1485
"Delete Topic" on page 1486
"Escape" on page 1488
"Escape (Response)" on page 1488
z/OS "Inquire Archive" on page 1489
z/OS "Inquire Archive (Response)" on page 1489
"Inquire Authentication Information Object" on page 1493
"Inquire Authentication Information Object (Response)" on page 1496
"Inquire Authentication Information Object Names" on page 1500
"Inquire Authentication Information Object Names (Response)" on page 1501
"Inquire Authority Records" on page 1502
"Inquire Authority Records (Response)" on page 1505
"Inquire Authority Service" on page 1508
"Inquire Authority Service (Response)" on page 1509
z/OS "Inquire CF Structure" on page 1509
z/OS "Inquire CF Structure (Response)" on page 1511
z/OS "Inquire CF Structure Names" on page 1514
z/OS "Inquire CF Structure Names (Response)" on page 1514
z/OS "Inquire CF Structure Status" on page 1515
z/OS "Inquire CF Structure Status (Response)" on page 1516
"Inquire Channel" on page 1520
"Inquire Channel (MQTT)" on page 1531
"Inquire Channel (Response)" on page 1532
"Inquire Channel Authentication Records" on page 1543
"Inquire Channel Authentication Records (Response)" on page 1547
"Inquire Channel Initiator" on page 1549

"Inquire Channel Initiator (Response)" on page 1550
"Inquire Channel Listener" on page 1552
"Inquire Channel Listener (Response)" on page 1554
"Inquire Channel Listener Status" on page 1556
"Inquire Channel Listener Status (Response)" on page 1558
"Inquire Channel Names" on page 1560
"Inquire Channel Names (Response)" on page 1562
"Inquire Channel Status" on page 1563
"Inquire Channel Status (MQTT)" on page 1576
"Inquire Channel Status (Response)" on page 1578
"Inquire Channel Status (Response) (MQTT)" on page 1589
"Inquire Cluster Queue Manager" on page 1591
"Inquire Cluster Queue Manager (Response)" on page 1595
"Inquire Communication Information Object" on page 1603
"Inquire Communication Information Object (Response)" on page 1604
"Inquire Connection" on page 1607
"Inquire Connection (Response)" on page 1611
"Inquire Entity Authority" on page 1618
"Inquire Entity Authority (Response)" on page 1620
z/OS "Inquire Group" on page 1623
z/OS "Inquire Group (Response)" on page 1623
z/OS "Inquire Log" on page 1625
z/OS "MQCMD_INQUIRE_LOG (Inquire Log) Response on z/OS" on page 1625
"Inquire Namelist" on page 1629
"Inquire Namelist (Response)" on page 1631
"Inquire Namelist Names" on page 1633
"Inquire Namelist Names (Response)" on page 1634
"Inquire Process" on page 1637
"Inquire Process (Response)" on page 1639
"Inquire Process Names" on page 1640
"Inquire Process Names (Response)" on page 1642
"Inquire Pub/Sub Status" on page 1642
"Inquire Pub/Sub Status (Response)" on page 1643
"Inquire Queue" on page 1646
"Inquire Queue (Response)" on page 1655
"Inquire Queue Manager" on page 1665
"Inquire Queue Manager (Response)" on page 1675
"MQCMD_INQUIRE_Q_MGR_STATUS (Inquire Queue Manager Status) on Distributed platforms" on page 1698
"MQCMD_INQUIRE_Q_MGR_STATUS (Inquire Queue Manager Status) Response on Distributed platforms" on page 1700
"Inquire Queue Names" on page 1703
"Inquire Queue Names (Response)" on page 1704
"Inquire Queue Status" on page 1705
"Inquire Queue Status (Response)" on page 1710
z/OS "Inquire Security" on page 1717
z/OS "Inquire Security (Response)" on page 1718
"Inquire Service" on page 1719
"Inquire Service (Response)" on page 1720

"Inquire Service Status" on page 1722
"Inquire Service Status (Response)" on page 1724
z/OS "Inquire SMDS" on page 1725
z/OS "Inquire SMDS (Response)" on page 1726
z/OS "Inquire SMDS Connection" on page 1727
z/OS "MQCMD_INQUIRE_SMDSCONN (Inquire SMDS Connection) Response on z/OS" on page 1727
z/OS "Inquire Storage Class" on page 1729
z/OS "Inquire Storage Class (Response)" on page 1731
z/OS "Inquire Storage Class Names" on page 1732
z/OS "Inquire Storage Class Names (Response)" on page 1734
"Inquire Subscription" on page 1734
"Inquire Subscription (Response)" on page 1738
"Inquire Subscription Status" on page 1742
"Inquire Subscription Status (Response)" on page 1744
z/OS "Inquire System" on page 1745
z/OS "MQCMD_INQUIRE_SYSTEM (Inquire System) Response on z/OS" on page 1746
"Inquire Topic" on page 1749
"Inquire Topic (Response)" on page 1753
"Inquire Topic Names" on page 1759
"Inquire Topic Names (Response)" on page 1760
"Inquire Topic Status" on page 1761
"Inquire Topic Status (Response)" on page 1762
z/OS "Inquire Usage" on page 1768
z/OS "Inquire Usage (Response)" on page 1769
z/OS "Move Queue" on page 1773
"Ping Channel" on page 1775
"Ping Queue Manager" on page 1778
"Purge Channel" on page 1779
z/OS "Recover CF Structure" on page 1779
"Refresh Cluster" on page 1780
"Refresh Queue Manager" on page 1781
"Refresh Security" on page 1784
z/OS "Reset CF Structure" on page 1786
"Reset Channel" on page 1787
"Reset Cluster" on page 1789
"Reset Queue Manager" on page 1790
"Reset Queue Statistics" on page 1791
"Reset Queue Statistics (Response)" on page 1792
z/OS "Reset SMDS" on page 1793
"Resolve Channel" on page 1794
z/OS "Resume Queue Manager" on page 1796
"Resume Queue Manager Cluster" on page 1797
z/OS "Reverify Security" on page 1798

z/OS "Set Archive" on page 1799
"Set Authority Record" on page 1802
"Set Channel Authentication Record" on page 1806
z/OS "Set Log" on page 1812
z/OS "Set System" on page 1816
"Start Channel" on page 1817
"Start Channel (MQTT)" on page 1820
"Start Channel Initiator" on page 1821
"Start Channel Listener" on page 1822
"Start Service" on page 1824
z/OS "Start SMDS Connection" on page 1825
"Stop Channel" on page 1825
"Stop Channel (MQTT)" on page 1829
z/OS "Stop Channel Initiator" on page 1830
"Stop Channel Listener" on page 1831
"Stop Connection" on page 1832
"Stop Service" on page 1832
z/OS "Stop SMDS Connection" on page 1833
z/OS "Suspend Queue Manager" on page 1834
"Suspend Queue Manager Cluster" on page 1834

How the 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 PCFs.

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. On distributed platforms, the parameters ***must*** occur in the 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 commands and responses" on page 1836). 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.

Table 85. MQIACF_COMMAND_INFO values

MQIACF_COMMAND_INFO value	**Meaning**
MQCMDI_CMDSCOPE_ACCEPTED	A command that specified *CommandScope* was entered. It has been passed to the one or more requested queue managers for processing
MQCMDI_CMDSCOPE_GENERATED	A command that specified *CommandScope* was generated in response to the command originally entered
MQCMDI_CMDSCOPE_COMPLETED	Processing for the command that specified *CommandScope* - 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

Table 85. MQIACF_COMMAND_INFO values (continued)	
MQIACF_COMMAND_INFO value	**Meaning**
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 *SecurityInterval* 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

z/OS In z/OS, PCF commands can return MQRC reason codes instead of MQRCCF codes

MQRCCF codes are used in UNIX, Linux or 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 MQRC_* error codes are given in the Messages and reason codes z/OS and IBM MQ for z/OS messages, completion, and reason codes documentation):

***Reason* (MQLONG)**
The value can be any of the following values:

MQRC_NONE
(0, X'000') No reason to report.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Message length greater than maximum for queue.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Not authorized for access.

MQRC_SELECTOR_ERROR
(2067, X'813') Attribute selector not valid.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.

MQRC_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 this product documentation, the commands and data responses are given in alphabetical order.

They can be usefully grouped as follows:

Authentication Information commands

- “Change, Copy, and Create Authentication Information Object” on page 1342
- “Delete Authentication Information Object” on page 1473
- “Inquire Authentication Information Object” on page 1493
- “Inquire Authentication Information Object Names” on page 1500

Authority Record commands

- “Delete Authority Record” on page 1474
- “Inquire Authority Records” on page 1502
- “Inquire Authority Service” on page 1508
- “Inquire Entity Authority” on page 1618
- “Set Authority Record” on page 1802

CF commands

z/OS

- “Backup CF Structure” on page 1341
- “Change, Copy, and Create CF Structure” on page 1351
- “Delete CF Structure” on page 1476
- “Inquire CF Structure” on page 1509
- “Inquire CF Structure Names” on page 1514
- “Inquire CF Structure Status” on page 1515
- “Recover CF Structure” on page 1779

Channel commands

- “Change, Copy, and Create Channel” on page 1356
- “Delete Channel” on page 1476

- “Inquire Channel” on page 1520
- z/OS “Inquire Channel Initiator” on page 1549
- “Inquire Channel Names” on page 1560
- “Inquire Channel Status” on page 1563
- “Ping Channel” on page 1775
- “Reset Channel” on page 1787
- “Resolve Channel” on page 1794
- “Start Channel” on page 1817
- z/OS “Start Channel Initiator” on page 1821
- “Stop Channel” on page 1825
- z/OS “Stop Channel Initiator” on page 1830

Channel commands (MQTT)

- “Change, Copy, and Create Channel (MQTT)” on page 1389
- “Delete Channel (MQTT)” on page 1478
- “Inquire Channel (MQTT)” on page 1531
- “Inquire Channel Status (MQTT)” on page 1576
- “Purge Channel” on page 1779
- “Start Channel (MQTT)” on page 1820
- “Stop Channel (MQTT)” on page 1829

Channel Authentication commands

- “Inquire Channel Authentication Records” on page 1543
- “Set Channel Authentication Record” on page 1806

Channel Listener commands

- “Change, Copy, and Create Channel Listener” on page 1394
- “Delete Channel Listener” on page 1478
- “Inquire Channel Listener” on page 1552
- “Inquire Channel Listener Status” on page 1556
- “Start Channel Listener” on page 1822
- “Stop Channel Listener” on page 1831

Cluster commands

- “Inquire Cluster Queue Manager” on page 1591
- “Refresh Cluster” on page 1780
- “Reset Cluster” on page 1789
- “Resume Queue Manager Cluster” on page 1797
- “Suspend Queue Manager Cluster” on page 1834

Communication Information commands

- “Change, Copy, and Create Communication Information Object” on page 1397

- “Delete Communication Information Object” on page 1478
- “Inquire Communication Information Object” on page 1603

Connection commands

- “Inquire Connection” on page 1607
- “Stop Connection” on page 1832

Escape command

- “Escape” on page 1488

Namelist commands

- “Change, Copy, and Create Namelist” on page 1400
- “Delete Namelist” on page 1479
- “Inquire Namelist” on page 1629
- “Inquire Namelist Names” on page 1633

Process commands

- “Change, Copy, and Create Process” on page 1403
- “Delete Process” on page 1480
- “Inquire Process” on page 1637
- “Inquire Process Names” on page 1640

Publish/subscribe commands

- “Change, Copy, and Create Subscription” on page 1458
- “Change, Copy, and Create Topic” on page 1462
- “Clear Topic String” on page 1472
- “Delete Subscription” on page 1485
- “Delete Topic” on page 1486
- “Inquire Pub/Sub Status” on page 1642
- “Inquire Subscription” on page 1734
- “Inquire Subscription Status” on page 1742
- “Inquire Topic” on page 1749
- “Inquire Topic Names” on page 1759
- “Inquire Topic Status” on page 1761

Queue commands

- “Change, Copy, and Create Queue” on page 1407
- “Clear Queue” on page 1471
- “Delete Queue” on page 1482
- “Inquire Queue” on page 1646
- “Inquire Queue Names” on page 1703
- “Inquire Queue Status” on page 1705
-

“Move Queue” on page 1773

- “Reset Queue Statistics” on page 1791

Queue Manager commands

- “Change Queue Manager” on page 1424
- “Inquire Queue Manager” on page 1665
- “MQCMD_INQUIRE_Q_MGR_STATUS (Inquire Queue Manager Status) on Distributed platforms” on page 1698
- “Ping Queue Manager” on page 1778
- “Refresh Queue Manager” on page 1781
- “Reset Queue Manager” on page 1790
- z/OS “Resume Queue Manager” on page 1796
- z/OS “Suspend Queue Manager” on page 1834

Security commands

- “Change Security” on page 1451
- “Inquire Security” on page 1717
- “Refresh Security” on page 1784
- z/OS “Reverify Security” on page 1798

Service commands

- “Change, Copy, and Create Service” on page 1453
- “Delete Service” on page 1484
- “Inquire Service” on page 1719
- “Inquire Service Status” on page 1722
- “Start Service” on page 1824
- “Stop Service” on page 1832

SMDS commands

z/OS

- “Change SMDS” on page 1452
- “Inquire SMDS” on page 1725
- “Inquire SMDS Connection” on page 1727
- “Reset SMDS” on page 1793
- “Start SMDS Connection” on page 1825
- “Stop SMDS Connection” on page 1833

Storage class commands

z/OS

- z/OS “Change, Copy, and Create Storage Class” on page 1455
- “Delete Storage Class” on page 1484
- “Inquire Storage Class” on page 1729
- “Inquire Storage Class Names” on page 1732

System commands

z/OS

- “Inquire Archive” on page 1489
- “Set Archive” on page 1799
- “Inquire Group” on page 1623
- “Inquire Log” on page 1625
- “Set Log” on page 1812
- “Inquire System” on page 1745
- “Set System” on page 1816
- “Inquire Usage” on page 1768

Data responses to commands

- “Escape (Response)” on page 1488
- z/OS “Inquire Archive (Response)” on page 1489
- “Inquire Authentication Information Object (Response)” on page 1496
- “Inquire Authentication Information Object Names (Response)” on page 1501
- “Inquire Authority Records (Response)” on page 1505
- “Inquire Authority Service (Response)” on page 1509
- z/OS “Inquire CF Structure (Response)” on page 1511
- z/OS “Inquire CF Structure Names (Response)” on page 1514
- z/OS “Inquire CF Structure Status (Response)” on page 1516
- “Inquire Channel (Response)” on page 1532
- “Inquire Channel Authentication Records (Response)” on page 1547
- “Inquire Channel Initiator (Response)” on page 1550
- “Inquire Channel Listener (Response)” on page 1554
- “Inquire Channel Listener Status (Response)” on page 1558
- “Inquire Channel Names (Response)” on page 1562
- “Inquire Channel Status (Response)” on page 1578
- “Inquire Channel Status (Response) (MQTT)” on page 1589
- “Inquire Cluster Queue Manager (Response)” on page 1595
- “Inquire Communication Information Object (Response)” on page 1604
- “Inquire Connection (Response)” on page 1611
- “Inquire Entity Authority (Response)” on page 1620
- z/OS “Inquire Group (Response)” on page 1623
- z/OS “MQCMD_INQUIRE_LOG (Inquire Log) Response on z/OS” on page 1625
- “Inquire Namelist (Response)” on page 1631
- “Inquire Namelist Names (Response)” on page 1634
- “Inquire Process (Response)” on page 1639
- “Inquire Process Names (Response)” on page 1642
- “Inquire Pub/Sub Status (Response)” on page 1643
- “Inquire Queue (Response)” on page 1655

- “Inquire Queue Manager (Response)” on page 1675
- “MQCMD_INQUIRE_Q_MGR_STATUS (Inquire Queue Manager Status) Response on Distributed platforms” on page 1700
- “Inquire Queue Names (Response)” on page 1704
- “Reset Queue Statistics (Response)” on page 1792
- “Inquire Queue Status (Response)” on page 1710
- z/OS “Inquire Security (Response)” on page 1718
- “Inquire Service (Response)” on page 1720
- “Inquire Service Status (Response)” on page 1724
- z/OS “Inquire Storage Class (Response)” on page 1731
- z/OS “Inquire Storage Class Names (Response)” on page 1734
- z/OS “Inquire SMDS (Response)” on page 1726
- z/OS “MQCMD_INQUIRE_SMDSCONN (Inquire SMDS Connection) Response on z/OS” on page 1727
- “Inquire Subscription (Response)” on page 1738
- “Inquire Subscription Status (Response)” on page 1744
- z/OS “MQCMD_INQUIRE_SYSTEM (Inquire System) Response on z/OS” on page 1746
- “Inquire Topic (Response)” on page 1753
- “Inquire Topic Names (Response)” on page 1760
- “Inquire Topic Status (Response)” on page 1762
- z/OS “Inquire Usage (Response)” on page 1769

z/OS Backup CF Structure

The Backup CF Structure (MQCMD_BACKUP_CF_STRUC) command initiates a CF application structure backup.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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: 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.

***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.

Change, Copy, and Create Authentication Information Object

The Change authentication information 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 Security 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).

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 *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 Security 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 Security 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 platforms other than z/OS, the maximum length is MQ_AUTH_INFO_CONN_NAME_LENGTH. On z/OS, it 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.

V 8.0.0.3 AuthenticationMethod (MQCFIN)
Authentication methods for user passwords (parameter identifier: MQIA_AUTHENTICATION_METHOD). Possible values are:

MQAUTHENTICATE_OS
Use the traditional UNIX platforms 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 UNIX 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.

V 8.0.0.2 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 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 all the groups to which the specified user belongs. The attribute to query is defined by the FindGroup value, typically *memberOf*.

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.

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

V 8.0.0.2 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).

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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.

A privileged user is one that has full administrative authorities for IBM MQ. See Privileged users for more information.

z/OS (This setting is not allowed on z/OS systems.)

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.

z/OS 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.

A privileged user is one that has full administrative authorities for IBM MQ. See Privileged users for more information.

z/OS (This setting is not allowed on z/OS systems.)

V 8.0.0.2 **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.

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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* .

V 8.0.0.2 **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*.

If left blank, the queue manager uses *member* or *memberOf* depending on the setting of **AuthorizationMethod**.

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2 , and is not applicable to Windows platforms.

The maximum length is MQ_LDAP_FIELD_LENGTH.

V 8.0.0.2 **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.

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

The maximum length is MQ_LDAP_FIELD_LENGTH.

V 8.0.0.2 **GroupNesting (MQCFIN)**
Whether groups are members of other groups (parameter identifier: MQCA_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.

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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* .

On platforms other than z/OS, the maximum length is MQ_DISTINGUISHED_NAME_LENGTH. On z/OS, it 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/OSonly.

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
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 parameter 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 using the MQQSGD_GROUP object of the same name as the *ToAuthInfoName* object (for Copy) or the *AuthInfoName* object (for Create).

QSGDisposition	Change	Copy, Create

MQQSGD_GROUP	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.
	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:
	`DEFINE AUTHINFO(name)`
	`REPLACE QSGDISP(COPY)`
	The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.
		The object definition resides in the shared repository. This definition is allowed only if the queue manager is in a queue-sharing group.
		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:
		`DEFINE AUTHINFO(name)`
		`REPLACE QSGDISP(COPY)`
		The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_PRIVATE	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.
MQQSGD_Q_MGR	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 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 SSL/TLS (parameter identifier MQIA_LDAP_SECURE_COMM).

MQSECCOMM_YES
Connectivity to the LDAP server is made securely using SSL/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 SSL/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 SSL/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.

z/OS Change, Copy, and Create CF Structure

The Change CF Structure 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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)**
CFConlos (parameter identifier: MQIA_CF_CFCONLOS).

Specifies the action to be taken when a queue manager loses connectivity to the CF structure. The following constant names are valid values for CFConlos:

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.

Only queue managers with a command level of 710 or higher and with OPMODE set to enable IBM WebSphere MQ 7.1.0 new functions can connect to a CF structure at *CFLevel* (5).

Structures are required to be at CFLEVEL(5) to support toleration of loss of connectivity.

z/OS 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.

***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.

***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)**
Offload (parameter identifier: MQIA_CF_OFFLOAD).

Specifies the OFFLOAD option for large (>63 K) shared messages on z/OS. 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.

MQCFOFFLD_NONE
Used when the property *Offload* has not been explicitly set.

Value can not be set unless CFLEVEL(5) is defined.

The default value is MQCFOFFLD_NONE if not at CFLEVEL(5).

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.

z/OS 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 *0FFLD1SZ*.

Value can not be set unless CFLEVEL(5) is defined.

***0FFLD2TH* (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 *0FFLD2SZ*.

Value can not be set unless CFLEVEL(5) is defined.

***0FFLD3TH* (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 *0FFLD3SZ*.

Value can not be set unless CFLEVEL(5) is defined.

***Recauto* (MQCFIN)**
Recauto (parameter identifier: MQIA_CF_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. 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).

***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.

***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 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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.

V 8.0.0.4

Table 86. Change, Copy, Create Channel parameters

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	V 8.0.0.4 AMQP
V 8.0.0.4 *AMQPKeepAlive*									✓
BatchHeartBeat	✓	✓					✓	✓	
BatchInterval	✓	✓					✓	✓	
BatchDataLimit	✓	✓					✓	✓	
BatchSize	✓	✓	✓	✓			✓	✓	
CertificateLabel	✓	✓	✓	✓			✓	✓	✓
ChannelDesc	✓	✓	✓	✓	✓	✓	✓	✓	✓
ChannelMonitoring	✓	✓	✓	✓		✓	✓	✓	
ChannelStatistics	✓	✓	✓	✓			✓	✓	
ChannelName (see footnote 1)	✓	✓	✓	✓	✓	✓	✓	✓	✓
ChannelType (see footnote 3)	✓	✓	✓	✓	✓	✓	✓	✓	✓
ClientChannelWeight					✓				

Table 86. Change, Copy, Create Channel parameters (continued)									
Parameter	**Sender**	**Server**	**Receiver**	**Requester**	**Client conn**	**Server conn**	**Cluster sender**	**Cluster receiver**	V 8.0.0.4 **AMQP**
ClusterName							✓	✓	
ClusterNameList							✓	✓	
CLWLChannelPriority							✓	✓	
CLWLChannelRank							✓	✓	
CLWLChannelWeight							✓	✓	
z/OS z/OS *CommandScope*	✓	✓	✓	✓	✓	✓	✓	✓	
ConnectionAffinity					✓				
ConnectionName	✓	✓		✓	✓		✓	✓	
DataConversion	✓	✓		✓	✓		✓	✓	
DefaultChannelDisposition	✓	✓	✓	✓		✓	✓	✓	
DefReconnect					✓				
DiscInterval	✓	✓				✓	✓	✓	
FromChannelName (see footnote 2)	✓	✓	✓	✓	✓	✓	✓	✓	
HeaderCompession	✓	✓	✓	✓	✓	✓	✓	✓	
HeartBeatInterval	✓	✓	✓	✓	✓	✓	✓	✓	
KeepAliveInterval	✓	✓	✓	✓	✓	✓	✓	✓	
LocalAddress	✓	✓		✓	✓		✓	✓	✓
LongRetryCount	✓	✓					✓	✓	
LongRetryInterval	✓	✓					✓	✓	
MaxInstances						✓			✓
MaxInstancesPerClient						✓			

Table 86. Change, Copy, Create Channel parameters (continued)									
Parameter	**Sender**	**Server**	**Receiver**	**Requester**	**Client conn**	**Server conn**	**Cluster sender**	**Cluster receiver**	V 8.0.0.4 **AMQP**
MaxMsgLength	✓	✓	✓	✓	✓	✓	✓	✓	✓
MCAName	✓	✓		✓			✓		
MCAType	✓	✓		✓			✓	✓	
MCAUserIdentifier			✓	✓		✓		✓	✓
MessageCompression	✓	✓	✓	✓	✓	✓	✓	✓	
ModeName	✓	✓		✓	✓		✓	✓	
MsgExit	✓	✓	✓	✓			✓	✓	
MsgRetryCount			✓	✓				✓	
MsgRetryExit			✓	✓				✓	
MsgRetryInterval			✓	✓				✓	
MsgRetryUserData			✓	✓				✓	
MsgUserData	✓	✓	✓	✓			✓	✓	
NetworkPriority								✓	
NonPersistentMsgSpeed	✓	✓	✓	✓			✓	✓	
Password	✓	✓		✓	✓		✓		
V 8.0.0.4 *Port*									✓
PropertyControl	✓	✓					✓	✓	
PutAuthority			✓	✓		✓		✓	
QMgrName					✓				
z/OS z/OS *QSGDisposition*	✓	✓	✓	✓	✓	✓	✓	✓	
ReceiveExit	✓	✓	✓	✓	✓	✓	✓	✓	
ReceiveUserData	✓	✓	✓	✓	✓	✓	✓	✓	
Replace	✓	✓	✓	✓	✓	✓	✓	✓	

Table 86. Change, Copy, Create Channel parameters (continued)									
Parameter	**Sender**	**Server**	**Receiver**	**Requester**	**Client conn**	**Server conn**	**Cluster sender**	**Cluster receiver**	V 8.0.0.4 **AMQP**
SecurityExit	✓	✓	✓	✓	✓	✓	✓	✓	
SecurityUserData	✓	✓	✓	✓	✓	✓	✓	✓	
SendExit	✓	✓	✓	✓	✓	✓	✓	✓	
SendUserData	✓	✓	✓	✓	✓	✓	✓	✓	
SeqNumberWrap	✓	✓	✓	✓			✓	✓	
SharingConversations					✓	✓			
ShortRetryCount	✓	✓					✓	✓	
ShortRetryInterval	✓	✓					✓	✓	
SSLCipherSpec	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSLClientAuth		✓	✓	✓		✓		✓	✓
SSLPeerName	✓	✓	✓	✓	✓	✓	✓	✓	✓
ToChannelName (see footnote 2)	✓	✓	✓	✓	✓	✓	✓	✓	
TpName	✓	✓		✓	✓	✓	✓	✓	
V 8.0.0.4 *TpRoot*									✓
TransportType	✓	✓	✓	✓	✓	✓	✓	✓	
V 8.0.0.4 *UseCltId*									✓
UseDLQ	✓	✓	✓	✓			✓	✓	
UserIdentifier	✓	✓		✓	✓		✓		
XmitQName	✓	✓							

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 `TrpType` 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 to AIX, HP-UX, Linux, IBM i, Solaris, and Windows only. See Configuring a queue manager cluster for more details.

***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.

V 8.0.0.4 **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.

V 8.0.0.4 **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)

V 8.0.0.4

AMQPKeepAlive **(MQCFIN)**
The AMQP channel keep alive interval (parameter identifier: MQIACH_AMQP_KEEP_ALIVE).

The keep alive time for an AMQP channel in seconds. 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-eating 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).

This interval is the approximate time in milliseconds that a channel keeps a batch open, if fewer than *BatchSize* messages have been transmitted in the current batch.

If *BatchInterval* is greater than zero, the batch is terminated by whichever of the following situations occurs first:

- *BatchSize* messages have been sent, or
- *BatchInterval* milliseconds have elapsed since the start of the batch.

If *BatchInterval* is zero, the batch is terminated by whichever of the following situations occurs first:

- *BatchSize* messages have been sent, or
- *BatchDataLimit* bytes have been sent, or
- the transmission queue becomes empty.

BatchInterval must be in the range 0 - 999999999.

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.

This 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, 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.

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 Version 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, 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.

***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

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.

***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 platforms other than z/OS, the maximum length of the string is 264. On z/OS, it 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, and UNIX systems, 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.

  On AIX, HP-UX, IBM i, Linux, Solaris, and Windows, 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.

- 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 a 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 MQCONNX, the client reconnects automatically.

MQRCN_Q_MGR
Unless overridden by MQCONNX, 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 MQCONNX MQI call.

Table 87. Automatic reconnection depends on the values set in the application and in the channel definition

DefReconnect	**Reconnection options set in the application**			
	MQCNO_RECONNECT	MQCNO_RECONNECT_Q_MGR	MQCNO_RECONNECT_AS_DEF	MQCNO_RECONNECT_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 in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, and z/OS.

- 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 in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, Linux, and z/OS.

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 switched on 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 Solaris, 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 platforms other than z/OS, 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)**
V 8.0.0.4 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.

V 8.0.0.4 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.

V 8.0.0.4 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 AIX, HP-UX, IBM i, Solaris, Windows, and Linux, 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)**
Header data compression techniques supported by the channel (parameter identifier: MQIACH_MSG_COMPRESSION). 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.

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, HP Integrity NonStop Server, UNIX systems, 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: AIX, HP-UX, IBM i, Solaris, Windows, and Linux.

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, HP Integrity NonStop Server, and UNIX systems, 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.

V 8.0.0.4

***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 messages 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.

z/OS 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.

z/OS 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.

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:

QSGDisposition	Change	Copy, Create
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 using the MQQSGD_GROUP object of the same name as the *ToChannelName* object (for Copy) or *ChannelName* object (for Create).

MQQSGD_GROUP	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.
	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:
	`DEFINE CHANNEL(channel-name)`
	`CHLTYPE(type)`
	`REPLACE QSGDISP(COPY)`
	The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.
		The object definition resides in the shared repository. This definition is allowed only if the queue manager is in a queue-sharing group.
		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:
		`DEFINE CHANNEL(channe-name)`
		`CHLTYPE(type)`
		`REPLACE QSGDISP(COPY)`
		The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_PRIVATE	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.
MQQSGD_Q_MGR	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 and UNIX systems, it is of the form

```
libraryname(functionname)
```

  Note: On IBM i systems, the following form is also supported for compatibility with older releases:

```
progname libname
```

  where *progname* 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 MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_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 MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

***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 two 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.

The SSLCipherSpec 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 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.

The following table shows the CipherSpecs that can be used with IBM MQ SSL and TLS.

Platform support "1" on page 1383	CipherSpec name	Protocol used	MAC algorithm	Encryption algorithm	Encryption bits	FIPS "2" on page 1383	Suite B
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Yes	No
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_256_CBC_SHA "3" on page 1383	TLS 1.0	SHA-1	AES	256	Yes	No
All	ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No
All	ECDHE_ECDSA_AES_256_CBC_SHA384 "3" on page 1383	TLS 1.2	SHA-384	AES	256	Yes	No

Platform support "1" on page 1383	CipherSpec name	Protocol used	MAC algorithm	Encryption algorithm	Encryption bits	FIPS "2" on page 1383	Suite B
distributed	ECDHE_ECDSA_AES_128_GCM_SHA256 "4" on page 1383	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	128 bit
distributed	ECDHE_ECDSA_AES_256_GCM_SHA384 "3" on page 1383 "4" on page 1383	TLS 1.2	AEAD AES-128 GCM	AES	256	Yes	192 bit
All	ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No
All	ECDHE_RSA_AES_256_CBC_SHA384 "3" on page 1383	TLS 1.2	SHA-384	AES	256	Yes	No
All	ECDHE_RSA_AES_128_GCM_SHA256 "4" on page 1383	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	No
All	ECDHE_RSA_AES_256_GCM_SHA384 "3" on page 1383 "4" on page 1383	TLS 1.2	AEAD AES-128 GCM	AES	SHA384	Yes	No
IBM i "5" on page 1383	ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	AEAD AES-128 GCM	AES	SHA256	Yes	No
IBM i	ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	AEAD AES-128 GCM	3DES	SHA256	Yes	No
IBM i	ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	AEAD AES-128 GCM	3DES	SHA256	Yes	No
IBM i	ECDHE_RSA_RC4_128_SHA256	TLS 1.2	AEAD AES-128 GCM	RSA	SHA256	Yes	No
IBM i	ECDHE_RSA_NULL_SHA256	TLS 1.2	AEAD AES-128 GCM	RSA	SHA256	Yes	No
IBM i	ECDHE_ECDSA_NULL_SHA256	TLS 1.2	AEAD AES-128 GCM	ECDSA	SHA256	Yes	No
IBM i	ECDHE_ECDSA_AES_256_GCM_SHA384 "3" on page 1383 "4" on page 1383	TLS 1.2	AEAD AES-128 GCM	AES	SHA384	Yes	No
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Yes	No

Platform support "1" on page 1383	CipherSpec name	Protocol used	MAC algorithm	Encryption algorithm	Encryption bits	FIPS "2" on page 1383	Suite B
z/OS UNIX Linux Windows	TLS_RSA_WITH_AES_256_CBC_SHA256 "3" on page 1383	TLS 1.2	SHA-256	AES	256	Yes	No
distributed	TLS_RSA_WITH_AES_128_GCM_SHA256 "4" on page 1383	TLS 1.2	AEAD AES-128 GCM	AES	128	Yes	No
distributed	TLS_RSA_WITH_AES_256_GCM_SHA384 "3" on page 1383 "4" on page 1383	TLS 1.2	AEAD AES-128 GCM	AES	256	Yes	No

Notes:

1. If no specific platform is noted, the CipherSpec is available on all platforms.
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. This CipherSpec cannot be used to secure a connection from the MQ Explorer to a queue manager unless the appropriate unrestricted policy files are applied to the JRE used by the Explorer.
4. Following a recommendation by NIST, GCM CipherSpecs have a restriction which means that after 2^32 TLS records are sent, using the same session key, the connection is terminated with message AMQ9288.

   To prevent this error from happening: avoid using GCM Ciphers, enable secret key reset, or start your IBM MQ queue manager with the environment variable GSK_ENFORCE_GCM_RESTRICTION=GSK_FALSE set.

   Important: The GCM restriction is active, regardless of the FIPS mode being used.
5. The CipherSpecs listed as supported on IBM i apply to Versions 7.2 and 7.3 of IBM i.

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 UNIX systems, Windows systems, and z/OS, 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.
- On UNIX and Windows systems, 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 SSL client.

The SSL client is the end of the message channel that initiates the connection. The SSL 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 SSL or TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different SSL or 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.

On platforms other than z/OS, 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 SSL 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:

Attribute	Description
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

Attribute	Description
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\*')
```

***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, HP Integrity NonStop Server, UNIX systems, 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 channels.

V 8.0.0.4

***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.

V 8.0.0.4

***UseCltId* (MQCFIN)**
Determines how authorization checks are done for AMQP channels. (parameter identifier: MQIACH_USE_CLIENT_ID).

Specifies that the client ID should be used for authorization checks for an AMQP channel, instead of the MCAUSER attribute value.

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 and UNIX systems, 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 1333.

***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
SSL CipherSpec not valid.

MQRCCF_SSL_CLIENT_AUTH_ERROR
SSL client authentication not valid.

MQRCCF_SSL_PEER_NAME_ERROR
SSL 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)

The Change Channel 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.

***MqiachProtocol* (MQCFIL)**
Client protocols supported by the MQTT channel (parameter identifier: MQIACH_PROTOCOL).

The value can be one or more of the following values:

V 8.0.0.3 **MQPROTO_MQTTV311**
The channel accepts connections from clients using the protocol defined by the MQTT Version 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 IBM MQ Telemetry.

If you specify no client protocols, the channel accepts connections from clients using any of the supported protocols.

V 8.0.0.3 If you are using IBM MQ Version 8.0.0, Fix Pack 3 or later, and your configuration includes an MQTT channel that was last modified in an earlier version of IBM MQ, 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 SSL client.

The SSL client is the end of the message channel that initiates the connection. The SSL 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, SSL 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.

***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 1333.

***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
SSL CipherSpec not valid.

MQRCCF_SSL_CLIENT_AUTH_ERROR
SSL client authentication not valid.

MQRCCF_SSL_PEER_NAME_ERROR
SSL 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 Listener

The Change Channel Listener 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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

Change, Copy, and Create Communication Information Object

The Change Communication Information Object 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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)

***CominfoName* (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)

***FromCominfoName* (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.

***ToCominfoName* (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)

***CominfoName* (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 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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.

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 *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.

z/OS

***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 SSL, 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.

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:

QSGDisposition	Change	Copy, Create
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 using the MQQSGD_GROUP object of the same name as the *ToNameListName* object (for Copy) or *NameListName* object (for Create).

MQQSGD_GROUP	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.
	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:
	`DEFINE NAMELIST(name)`
	`REPLACE QSGDISP(COPY)`
	The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.
		The object definition resides in the shared repository. This is allowed only if the queue manager is in a queue-sharing group.
		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:
		`DEFINE NAMELIST(name)`
		`REPLACE QSGDISP(COPY)`
		The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_PRIVATE	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.

QSGDisposition	Change	Copy, Create
MQQSGD_Q_MGR	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 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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.

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 *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_WINDOWS_NT
Windows or Windows 95, Windows 98 application.

MQAT_DOS
DOS client application.

MQAT_WINDOWS
Windows client application.

MQAT_UNIX
UNIX application.

MQAT_AIX
AIX application (same value as MQAT_UNIX).

MQAT_CICS
CICS transaction.

MQAT_NSK
HP Integrity NonStop Server application.

z/OS **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 HP Integrity NonStop Server: MQAT_NSK, MQAT_DOS, MQAT_WINDOWS, and MQAT_DEFAULT are supported.

- On UNIX systems: 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.
- z/OS On z/OS: MQAT_DOS, MQAT_IMS MQAT_MVS, MQAT_UNIX, MQAT_CICS, and MQAT_DEFAULT are supported.

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. 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.

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:

QSGDisposition	Change	Copy, Create
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 using the MQQSGD_GROUP object of the same name as the *ToProcessName* object (for Copy) or *ProcessName* object (for Create).

MQQSGD_GROUP	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.
	`DEFINE PROCESS(process-name)`
	`REPLACE QSGDISP(COPY)`
	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.
		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.
		`DEFINE PROCESS(process-name)`
		`REPLACE QSGDISP(COPY)`
		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.

MQQSGD_PRIVATE	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.
MQQSGD_Q_MGR	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 *ApplId*) 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 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 *QSGDisposition* . 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.

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 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 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.

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 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).

V 8.0.0.4 This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name and value pairs have the form NAME(VALUE). Single quotation marks must be escaped with another single quotation mark.

CAPEXPRY (*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 CAPEXPRY 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.

***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 in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, and Linux.

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.

***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
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, HP Integrity NonStop Server, HP-UX, IBM i, Solaris, Linux, Windows, 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 if the queue is on AIX, HP-UX, IBM i, Solaris, Linux, Windows, or z/OS
- 640,000 if the queue is on any other IBM MQ platform.

***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 on AIX, HP-UX, Linux, IBM i, Solaris, Windows, and z/OS. 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 1327.

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 1327.

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 1327.

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 1327.

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:

QSGDisposition	Change	Copy, Create
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 using the MQQSGD_GROUP object of the same name as the *ToQName* object (for Copy) or the *QName* 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.

MQQSGD_GROUP	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.
	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:
	`DEFINE QUEUE(q-name)`
	`REPLACE QSGDISP(COPY)`
	The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.
		The object definition resides in the shared repository. This value is allowed only in a shared queue manager environment.
		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:
		`DEFINE QUEUE(q-name)`
		`REPLACE QSGDISP(COPY)`
		The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_PRIVATE	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.
MQQSGD_Q_MGR	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. For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.

QSGDisposition	Change	Copy, Create
MQQSGD_SHARED	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.	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: • *CFStructure* is nonblank • *IndexType* is not MQIT_MSG_TOKEN • The queue is not one of the following: – SYSTEM.CHANNEL.INITQ – SYSTEM.COMMAND.INPUT

***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.

MQMON_Q_MGR
The value of the queue manager's *QueueMonitoring* parameter is inherited by the queue.

MQMON_LOW
If the value of the queue manager *QueueMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on. The rate of data collection is low for this queue.

MQMON_MEDIUM
If the value of the queue manager *QueueMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on. The rate of data collection is moderate for this queue.

MQMON_HIGH
If the value of the queue manager *QueueMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on. The rate of data collection is high for this queue.

***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

This parameter is valid only on IBM i, UNIX systems, and Windows.

***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 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 UNIX systems, 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.

z/OS ***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.

***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 1333.

***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.

Change Queue Manager

The Change Queue Manager (MQCMD_CHANGE_Q_MGR) command changes the specified attributes of the queue manager.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

For any optional parameters that are omitted, the value does not change.

Required parameters:
None

Optional parameters (Change Queue Manager)

***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 MQCONNX API call.

This parameter is valid only on IBM i, UNIX, Linux, and Windows systems.

***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 IBM i, UNIX, Linux, and Windows systems.

***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. This value is not supported on z/OS.

***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:

- z/OS On z/OS: *ibmWebSphereMQXXXX* where *XXXX* is the queue manager name.

  See z/OS systems for more information.
- 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 UNIX and Linux and Windows, *ibmwebspheremqxxxx* where *xxxx* is the queue manager name folded to lowercase.

***CertificateValPolicy* (MQCFIN)**
Specifies which SSL/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 UNIX, 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 an SSL channel. If the process pooling process has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command **REFRESH SECURITY TYPE(SSL)**. The process pooling process is amqrmppa on UNIX, Linux, and Windows systems.
- 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.

You can select MQCFCONLOS_TOLERATE only if all the queue managers in the queue-sharing group are at command level 710 or greater and have OPMODE set to NEWFUNC.

***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, UNIX, 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, UNIX, 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 1356.

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: IBM i, z/OS, UNIX, 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.

***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 IBM i, UNIX, Linux, and Windows systems.

***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.

On z/OS, 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.

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 switched off.

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.

On z/OS, 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 switched off.

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, 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.

***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 1356.

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.

***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 IBM i, UNIX, Linux, and Windows systems.

***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 UNIX, Linux, and Windows platforms) or the security component (on z/OS platforms) 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 and 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. This behavior is implicit in versions of IBM WebSphere MQ, earlier than Version 7.5. In earlier versions, the queue manager attribute DefClusterXmitQueueType was not present.

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. Refer to 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. Refer to 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 QSG name when they connect.

z/OS See Unit of recovery disposition in a queue-sharing group.

z/OS ***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.

z/OS ***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.

***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.

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.

***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 IBM i, UNIX, Linux, and Windows systems.

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.

***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 IBM i, UNIX, Linux, and Windows systems.

***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 IBM i, UNIX, Linux, and Windows systems.

***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 WebSphere Message Broker (now known as IBM Integration Bus) 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.

z/OS ***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:

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.

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 IBM i, UNIX, Linux, and Windows systems.

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.
- Tthis 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 ***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. 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 SSL 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/SSL 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:

- On IBM i, UNIX, Linux, and Windows systems when a new channel process is started.
- For channels that run as threads of the channel initiator on IBM i, UNIX, Linux, and Windows systems, when the channel initiator is restarted.
- For channels that run as threads of the listener on IBM i, UNIX, Linux, and Windows systems, when the listener is restarted.
- On z/OS, when the channel initiator is restarted.
- When a **REFRESH SECURITY TYPE(SSL)** command is issued.
- 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 LDAPCRL 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 SSL 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 supported on UNIX, Linux, and Windows systems only.

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 IBM Global Security Kit (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 SSL 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 SSL key repository.

***SSLEvent* (MQCFIN)**
Controls whether SSL 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, UNIX, Linux, and Windows platforms only.

The value can be any of the following values:

MQSSL_FIPS_NO
IBM MQ provides an implementation of SSL 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 SSL 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 UNIX, Linux, and Windows systems, when a new channel process is started.
- For channels that run as threads of the channel initiator on UNIX, Linux, and Windows systems, when the channel initiator is restarted.
- For channels that run as threads of the listener on UNIX, Linux, and Windows systems, 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 an SSL channel. If the process pooling process has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command **REFRESH SECURITY TYPE(SSL)**. The process pooling process is **amqrmppa** on UNIX, Linux, and Windows systems.
- 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 SSL 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* , where *keyfile* is specified without the suffix (.kdb), and identifies a GSKit key database file. 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 SSLKEYR parameter to a value other than *SYSTEM, IBM MQ unregisters the queue manager as an application with DCM.
- On UNIX, it is of the form *pathname*/*keyfile* and on Windows *pathname**keyfile* , where *keyfile* is specified without the suffix (.kdb), and identifies a GSKit key database file. The default value for UNIX platforms is /var/mqm/qmgrs/QMGR/ssl/key, and on Windows it is C:\Program Files\IBM\WebSphere MQ\qmgrs\QMGR\ssl\key, where QMGR is replaced by the queue manager name (on UNIX, Linux, and Windows).

On IBM i, UNIX, Linux, and Windows systems, the syntax of this parameter is validated to ensure that it contains a valid, absolute, directory path.

If SSLKEYR is blank, or is a value that does not correspond to a key ring or key database file, channels using SSL fail to start.

Changes to SSLKeyRepository become effective:

- On IBM i, UNIX, Linux, and Windows platforms, when a new channel process is started.
- For channels that run as threads of the channel initiator on IBM i, UNIX, Linux, and Windows platforms, when the channel initiator is restarted.
- For channels that run as threads of the listener on IBM i, UNIX, Linux, and Windows platforms, when the listener is restarted.

- On z/OS, when the channel initiator is restarted.

***SSLKeyResetCount* (MQCFIN)**
SSL key reset count (parameter identifier: MQIA_SSL_RESET_COUNT).

Specifies when SSL 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 an SSL/TLS secret key reset count between 1 byte through 32 KB, SSL/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 SSL/TLS secret key reset values.

***SSLTasks* (MQCFIN)**
Number of server subtasks to use for processing SSL calls (parameter identifier: MQIA_SSL_TASKS). This parameter applies to z/OS only.

The number of server subtasks to use for processing SSL calls. To use SSL 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.

***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 IBM i, UNIX, Linux, and Windows.

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 to z/OS only.

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 to z/OS only.

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 to z/OS only.

***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 1333.

***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.

IBM i **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 UNIX, Linux and Windows

z/OS Change Security

The Change Security command changes specified attributes of an existing security definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

The Change Security (MQCMD_CHANGE_SECURITY) command defines system-wide security options.

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.

z/OS Change SMDS

The Change SMDS (MQCMD_CHANGE_SMDS) command changes the attributes of shared message data set.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

The Change SMDS (MQCMD_CHANGE_SMDS) 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.

Change, Copy, and Create Service

The Change Service 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

z/OS Change, Copy, and Create Storage Class

The Change Storage Class 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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:

QSGDisposition	Change	Copy, Create
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 using the MQQSGD_GROUP object of the same name as the *ToStorageClassName* object (for Copy) or the *StorageClassName* object (for Create).

QSGDisposition	Change	Copy, Create

MQQSGD_GROUP	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.
	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:
	`DEFINE STGCLASS(storage-class)`
	`REPLACE QSGDISP(COPY)`
	The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.
		The object definition resides in the shared repository. This parameter is allowed only if the queue manager is in a queue-sharing group.
		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:
		`DEFINE STGCLASS(storage-class)`
		`REPLACE QSGDISP(COPY)`
		The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_PRIVATE	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.
MQQSGD_Q_MGR	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 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 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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_SUBSCRIPTION_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).

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 *ToSubscriptionName* and the disposition MQQSGD_GROUP is used.

The maximum length of the string is MQ_SUBSCRIPTION_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)

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.

***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 *ApplIdentityData* 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 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.

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 WebSphere Message Broker. This value is the supplied default value.

Change, Copy, and Create Topic

The Change Topic 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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).

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 *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)

***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`).

V 8.0.0.4 This attribute contains the values of attributes, as pairs of attribute name and value, separated by at least one space. The attribute name and value pairs have the form NAME(VALUE). Single quotation marks must be escaped with another single quotation mark.

CAPEXPRY (*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 CAPEXPRY 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-ridden 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:

QSGDisposition	Change	Copy, Create
MQQSGD_COPY	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 by using the MQQSGD_GROUP object of the same name as the *ToTopicName* object (for Copy) or *TopicName* object (for Create).

QSGDisposition	Change	Copy, Create

MQQSGD_GROUP	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.
	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:
	`DEFINE TOPIC(name)`
	`REPLACE QSGDISP(COPY)`
	The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.
		The object definition resides in the shared repository. This definition is allowed only if the queue manager is in a queue-sharing group.
		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:
		`DEFINE TOPIC(name)`
		`REPLACE QSGDISP(COPY)`
		The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_PRIVATE	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.
MQQSGD_Q_MGR	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-subcription 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.

Clear Queue

The Clear Queue (MQCMD_CLEAR_Q) command deletes all the messages from a local queue.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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_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 1333.

***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.

Clear Topic String

The Clear Topic String (MQCMD_CLEAR_TOPIC_STRING) command clears the retained message which is stored for the specified topic.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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.

Delete Authentication Information Object

The Delete authentication information (MQCMD_DELETE_AUTH_INFO) command deletes the specified authentication information object.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

Delete Authority Record

The Delete Authority Record (MQCMD_DELETE_AUTH_REC) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

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 1333.

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.

z/OS Delete CF Structure

The Delete CF Structure (MQCMD_DELETE_CF_STRUC) command deletes an existing CF application structure definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

Delete Channel

The Delete Channel (MQCMD_DELETE_CHANNEL) command deletes the specified channel definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 IBM MQ Telemetry.

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.

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.

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 1333.

Error codes

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.

Delete Channel (MQTT)

The Delete Telemetry Channel (MQCMD_DELETE_CHANNEL) 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 1333.

***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.

Delete Channel Listener

The Delete Channel Listener (MQCMD_DELETE_LISTENER) command deletes an existing channel listener definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux systems	Windows	z/OS
	✓	✓	✓	

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.

Delete Communication Information Object

The Delete Communication Information Object (MQCMD_DELETE_COMM_INFO) command deletes the specified communication information object.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

Required parameter

***CominfoName* (MQCFST)**
The name of the communication information definition to be deleted (parameter identifier: MQCA_COMM_INFO_NAME).

Delete Namelist

The Delete Namelist (MQCMD_DELETE_NAMELIST) command deletes an existing namelist definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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 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.

Delete Policy

The Delete Policy (MQCMD_DELETE_PROT_POLICY) command deletes a security policy.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

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 1333.

***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.

Delete Process

The Delete Process (MQCMD_DELETE_PROCESS) command deletes an existing process definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

Delete Queue

The Delete Queue (MQCMD_DELETE_Q) command deletes a queue.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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.

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.

***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.

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 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.

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 1333.

***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.

Delete Service

The Delete Service (MQCMD_DELETE_SERVICE) command deletes an existing service definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux systems	Windows	z/OS
	✓	✓	✓	

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.

z/OS Delete Storage Class

The Delete Storage Class (MQCMD_DELETE_STG_CLASS) command deletes an existing storage class definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

Delete Subscription

The Delete Subscription (MQCMD_DELETE_SUBSCRIPTION) command deletes a subscription.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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

***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.

Delete Topic

The Delete Topic (MQCMD_DELETE_TOPIC) command deletes the specified administrative topic object.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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

***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.

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.

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 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.

Escape

The Escape (MQCMD_ESCAPE) command conveys any IBM MQ command (MQSC) to a remote queue manager.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	

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: MQCACF_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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_ESCAPE_TYPE_ERROR
Escape type not valid.

Escape (Response)

The response to the Escape (MQCMD_ESCAPE) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	

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.

z/OS Inquire Archive

The Inquire Archive (MQCMD_INQUIRE_ARCHIVE) command returns archive system parameters and information.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 Inquire Archive (Response)

The response to the Inquire Archive (MQCMD_INQUIRE_ARCHIVE) command consists of the response header followed by the *ParameterType* structure and the combination of attribute parameter structures determined by the value of *ParameterType*.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 us 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.

Inquire Authentication Information Object

The Inquire authentication information object (MQCMD_INQUIRE_AUTH_INFO) command inquires about the attributes of authentication information objects.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 1845 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 1852.

If you specify a string filter, you cannot also specify an integer filter using the **IntegerFilterCommand** parameter.

Inquire Authentication Information Object (Response)

The response of the Inquire authentication information (MQCMD_INQUIRE_AUTH_INFO) 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).

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 Security for more information.

V 8.0.0.3 **AuthenticationMethod (MQCFIN)**
Authentication methods for user passwords (parameter identifier: MQIA_AUTHENTICATION_METHOD). Possible values are:

MQAUTHENTICATE_OS
Use the traditional UNIX platforms password verification method.

MQAUTHENTICATE_PAM
Use the Pluggable Authentication Method to authenticate the user passwords.

You can set the PAM value only on UNIX 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.

V 8.0.0.2 **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 all the users belonging to that group.

MQLDAP_AUTHORMD_SEARCHUSER
A user entry in the LDAP repository contains an attribute listing all the groups to which the specified user belongs.

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

V 8.0.0.2 **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).

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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. z/OS (This setting is not allowed on z/OS systems.)

V 8.0.0.2 **ClassGroup (MQCFST)**
The LDAP object class used for group records in the LDAP repository (parameter identifier MQCA_LDAP_GROUP_OBJECT_CLASS).

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

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.

V 8.0.0.2 **FindGroup (MQCFST)**
Name of the attribute used within an LDAP entry to determine group membership (parameter identifier MQCA_LDAP_FIND_GROUP_FIELD).

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

The maximum length of the string is MQ_LDAP_FIELD_LENGTH.

V 8.0.0.2 **GroupField (MQCFST)**
LDAP attribute that represents a simple name for the group (parameter identifier MQCA_LDAP_GROUP_ATTR_FIELD).

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2, and is not applicable to Windows platforms.

The maximum length of the string is MQ_LDAP_FIELD_LENGTH.

V 8.0.0.2 **GroupNesting (MQCFIN)**
Whether groups are members of other groups (parameter identifier MQIA_LDAP_USE_NESTED_GROUPS). 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.

This attribute is valid only for an **AuthInfoType** of *MQAIT_IDPW_LDAP* V 8.0.0.2 , and is not applicable to Windows platforms.

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 SSL/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)..

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.

Inquire Authentication Information Object Names

The Inquire authentication information names (MQCMD_INQUIRE_AUTH_INFO_NAMES) command asks for a list of authentication information names that match the generic authentication information name specified.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

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.

Inquire Authentication Information Object Names (Response)

The response to the inquire authentication information names (MQCMD_INQUIRE_AUTH_INFO_NAMES) command consists of the response header followed by a parameter structure giving zero or more names that match the specified authentication information name.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

z/OS Additionally, on z/OS only, a parameter structure, *QSGDispositions* (with the same number of entries as the *AuthInfoNames* structure), is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *AuthInfoNames* structure.

Always returned:

AuthInfoNames z/OS, *QSGDispositions*

Returned if requested:
None

Response data

***AuthInfoNames* (MQCFSL)**
List of authentication information object names (parameter identifier: MQCACF_AUTH_INFO_NAMES).

z/OS

***QSGDispositions* (MQCFIL)**
List of QSG 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.

Inquire Authority Records

The Inquire Authority Records (MQCMD_INQUIRE_AUTH_RECS) command retrieves authority records associated with a profile name.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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 1333.

***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.

Inquire Authority Records (Response)

The response to the Inquire Authority Records (MQCMD_INQUIRE_AUTH_RECS) command consists of the response header followed by the *QMgrName*, *Options*, *ProfileName*, and *ObjectType* structures and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

Inquire Authority Service

The Inquire Authority Service (MQCMD_INQUIRE_AUTH_SERVICE) command retrieves information about the level of function supported by installed authority managers.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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 1333.

***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.

Inquire Authority Service (Response)

The response to the Inquire Authority Service (MQCMD_INQUIRE_AUTH_SERVICE) command consists of the response header followed by the *ServiceComponent* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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:

MQUIDSUPP_YES
The authority service supports user IDs.

MQUIDSUPP_NO
The authority service does not support user IDs.

z/OS Inquire CF Structure

The Inquire CF Structure (MQCMD_INQUIRE_CF_STRUC) command returns information about the attributes of one or more CF application structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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_CFCONLOS
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 1845 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 1852 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

z/OS Inquire CF Structure (Response)

The response to the Inquire CF Structure (MQCMD_INQUIRE_CF_STRUC) command consists of the response header followed by the *CFStrucName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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_CFCONLOS).

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 CFCONLOS 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 Inquire CF Structure Names

The Inquire CF Structure Names (MQCMD_INQUIRE_CF_STRUC_NAMES) command inquires for a list of CF application structure names that match the generic CF structure name specified.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 Inquire CF Structure Names (Response)

The response to the Inquire CF Structure Names (MQCMD_INQUIRE_CF_STRUC_NAMES) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

Always returned:
CFStrucNames

Returned if requested:
None

Response data

***CFStrucNames* (MQCFSL)**
List of CF application structure names (parameter identifier: MQCACF_CF_STRUC_NAMES).

z/OS Inquire CF Structure Status

The Inquire CF Structure Status (MQCMD_INQUIRE_CF_STRUC_STATUS) command inquires about the status of a CF application structure.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 1845 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 1852 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

z/OS Inquire CF Structure Status (Response)

The response to the Inquire CF Structure Status (MQCMD_INQUIRE_CF_STRUC_STATUS) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

***CFStrucType* (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.

Inquire Channel

The Inquire Channel (MQCMD_INQUIRE_CHANNEL) command inquires about the attributes of IBM MQ channel definitions.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 - default value used if the parameter is not specified:

MQIACF_ALL
All attributes.

or a combination of the parameters in the following table:

V 8.0.0.4

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	V 8.0.0.4 AMQP
MQCA_ALTERATION_DATE Date on which the definition was last altered	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQCA_ALTERATION_TIME Time at which the definition was last altered	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQCA_CERT_LABEL Certificate label	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQCA_CLUSTER_NAME Name of local queue manager							✓	✓	
MQCA_CLUSTER_NAMELIST Name of local queue manager							✓	✓	
MQCA_Q_MGR_NAME Name of local queue manager					✓				
MQCACH_CHANNEL_NAME Channel name. You cannot use this attribute as a filter keyword.	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQCACH_CONNECTION_NAME Connection name	✓	✓		✓	✓		✓	✓	

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	V 8.0.0.4 AMQP
MQCACH_DESC Description	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQCACH_LOCAL_ADDRESS Local communications address for the channel	✓	✓		✓	✓		✓	✓	✓
MQCACH_MCA_NAME Message channel agent name	✓	✓		✓			✓		
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 name			✓	✓				✓	
MQCACH_MSG_EXIT_NAME Message exit name	✓	✓	✓	✓			✓	✓	
MQCACH_MSG_EXIT_USER_DATA Message exit user data	✓	✓	✓	✓			✓	✓	
MQCACH_PASSWORD Password	✓	✓		✓	✓		✓		
MQCACH_RCV_EXIT_NAME Receive exit name	✓	✓	✓	✓	✓	✓	✓	✓	
MQCACH_RCV_EXIT_USER_DATA Receive exit user data	✓	✓	✓	✓	✓	✓	✓	✓	

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	V 8.0.0.4 AMQP
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 SSL cipher spec	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQCACH_SSL_PEER_NAME SSL peer name	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQCACH_TP_NAME Transaction program name	✓	✓		✓	✓	✓	✓	✓	
V 8.0.0.4 **MQCACH_TP_ROOT** Topic root for AMQP channel									✓
MQCACH_USER_ID User identifier	✓	✓		✓	✓		✓		
MQCACH_XMIT_Q_NAME Transmission queue name	✓	✓							
MQIA_MONITORING_CHANNEL Online monitoring data collection	✓	✓	✓	✓		✓	✓	✓	

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	V 8.0.0.4 AMQP
MQIA_PROPERTY_CONTROL Property control attribute	✓	✓					✓	✓	
MQIA_STATISTICS_CHANNEL Online statistics collection	✓	✓	✓	✓			✓	✓	
MQIA_USE_DEAD_LETTER_Q Determines whether the dead-letter queue is used when messages cannot be delivered by channels.	✓	✓	✓	✓			✓	✓	
V 8.0.0.4 **MQIACH_AMQP_KEEP_ALIVE** AMQP channel keep alive interval									✓
MQIACH_BATCH_HB Value to use for batch heartbeating	✓	✓					✓	✓	
MQIACH_BATCH_INTERVAL Batch wait interval (seconds)	✓	✓					✓	✓	
MQIACH_BATCH_DATA_LIMIT Batch data limit (kilobytes)	✓	✓					✓	✓	
MQIACH_BATCH_SIZE Batch size	✓	✓	✓	✓			✓	✓	
MQIACH_CHANNEL_TYPE Channel type	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQIACH_CLIENT_CHANNEL_WEIGHT Client Channel Weight					✓				

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	V 8.0.0.4 AMQP
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_CONNECTION_AFFINITY Connection Affinity					✓				
MQIACH_DATA_CONVERSION Whether sender must convert application data	✓	✓					✓	✓	
MQIACH_DEF_RECONNECT Default reconnection option					✓				
MQIACH_DISC_INTERVAL Disconnection interval	✓	✓				✓	✓	✓	
MQIACH_HB_INTERVAL Heartbeat interval (seconds)	✓	✓	✓	✓	✓	✓	✓	✓	
MQIACH_HDR_COMPRESSION List of header data compression techniques supported by the channel	✓	✓	✓	✓	✓	✓	✓	✓	
MQIACH_KEEP_ALIVE_INTERVAL KeepAlive interval	✓	✓	✓	✓	✓	✓	✓	✓	

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	V 8.0.0.4 AMQP
MQIACH_LONG_RETRY Long retry count	✓	✓					✓	✓	
MQIACH_LONG_TIMER Long timer	✓	✓					✓	✓	
MQIACH_MAX_INSTANCES Maximum number of simultaneous instances of a server-connection channel that can be started.						✓			✓
MQIACH_MAX_INSTS_PER_ CLIENT Maximum number of simultaneous instances of a server-connection channel that can be started from a single client.						✓			
MQIACH_MAX_MSG_LENGTH Maximum message length	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQIACH_MCA_TYPE MCA type	✓	✓		✓			✓	✓	
MQIACH_MR_COUNT Message retry count			✓	✓				✓	
MQIACH_MSG_COMPRESSION List of message data compression techniques supported by the channel	✓	✓	✓	✓	✓	✓	✓	✓	
MQIACH_MR_INTERVAL Message retry interval (milliseconds)			✓	✓				✓	
MQIACH_NPM_SPEED Speed of nonpersistent messages	✓	✓	✓	✓			✓	✓	

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver	V 8.0.0.4 AMQP
V 8.0.0.4 **MQIACH_PORT** AMQP port number									✓
MQIACH_PUT_AUTHORITY Put authority			✓	✓		✓		✓	
MQIACH_RESET_REQUESTED Sequence number of outstanding request when a RESET CHANNEL command is used	✓	✓	✓	✓			✓	✓	
MQIACH_SEQUENCE_NUMBER_ WRAP Sequence number wrap	✓	✓	✓	✓			✓	✓	
MQIACH_SHARING_CONVERSATIONS Value of Sharing Conversations						✓			
MQIACH_SHORT_RETRY Short retry count	✓	✓					✓	✓	
MQIACH_SHORT_TIMER Short timer	✓	✓					✓	✓	
MQIACH_SSL_CLIENT_AUTH SSL client authentication	✓	✓	✓	✓		✓		✓	✓
V 8.0.0.4 **MQIACH_USE_CLIENT_ID** Specify that the client ID is used for authorization checks for an AMQP channel									✓
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.

***Certificatelabel* (MQCFST)**
Certificate label (parameter identifier: MQCA_CERT_LABEL).

Specifies the certificate label in use.

The maximum length is MQ_CERT_LABEL_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 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.

V 8.0.0.4 **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.

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.

z/OS

***DefaultChannelDisposition* (MQCFIN)**
Default channel disposition (parameter identifier: MQIACH_CHANNEL_DISP).

This parameter is not allowed for client-connection (CLNTCONN) channels.

This parameter applies to z/OS only.

Specifies the disposition of the channels for which information is to be returned. If this parameter is not present (or if MQCHLD_ALL is specified), channels of all channel dispositions are eligible. The value can be any of the following values:

MQCHLD_ALL
Returns requested information for all eligible channels.

MQCHLD_PRIVATE
Returns requested information for PRIVATE channels.

MQCHLD_SHARED
Returns requested information for channels with channel disposition that is defined as either MQCHLD_SHARED or MQCHLD_FIXSHARED.

***DefReconnect* (MQCFIN)**
Client channel default reconnection option (parameter identifier: MQIACH_DEF_RECONNECT).

The default automatic client reconnection option. You can configure a 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.

***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 1845 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.

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 *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 1852 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 1333.

***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.

Inquire Channel (MQTT)

The Inquire Channel (MQCMD_INQUIRE_CHANNEL) 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
SSL 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
SSL cipher spec.

MQCACH_SSL_KEY_PASSPHRASE
SSL 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_SSL_CLIENT_AUTH
SSL 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 1333.

***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.

Inquire Channel (Response)

The response to the Inquire Channel (MQCMD_INQUIRE_CHANNEL) 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).

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

If a generic channel name was specified, one such message is generated for each channel found.

Always returned:

ChannelName , *ChannelType* , z/OS *DefaultChannelDisposition* , z/OS *QSGDisposition*

Returned if requested:

AlterationDate, *AlterationTime*, *BatchDataLimit*, *BatchHeartbeat*, *BatchInterval*, *BatchSize*, *Certificatelabel*, *ChannelDesc*, *ChannelMonitoring*, *ChannelStartTime*, *ChannelStartDate*, *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*, *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.

***ChannelStartDate* (MQCFST)**
The date that the channel started (parameter identifier: MQCACH_CHANNEL_START_DATE). The length is specified by the value MQ_DATE_LENGTH.

***ChannelStartTime* (MQCFST)**
The time that the channel started (parameter identifier: MQCACH_CHANNEL_START_TIME). The length is specified by the value MQ_TIME_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, 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.

***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 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).

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 MQCONNX, the client is not reconnected automatically.

MQRCN_YES
Unless overridden by MQCONNX, the client reconnects automatically.

MQRCN_Q_MGR
Unless overridden by MQCONNX, 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 MQCONNX 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.

In the following environments, 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: IBM i, Windows, UNIX and Linux. An MQCFSL structure is always used on z/OS.

***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.

In the following environments, if more than one message 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: IBM i, Windows, UNIX and Linux. An MQCFSL structure is always used on z/OS.

***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

Message properties	Result
The message contains a property with a prefix of **mcd.**, **jms.**, **usr.** or **mqext.**	All optional message properties (where the **Support** 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 it 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 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 **Support** 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 **Support** 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 *content='properties'* 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.

In the following environments, 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: IBM i, Windows, UNIX and Linux. An MQCFSL structure is always used on z/OS.

***ReceiveUserData* (MQCFST)**
Receive exit user data (parameter identifier: MQCACH_RCV_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, 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: IBM i, Windows, UNIX and Linux. An MQCFSL structure is always used on z/OS.

***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.

In the following environments, 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: IBM i, Windows, UNIX and Linux. An MQCFSL structure is always used on z/OS.

***SendUserData* (MQCFST)**
Send exit user data (parameter identifier: MQCACH_SEND_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, 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: IBM i, Windows, UNIX and Linux. An MQCFSL structure is always used on z/OS.

***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).

***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 SSL client.

***SSLPeerName* (MQCFST)**
Peer name (parameter identifier: MQCACH_SSL_PEER_NAME).

Note: An alternative way of restricting connections into channels by matching against the SSL or TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different SSL or 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 SSL 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.

Inquire Channel Authentication Records

The Inquire Channel Authentication Records (MQCMD_INQUIRE_CHLAUTH_RECS) command retrieves the allowed partner details and mappings to MCAUSER for a channel or set of channels.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	2CR

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 1840 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 z/OS (on platforms other than z/OS), and you have not set the **Connauth** field on the queue manager, or the value of **CheckClient** is None, the connection fails. You receive message AMQ9793 z/OS on platforms other than z/OS, and message CSQX793E on z/OS.

***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 MQCONNX 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 1845 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 wild cards. 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 1852 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 SSL 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

Inquire Channel Authentication Records (Response)

The response to the Inquire Channel Authentication Records (MQCMD_INQUIRE_CHLAUTH_RECS) command consists of the response header followed by the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	2CR

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 SSL 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 SSL 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 Windows, all members of the mqm group, the Administrators group and SYSTEM.
- On UNIX 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 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.

z/OS Inquire Channel Initiator

The Inquire Channel Initiator (MQCMD_INQUIRE_CHANNEL_INIT) command returns information about the channel initiator.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 Inquire Channel Initiator (Response)

The response to the Inquire Channel Initiator (MQCMD_INQUIRE_CHANNEL_INIT) 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).

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 SSL server subtasks (parameter identifier: MQIACH_SSLTASKS_MAX).

***SSLTasksStarted* (MQCFIN)**
The number of active SSL 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.

Inquire Channel Listener

The Inquire Channel Listener (MQCMD_INQUIRE_LISTENER) command inquires about the attributes of existing IBM MQ listeners.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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 1845 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 1852 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).

Inquire Channel Listener (Response)

The response to the Inquire Channel Listener (MQCMD_INQUIRE_LISTENER) command consists of the response header followed by the *ListenerName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

Inquire Channel Listener Status

The Inquire Channel Listener Status (MQCMD_INQUIRE_LISTENER_STATUS) command inquires about the status of one or more IBM MQ listener instances.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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 1845 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 "MQCFSF - PCF string filter parameter" on page 1852 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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_LSTR_STATUS_NOT_FOUND
Listener status not found.

Inquire Channel Listener Status (Response)

The response to the Inquire Channel Listener Status (MQCMD_INQUIRE_LISTENER_STATUS) command consists of the response header followed by the *ListenerName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

Inquire Channel Names

The Inquire Channel Names (MQCMD_INQUIRE_CHANNEL_NAMES) command inquires a list of IBM MQ channel names that match the generic channel name, and the optional channel type specified.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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.

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 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 1333.

***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.

Inquire Channel Names (Response)

The response to the Inquire Channel Names (MQCMD_INQUIRE_CHANNEL_NAMES) command consists of one response per client connection channel (except for SYSTEM.DEF.CLNTCONN), and a final message with all the remaining channels.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 QSG 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.

Inquire Channel Status

The Inquire Channel Status (MQCMD_INQUIRE_CHANNEL_STATUS) command inquires about the status of one or more channel instances.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 four options:

- **XmitQname** (MQCACH_XMIT_Q_NAME)
- **ConnectionName** (MQCACH_CONNECTION_NAME)
- z/OS **ChannelDisposition** (MQIACH_CHANNEL_DISP)
- **ChannelInstanceType** (MQIACH_CHANNEL_INSTANCE_TYPE)

distributed 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.

distributed Without this authority this command runs without error, but a value of zero is output for the **MsgsAvailable** parameter of the "Inquire Channel Status (Response)" on page 1578 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

z/OS ***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 88 on page 1565, Table 89 on page 1566, and Table 90 on page 1566.

Table 88. ChannelDisposition and CommandScope for Inquire Channel Status, Current

ChannelDisposition	***CommandScope* blank or local queue manager**	***CommandScope(qmgr-name)***	***CommandScope(*)***
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

Table 88. ChannelDisposition and CommandScope for Inquire Channel Status, Current (continued)			
ChannelDisposition	***CommandScope* blank or local queue manager**	***CommandScope(qmgr-name)***	***CommandScope(*)***
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 89. ChannelDisposition and CommandScope for Inquire Channel Status, Short			
ChannelDisposition	***CommandScope* blank or local queue manager**	***CommandScope(qmgr-name)***	***CommandScope(*)***
MQCHLD_PRIVATE	*ChannelStatus* and short status for current private channels on the local queue manager	*ChannelStatus* and short status for current private channels on the named queue manager	*ChannelStatus* and short status for current private channels on all active queue managers
MQCHLD_SHARED	*ChannelStatus* and short status for current shared channels on all active queue managers in the queue-sharing group	Not permitted	Not permitted
MQCHLD_ALL	*ChannelStatus* and short status for current private channels on the local queue manager and current shared channels in the queue-sharing group("1" on page 1566)	*ChannelStatus* and short status for current private channels on the named queue manager	*ChannelStatus* and short status for current private, and shared, channels on all active queue managers in the queue-sharing group("1" on page 1566)

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.

Table 90. ChannelDisposition and CommandScope for Inquire Channel Status, Saved			
ChannelDisposition	***CommandScope* blank or local queue manager**	***CommandScope(qmgr-name)***	***CommandScope(*)***
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_SHORT_PEER_NAME
SSL 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.

V 8.0.0.4 **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.

This parameter does not apply to z/OS.

MQIACH_SHORT_RETRIES_LEFT
Number of short retry attempts remaining.

MQIACH_SSL_KEY_RESETS
Number of successful SSL key resets.

MQIACH_SSL_RESET_DATE
Date of previous successful SSL secret key reset.

MQIACH_SSL_RESET_TIME
Time of previous successful SSL 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

- z/OS 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 1845 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 1852 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 1333.

***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.

Inquire Channel Status (AMQP)

The Inquire Channel Status (MQCMD_INQUIRE_CHANNEL_STATUS) (AMQP) 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 may 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

***ClientIdentifer* (MQCFST)**
The ClientId 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 1333.

***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.

Inquire Channel Status (MQTT)

The Inquire Channel Status (MQCMD_INQUIRE_CHANNEL_STATUS) (MQTT) 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 IBM 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 IBM 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 IBM MQ Telemetry command is truncated by the IBM 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

***ClientIdentifer* (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).

KAINT
The keep alive interval of the client (parameter identifier: MQIACH_KEEP_ALIVE_INTERVAL).

MCANAME
Message channel agent name (parameter identifier: MQCACH_MCA_USER_ID).

MSGSNT
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 1333.

***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.

Inquire Channel Status (Response)

The response to the Inquire Channel Status (MQCMD_INQUIRE_CHANNEL_STATUS) command consists of the response header followed by several structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

These structures are

- The *ChannelName* structure,
- z/OS 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 *RemoteApplTag* 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, if the value for any of these parameters exceeds 999999999, it is returned as 999999999:

- *Batches*
- *BuffersReceived*
- *BuffersSent*
- *BytesReceived*
- *BytesSent*
- *CompressionTime*
- *CurrentMsgs*
- *ExitTime*
- *Msgs*
- *NetTime*

- *SSLKeyResets*
- *XQTime*

Always returned:

z/OS *ChannelDisposition* , *ChannelInstanceType* , *ChannelName* , *ChannelStatus* , *ChannelType* , *ConnectionName* , *RemoteApplTag* , *RemoteQMgrName* , *StopRequested* , *SubState* , *XmitQName*

Returned if requested:

Batches, *BatchSize*, *BatchSizeIndicator*, *BuffersReceived*, *BuffersSent*, *BytesReceived*, *BytesSent*, *ChannelMonitoring*, *ChannelStartDate*, *ChannelStartTime*, *ClientIdentifier*, *CompressionRate*, *CompressionTime*, *CurrentLUWID*, *CurrentMsgs*, *CurrentSequenceNumber*, *CurrentSharingConversations*, *ExitTime*, *HeaderCompression*, *HeartbeatInterval*, *InDoubtInbound*, *InDoubtStatus*, *InDoubtOutbound*, *KeepAliveInterval*, *LastLUWID*, *LastMsgDate*, *LastMsgTime*, *LastSequenceNumber*, *LocalAddress*, *LongRetriesLeft*, *MaxMsgLength*, *MaxSharingConversations*, *MCAJobName*, *MCAStatus*, *MCAUserIdentifier*, *MessageCompression*, *Msgs*, *MsgsAvailable*, *MsgsReceived*, *MsgsSent*, *NetTime*, *NonPersistentMsgSpeed*, *PendingOutbound*, *QMgrName*, *ResponseType*, *RemoteVersion*, *RemoteProduct*, *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).

BytesSent **(MQCFIN)**
Number of bytes sent (parameter identifier: MQIACH_BYTES_SENT).

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 switched off.

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.

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.

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).

The maximum length of the string is MQ_SHORT_CONN_NAME_LENGTH.

***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-quiesce
- 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 header 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:

Table 91. Product Identifier values	
Product Identifier	**Description**
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

MQSECPROT_SSLV30
SSL version 3.0

This protocol is deprecated. See Deprecated CipherSpecs

MQSECPROT_TLSV10
TLS version 1.0

MQSECPROT_TLSV12
TLS version 1.2

This parameter is not available on z/OS.

***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 SSL 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)**
SSL secret key resets (parameter identifier: MQIACH_SSL_KEY_RESETS).

The number of successful SSL secret key resets that have occurred for this channel instance since the channel started. If SSL secret key negotiation is enabled, the count is incremented whenever a secret key reset is performed.

***SSLKeyResetTime* (MQCFST)**
Time of the previous successful SSL 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. This limit might mean that exceptionally long 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
SSL 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)**
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.

Inquire Channel Status (Response) (MQTT)

The response to the Inquire Channel Status (MQCMD_INQUIRE_CHANNEL_STATUS) 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: MQIACF_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)

Inquire Cluster Queue Manager

The Inquire Cluster Queue Manager (MQCMD_INQUIRE_CLUSTER_Q_MGR) command inquires about the attributes of IBM MQ queue managers in a cluster.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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
SSL cipher spec.

MQIACH_SSL_CLIENT_AUTH
SSL client authentication.

MQCACH_SSL_PEER_NAME
SSL 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 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.

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 1845 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 1852 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.

Inquire Cluster Queue Manager (Response)

The response to the Inquire Cluster Queue Manager (MQCMD_INQUIRE_CLUSTER_Q_MGR) command consists of three parts. The response header is followed by the *QMgrName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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* , z/OS *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.

This parameter is returned if the channel is a cluster-sender channel (CLUSSDR) only.

***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).

z/OS

***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.

In the following environments 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. The environments are: AIX, HP-UX, IBM i, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

***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.

In the following environments, 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. The environments are: AIX, HP-UX, IBM i, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

***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.

In the following environments, 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. The environments are: AIX, HP-UX, IBM i, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

***ReceiveUserData* (MQCFST)**
Receive exit user data (parameter identifier: MQCACH_RCV_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, more than one receive exit user data string can be defined for the channel. If more than one string is defined, the list of strings is returned in an MQCFSL structure instead of an (MQCFST) structure. The environments are: AIX, HP-UX, IBM i, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

***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.

In the following environments, 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. The environments are: AIX, HP-UX, IBM i, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

***SendUserData* (MQCFST)**
Send exit user data (parameter identifier: MQCACH_SEND_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, more than one send exit user data string can be defined for the channel. If more than one string is defined, the list of strings is returned in an MQCFSL structure instead of an (MQCFST) structure. The environments are: AIX, HP-UX, IBM i, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

***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 SSL 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 SSL 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

Inquire Communication Information Object

The Inquire Communication Information Object (MQCMD_INQUIRE_COMM_INFO) command inquires about the attributes of existing IBM MQ communication information objects.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

Required parameters:
CominfoName

Optional parameters:
CominfoAttrs, *IntegerFilterCommand*, *StringFilterCommand*

Required parameters

***CominfoName* (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

***CominfoAttrs* (MQCFIL)**
Cominfo 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
Cominfo 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 *CominfoAttrs* 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 1845 for information about using this filter condition.

If you specify an integer filter for *CominfoType* (MQIA_COMM_INFO_TYPE), you cannot also specify the *CominfoType* 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 *CominfoAttrs* 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 1852 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Communication Information Object (Response)

The response to the Inquire Communication Information Object (MQCMD_INQUIRE_COMM_INFO) command consists of the response header followed by the CominfoName structure, and the requested combination of attribute parameter structures (where applicable).

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

If a generic communication information name was specified, one such message is generated for each object found.

Always returned:
CominfoName

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 trasmitted 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.

***CominfoName* (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.

Inquire Connection

The Inquire connection (MQCMD_INQUIRE_CONNECTION) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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 1840 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.

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_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_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 1845 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 1852 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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_CONNECTION_ID_ERROR
Connection identifier not valid.

Inquire Connection (Response)

The response to the Inquire Connection (MQCMD_INQUIRE_CONNECTION) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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* , *ConnectionName* , *ConnectionOptions* , 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.

***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).

***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 *ApplType* 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).

On z/OS platforms, this parameter is returned as an 8-byte RBA. On platforms other than z/OS, this parameter is an 8-byte transaction identifier.

The maximum length of the string is MQ_UOW_ID_LENGTH.

z/OS

***QSGDispositon* (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.

Inquire Entity Authority

The Inquire Entity Authority (MQCMD_INQUIRE_ENTITY_AUTH) command inquires about authorizations of an entity to a specified object.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

  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 1333.

***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.

Inquire Entity Authority (Response)

Each response to the Inquire Entity Authority (MQCMD_INQUIRE_AUTH_RECS) command consists of the response header followed by the *QMgrName*, *Options*, and *ObjectName* structures and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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 Inquire Group

The Inquire Group (MQCMD_INQUIRE_QSG) command inquires about the queue-sharing group to which the queue manager is connected.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 Inquire Group (Response)

The response to the Inquire Group (MQCMD_INQUIRE_QSG) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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_700
Level 700 of system control commands.

MQCMDL_LEVEL_701
Level 701 of system control commands.

MQCMDL_LEVEL_710
Level 710 of system control commands.

MQCMDL_LEVEL_800
Level 800 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.

z/OS Inquire Log

The Inquire Log (MQCMD_INQUIRE_LOG) command returns log system parameters and information.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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_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*

Returned if *ParameterType* is MQSYSP_TYPE_LOG_COPY (one message is returned for each log copy):
DataSetName ,*LogCopyNumber* ,*LogUsed* ,*ZHyperWrite*

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)
Specifies whether the zHyperWrite feature is enabled (parameter identifier: MQIACF_SYSP_ZHYPERWRITE).

The value can be one of the following values:

MQSYSP_YES
zHyperWrite is enabled.

MQSYSP_NO
zHyperWrite is not enabled.

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.

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).

ZHyperWrite (MQCFIN)
Specifies whether the zHyperWrite feature is enabled (parameter identifier: MQIACF_SYSP_ZHYPERWRITE).

The value can be any of the following values:

MQSYSP_YES
zHyperWrite is enabled.

MQSYSP_NO
zHyperWrite is not enabled.

Inquire Namelist

The Inquire Namelist (MQCMD_INQUIRE_NAMELIST) command inquires about the attributes of existing IBM MQ namelists.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

Required parameters:
NamelistName

Optional parameters:
z/OS *CommandScope* ,*IntegerFilterCommand* ,*NamelistAttrs* , z/OS *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

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.

***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 1845 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 SSL, 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 1852 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Namelist (Response)

The response to the Inquire Namelist (MQCMD_INQUIRE_NAMELIST) command consists of the response header followed by the *NamelistName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

If a generic namelist name was specified, one such message is generated for each namelist found.

Always returned:

NamelistName , z/OS *QSGDisposition*

Returned if requested:

AlterationDate ,*AlterationTime* ,*NameCount* ,*NamelistDesc* , z/OS
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.

z/OS

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 SSL, 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.

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 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.

Inquire Namelist Names

The Inquire Namelist Names (MQCMD_INQUIRE_NAMELIST_NAMES) command inquires for a list of namelist names that match the generic namelist name specified.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

Inquire Namelist Names (Response)

The response to the Inquire Namelist Names (MQCMD_INQUIRE_NAMELIST_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified namelist name.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 QSG 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.

Inquire Policy

The Inquire Policy (MQCMD_INQUIRE_PROT_POLICY) command inquires about the policy, or policies, set on a queue.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

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

Inquire Policy (Response)

The response to the Inquire Policy (MQCMD_INQUIRE_PROT_POLICY) command consists of the response header followed by the *PolicyName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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* ,*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).

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.

Inquire Process

The Inquire Process (MQCMD_INQUIRE_PROCESS) command inquires about the attributes of existing IBM MQ processes.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

***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 1845 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 1852 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Process (Response)

The response to the Inquire Process (MQCMD_INQUIRE_PROCESS) command consists of the response header followed by the *ProcessName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux systems	Windows	z/OS
✓	✓	✓	✓	✓

If a generic process name was specified, one such message is generated for each process found.

Always returned:

ProcessName , z/OS *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.

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.

***UserData* (MQCFST)**
User data (parameter identifier: MQCA_USER_DATA).

The maximum length of the string is MQ_PROCESS_USER_DATA_LENGTH.

Inquire Process Names

The Inquire Process Names (MQCMD_INQUIRE_PROCESS_NAMES) command inquires for a list of process names that match the generic process name specified.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

Inquire Process Names (Response)

The response to the Inquire Process Names (MQCMD_INQUIRE_PROCESS_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified process name.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

Additionally, on z/OS only, a parameter structure, *QSGDispositions* (with the same number of entries as the *ProcessNames* structure) is returned. 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 QSG 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.

Inquire Pub/Sub Status

The Inquire Pub/Sub Status (MQCMD_INQUIRE_PUBSUB_STATUS) command inquires about the status of publish/subscribe connections.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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.

Inquire Pub/Sub Status (Response)

The response to the Inquire publish/subscribe Status (MQCMD_INQUIRE_PUBSUB_STATUS) command consists of the response header followed by the attribute structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	2CR

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).

Inquire Queue

Use the Inquire Queue command MQCMD_INQUIRE_Q to query the attributes of IBM MQ queues.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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 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.

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 1845 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:

Table 92. Inquire Queue command, queue attributes

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
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	✓	✓			

Table 92. Inquire Queue command, queue attributes (continued)					
	Local queue	**Model queue**	**Alias queue**	**Remote queue**	**Cluster queue**
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	✓	✓			
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	✓	✓	✓	✓	✓

Table 92. Inquire Queue command, queue attributes (continued)					
	Local queue	**Model queue**	**Alias queue**	**Remote queue**	**Cluster queue**
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				✓	
z/OS z/OS MQCA_STORAGE_CLASS Storage class. MQCA_STORAGE_CLASS is valid on z/OS only	✓	✓			
MQCA_TPIPE_NAME The **TPIPE** name used for communication with OTMA using the IBM MQ IMS bridge	✓				
MQCA_TRIGGER_DATA Trigger data	✓	✓			
MQCA_XMIT_Q_NAME Transmission queue name				✓	
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	✓		✓	✓	✓

Table 92. Inquire Queue command, queue attributes (continued)					
	Local queue	**Model queue**	**Alias queue**	**Remote queue**	**Cluster queue**
`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	✓	✓			
`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	✓	✓			

Table 92. Inquire Queue command, queue attributes (continued)					
	Local queue	**Model queue**	**Alias queue**	**Remote queue**	**Cluster queue**
MQIA_MAX_Q_DEPTH Maximum number of messages allowed on queue	✓	✓			
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	✓				
z/OS z/OS 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.	✓	✓			

Table 92. Inquire Queue command, queue attributes (continued)					
	Local queue	**Model queue**	**Alias queue**	**Remote queue**	**Cluster queue**
`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	✓		✓	✓	
`MQIA_SHAREABILITY` Whether queue can be shared	✓	✓			
`MQIA_STATISTICS_Q` Statistics data collection. `MQIA_STATISTICS_Q` is valid only on AIX, HP-UX, Linux, IBM i, Solaris, and Windows.	✓	✓			
`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	✓	✓			

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. 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 *QAttrs* 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 platforms other than z/OS, if this parameter is present, it must occur immediately after the *QName* parameter.

z/OS ***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 1852 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 1333.

Reason **(MQLONG)**
The value can be any of the following values:

MQRCCF_Q_TYPE_ERROR
Queue type not valid.

Inquire Queue (Response)

The response to the Inquire Queue command MQCMD_INQUIRE_Q 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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* ,*CFStructure* ,*ClusterChannelName* ,*ClusterDate* ,*ClusterName* ,

ClusterNamelist ,*ClusterQType* ,*ClusterTime* ,*CLWLQueuePriority* ,
CLWLQueueRank ,*CLWLUseQ* ,*CreationDate* ,*CreationTime* ,*CurrentQDepth* ,
Custom ,*DefaultPutResponse* ,*DefBind* ,*DefinitionType* ,*DefInputOpenOption* ,
DefPersistence ,*DefPriority* ,*DefReadAhead* ,*DistLists* ,*HardenGetBackout* ,
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* ,
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.

***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 and 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 in not enabled for this queue. Messages are not sent ahead to the client regardless of whether read ahead is requested by the client application.

***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 in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, and Linux.

***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.

***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)**
Putt 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.

***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 valid only on IBM i, UNIX systems, and Windows.

***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.

***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.

Inquire Queue Manager

The Inquire Queue Manager (**MQCMD_INQUIRE_Q_MGR**) command inquires about the attributes of a queue manager.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 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.

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_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.

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.

MQCA_IGQ_USER_ID
Intra-group queuing user identifier. This parameter is valid on z/OS only.

MQCA_LU_GROUP_NAME
Generic LU name for the LU 6.2 listener. MQCA_LU_GROUP_NAME is valid on z/OS only.

MQCA_LU_NAME
LU name to use for outbound LU 6.2 transmissions. MQCA_LU_NAME is valid on z/OS only.

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.

MQCA_QSG_CERT_LABEL
Queue sharing group certificate label. This parameter attribute is valid on z/OS only.

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
SSL certificate revocation location namelist.

MQCA_SSL_CRYPTO_HARDWARE
Parameters to configure the SSL cryptographic hardware. This parameter is supported on UNIX, Linux, and Windows platforms only.

MQCA_SSL_KEY_REPOSITORY
Location and name of the SSL key repository.

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

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, HP-UX, IBM i, Linux, Solaris, and Windows.

MQIA_ACCOUNTING_INTERVAL
Intermediate accounting data collection interval. MQIA_ACCOUNTING_INTERVAL is valid only on AIX, HP-UX, IBM i, Linux, Solaris, and Windows.

MQIA_ACCOUNTING_MQI
Specifies whether accounting information is to be collected for MQI data.
MQIA_ACCOUNTING_MQI is valid only on AIX, HP-UX, IBM i, Linux, Solaris, and Windows.

MQIA_ACCOUNTING_Q
Accounting data collection for queues.

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.

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.

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.

MQIA_AUTHORITY_EVENT
Control attribute for authority events.

MQIA_BRIDGE_EVENT
Control attribute for IMS bridge events. MQIA_BRIDGE_EVENT is valid only on z/OS.

MQIA_CERT_VAL_POLICY
Specifies which SSL/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 on only UNIX, Linux, and Windows. For more information, see Certificate validation policies in IBM MQ.

MQIA_CHANNEL_AUTO_DEF
Control attribute for automatic channel definition. MQIA_CHANNEL_AUTO_DEF is not valid on 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.

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.

MQIA_CHINIT_DISPATCHERS
Number of dispatchers to use for the channel initiator. MQIA_CHINIT_DISPATCHERS is valid on z/OS only.

MQIA_CHINIT_SERVICE_PARM
Reserved for use by IBM. MQIA_CHINIT_SERVICE_PARM is valid only on 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.

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.

MQIA_DIST_LISTS
Distribution list support. This parameter is not valid on 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.

MQIA_EXPIRY_INTERVAL
Expiry interval. This parameter is valid on z/OS only.

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.

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.

MQIA_INTRA_GROUP_queuing
Intra-group queuing support. This parameter is valid on z/OS only.

MQIA_IP_ADDRESS_VERSION
IP address version selector.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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 on z/OS only.

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 on z/OS only.

MQIA_SSL_EVENT
Control attribute for SSL events.

MQIA_SSL_FIPS_REQUIRED
Specifies whether only FIPS-certified algorithms are to be used if cryptography is executed in IBM MQ rather then in the cryptographic hardware itself.

MQIA_SSL_RESET_COUNT
SSL key reset count.

MQIA_SSL_TASKS
SSL 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.

MQIA_STATISTICS_INTERVAL
Statistics data collection interval. MQIA_STATISTICS_INTERVAL is valid only on AIX, HP-UX, IBM i, Linux, Solaris, and Windows.

MQIA_STATISTICS_MQI
Specifies whether statistics monitoring data is to be collected for the queue manager. MQIA_STATISTICS_MQI is valid only on AIX, HP-UX, IBM i, Linux, Solaris, and Windows.

MQIA_STATISTICS_Q
Specifies whether statistics monitoring data is to be collected for queues. MQIA_STATISTICS_Q is valid only on AIX, HP-UX, IBM i, Linux, Solaris, 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 SSL and 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 on z/OS only.

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 on z/OS only.

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 on z/OS only.

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_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
- 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_COMMAND_INPUT_Q_NAME
- MQCA_CUSTOM
- MQCA_DEAD_LETTER_Q_NAME
- MQCA_Q_MGR_NAME
- MQCA_QSG_NAME
- MQCA_VERSION
- MQIA_ACCOUNTING_CONN_OVERRIDE
- MQIA_ACCOUNTING_INTERVAL
- MQIA_ACCOUNTING_Q
- MQIA_ACTIVITY_CONN_OVERRIDE
- MQIA_ACTIVITY_RECORDING
- MQIA_ACTIVITY_TRACE
- MQCA_ALTERATION_DATE
- MQCA_ALTERATION_TIME
- MQIA_CMD_SERVER_CONTROL
- MQIA_CODED_CHAR_SET_ID
- MQIA_COMMAND_LEVEL
- MQIA_CPI_LEVEL
- MQIA_DIST_LISTS
- MQIA_EXPIRY_INTERVAL
- MQIA_MAX_HANDLES
- MQIA_MAX_MSG_LENGTH
- MQIA_MAX_PRIORITY
- MQIA_MAX_PROPERTIES_LENGTH
- MQIA_MAX_UNCOMMITTED_MSGS
- MQIA_MONITORING_Q
- 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

Inquire Queue Manager (Response)

The response to the Inquire Queue Manager (MQCMD_INQUIRE_Q_MGR) command consists of the response header followed by the *QMgrName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

Always returned:
QMgrName

Returned if requested:
AccountingConnOverride ,*AccountingInterval* ,*ActivityConnOverride* , *ActivityRecording* ,*ActivityTrace* ,*AdoptNewMCACheck* ,*AdoptNewMCAType* , *AlterationDate* ,*AlterationTime* ,*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* ,*InhibitEvent* , *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* ,*SharedQQmgrName* , *Splcap* ,*SSLCRLNamelist* ,*SSLCryptoHardware* ,*SSLEvent* ,*SSLFIPSRequired* , *SSLKeyRepository* ,*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 MQCONNX API call.

This parameter applies only to AIX, HP-UX, IBM i, Linux, Solaris, 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, HP-UX, IBM i, Linux, Solaris, 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 MQCONNX 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, Unix systems, 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.

***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 ACTVCONO 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, Unix systems, 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.

***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.

***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 for this queue manager to use. The label identifies which personal certificate in the key repository has been selected.

The maximum length of the string is MQ_CERT_LABEL_LENGTH.

***CertificateValPolicy* (MQCFIN)**
Specifies which SSL/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 only valid on UNIX, 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.

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 switched off.

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.

On z/OS, 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 switched off.

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.

On z/OS, 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.

***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_600
Level 600 of system control commands.

This value is returned by the following versions:

- IBM WebSphere MQ for AIX Version 6.0
- IBM WebSphere MQ for HP-UX Version 6.0
- IBM WebSphere MQ for iSeries Version 6.0
- IBM WebSphere MQ for Linux Version 6.0
- IBM WebSphere MQ for Solaris Version 6.0
- IBM WebSphere MQ for Windows Version 6.0
- IBM WebSphere MQ for z/OS Version 6.0

MQCMDL_LEVEL_700
Level 700 of system control commands.

This value is returned by the following versions:

- IBM WebSphere MQ for AIX Version 7.0
- IBM WebSphere MQ for HP-UX Version 7.0
- IBM WebSphere MQ for IBM i Version 7.0
- IBM WebSphere MQ for Linux Version 7.0
- IBM WebSphere MQ for Solaris Version 7.0
- IBM WebSphere MQ for Windows Version 7.0

- IBM WebSphere MQ for z/OS Version 7.0

MQCMDL_LEVEL_701
Level 701 of system control commands.

This value is returned by the following versions:

- IBM WebSphere MQ for AIX Version 7.0.1
- IBM WebSphere MQ for HP-UX Version 7.0.1
- IBM WebSphere MQ for IBM i Version 7.0.1
- IBM WebSphere MQ for Linux Version 7.0.1
- IBM WebSphere MQ for Solaris Version 7.0.1
- IBM WebSphere MQ for Windows Version 7.0.1
- IBM WebSphere MQ for z/OS Version 7.0.1

MQCMDL_LEVEL_710
Level 710 of system control commands.

This value is returned by the following versions:

- IBM WebSphere MQ for AIX Version 7.1
- IBM WebSphere MQ for HP-UX Version 7.1
- IBM WebSphere MQ for IBM i Version 7.1
- IBM WebSphere MQ for Linux Version 7.1
- IBM WebSphere MQ for Solaris Version 7.1
- IBM WebSphere MQ for Windows Version 7.1
- IBM WebSphere MQ for z/OS Version 7.1

MQCMDL_LEVEL_750
Level 750 of system control commands.

This value is returned by the following versions:

- IBM WebSphere MQ for AIX Version 7.5
- IBM WebSphere MQ for HP-UX Version 7.5
- IBM WebSphere MQ for IBM i Version 7.5
- IBM WebSphere MQ for Linux Version 7.5
- IBM WebSphere MQ for Solaris Version 7.5
- IBM WebSphere MQ for Windows Version 7.5

MQCMDL_LEVEL_800
Level 800 of system control commands.

This value is returned by the following versions:

- IBM WebSphere MQ for AIX Version 8.0
- IBM WebSphere MQ for HP-UX Version 8.0
- IBM WebSphere MQ for IBM i Version 8.0
- IBM WebSphere MQ for Linux Version 8.0
- IBM WebSphere MQ for Solaris Version 8.0
- IBM WebSphere MQ for Windows Version 8.0
- IBM WebSphere MQ for z/OS Version 8.0

MQCMDL_LEVEL_801
Level 801 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX Version 8.0.0.2
- IBM MQ for HP-UX Version 8.0.0.2
- IBM MQ for IBM i Version 8.0.0.2
- IBM MQ for Linux Version 8.0.0.2
- IBM MQ for Solaris Version 8.0.0.2

V 8.0.0.3 **MQCMDL_LEVEL_802**
Level 802 of system control commands.

This value is returned by the following versions:

- IBM MQ for AIX Version 8.0.0.3
- IBM MQ for HP-UX Version 8.0.0.3
- IBM MQ for IBM i Version 8.0.0.3
- IBM MQ for Linux Version 8.0.0.3
- IBM MQ for Solaris Version 8.0.0.3
- IBM MQ for Windows Version 8.0.0.3

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.

***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)**

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.

***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 and 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. This behavior is implicit in versions of IBM WebSphere MQ, earlier than Version 7.5. In earlier versions, the queue manager attribute DefClusterXmitQueueType was not present.

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. Refer to 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. Refer to 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 SSL and 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 QSG 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.

***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.

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, HP-UX, IBM i, Linux, Solaris, 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, HP-UX, IBM i, Linux, Solaris, 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, HP-UX, IBM i, Linux, Solaris, 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).

V 8.0.0.2 **MQPL_APPLIANCE**
IBM MQ Appliance

MQPL_NSK
HP Integrity NonStop Server.

MQPL_OS400
IBM i.

MQPL_UNIX
UNIX systems.

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

WebSphere Message Broker (now known as IBM Integration Bus) 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 for this queue sharing group to use. The label identifies which personal certificate in the key repository has been selected.

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.

***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 AIX, HP-UX, Linux, IBM i, Solaris, and Windows.

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.

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.

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.

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)**
If the IBM MQ 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 IBM MQ 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 IBM MQ AMS component is installed for the version of IBM MQ that the queue manager is running under.

MQCAP_NOT_SUPPORTED
If the IBM MQ AMS component is not installed.

***SSLCRLNamelist* (MQCFST)**
The SSL 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 LDAPCRL 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)**
Parameters to configure the SSL 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 supported on AIX, HP-UX, Solaris, Linux, and Windows only.

***SSLEvent* (MQCFIN)**
Controls whether SSL 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 Windows Linux UNIX and z/OS platforms.

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 SSL 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.

***SSLKeyResetCount* (MQCFIN)**
SSL key reset count (parameter identifier: MQIA_SSL_RESET_COUNT).

The number of unencrypted bytes that initiating SSL channel MCAs send or receive before renegotiating the secret key.

z/OS ***SSLTasks* (MQCFIN)**
Number of server subtasks used for processing SSL calls (parameter identifier: MQIA_SSL_TASKS).

The number of server subtasks used for processing SSL 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.

***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 AIX, HP-UX, Linux, IBM i, Solaris, and Windows.

***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 IBM 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 IBM i, Unix systems, and Windows.

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 UNIX, Linux and Windows

IBM i distributed MQCMD_INQUIRE_Q_MGR_STATUS (Inquire Queue Manager Status) on Distributed platforms

The Inquire Queue Manager Status (MQCMD_INQUIRE_Q_MGR_STATUS) PCF command inquires about the status of the local queue manager.

Optional parameters

QMStatusAttrs (MQCFIL)

Queue manager status attributes (parameter identifier: MQIACF_Q_MGR_STATUS_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_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.

MQCACF_LOG_PATH
Location of the recovery log extents.

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.

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.

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.

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.

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.

MQIACF_LDAP_CONNECTION_STATUS
Current status of the connection to the LDAP server.

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.

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_PERMIT_STANDBY
Whether a standby instance is permitted.

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.

IBM i distributed MQCMD_INQUIRE_Q_MGR_STATUS (Inquire Queue Manager Status) Response on Distributed platforms

The response to the Inquire Queue Manager Status (MQCMD_INQUIRE_Q_MGR_STATUS) PCF command consists of the response header followed by the *QMgrName* and *QMgrStatus* structures and the requested combination of attribute parameter structures.

Always returned:
QMgrName, *QMgrStatus*

Returned if requested:
ArchiveLog, *ArchiveLogSize*, *ChannelInitiatorStatus*, *CommandServerStatus*, *ConnectionCount*, *CurrentLog*, *InstallationDesc*, *InstallationName*, *InstallationPath*, *LDAPConnectionStatus*, *LogInUse*, *LogPath*, *LogUtilization*, *MediaRecoveryLog*, *MediaRecoveryLogSize*, *PermitStandby*, *RestartRecoveryLogSize*, *ReusableLogSize*, *StartDate*, *StartTime*

Response data

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).

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.

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.

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.

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.

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).

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.

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.

Inquire Queue Names

The Inquire Queue Names (MQCMD_INQUIRE_Q_NAMES) command inquires a list of queue names that match the generic queue name, and the optional queue type specified.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 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 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.

Inquire Queue Names (Response)

The response to the Inquire Queue Names (MQCMD_INQUIRE_Q_NAMES) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

z/OS

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 , z/OS *QSGDispositions* , *QTypes*

Returned if requested:
None

Response data

***QNames* (MQCFSL)**
List of queue names (parameter identifier: MQCACF_Q_NAMES).

z/OS

***QSGDispositions* (MQCFIL)**
List of QSG 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.

Inquire Queue Status

The Inquire Queue Status (MQCMD_INQUIRE_Q_STATUS) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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 1840 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.

z/OS

***CommandScope* (MQCFST)**
Command scope (parameter identifier: MQCACF_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 1845 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 if not specified is MQQSOT_ALL.

Filtering is not supported for this parameter.

z/OS

***QSGDispositon* (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 *QSGDispositon* 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_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 *ApplTag*. 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 1852 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 1333 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.

Inquire Queue Status (Response)

The response to the Inquire Queue Status (MQCMD_INQUIRE_Q_STATUS) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

Always returned:

QName , z/OS *QSGDisposition* , *StatusType*

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:
CurrentQDepth, *LastGetDate*, *LastGetTime*, *LastPutDate*, *LastPutTime*, *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* , *ProcessId* , z/OS *PSBName* , z/OS *PSTId* , *QMgrUOWId* , z/OS *TaskNumber* , *ThreadId* , z/OS *TransactionId* , *UOWIdentifier* , *UOWType* , *UserIdentifier*

Response data if StatusType is MQIACF_Q_STATUS

***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.

***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.L0G 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 on AIX, HP-UX, Linux, IBM i, Solaris, and Windows.

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 switched off.

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
On platforms other than z/OS, 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.

MQAT_BATCH
Application using a batch connection. MQAT_BATCH applies only to z/OS.

MQAT_RRS_BATCH
RRS-coordinated application using a batch connection. MQAT_RRS_BATCH applies only to z/OS.

MQAT_CICS
A CICS transaction. MQAT_CICS applies only to 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.

***Conname* (MQCFST)**
Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

The maximum length of the string is MQ_CONN_NAME_LENGTH.

z/OS

***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.

***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.

***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

MQUOWT_CICS
z/OS Valid only on z/OS.

MQUOWT_RRS
z/OS Valid only on z/OS.

MQUOWT_IMS
z/OS 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 Inquire Security

The Inquire Security (MQCMD_INQUIRE_SECURITY) command returns information about the current settings for the security parameters.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

***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 Inquire Security (Response)

The response to the Inquire Security (MQCMD_INQUIRE_SECURITY) command consists of the response header followed by the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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* attribute 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.

Inquire Service

The Inquire Service (MQCMD_INQUIRE_SERVICE) command inquires about the attributes of existing IBM MQ services.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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 1845 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 1852 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Service (Response)

The response to the Inquire Service (MQCMD_INQUIRE_SERVICE) command consists of the response header followed by the *ServiceName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

Inquire Service Status

The Inquire Service Status (MQCMD_INQUIRE_SERVICE_STATUS) command inquires about the status of one or more IBM MQ service instances.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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 1845 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 1852 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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_SERV_STATUS_NOT_FOUND
Service status not found.

Inquire Service Status (Response)

The response to the Inquire Service Status (MQCMD_INQUIRE_SERVICE_STATUS) command consists of the response header followed by the *ServiceName* structure and the requested combination of attribute parameter structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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: MQIACH_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 Inquire SMDS

The Inquire SMDS (MQCMD_INQUIRE_SMDS) command inquires about the attributes of shared message data sets for a CF application structure.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 Inquire SMDS (Response)

The response to the Inquire SMDS (MQCMD_INQUIRE_SMDS) command returns the attribute parameters of the shared message data set connection.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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

z/OS Inquire SMDS Connection

The response to the Inquire SMDS Connection (MQCMD_INQUIRE_SMDSCONN) command returns status and availability information about the connection between the queue manager and the shared message data sets for the specified *CFStrucName*.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

z/OS 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).

z/OS Inquire Storage Class

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

The Inquire Storage Class (MQCMD_INQUIRE_STG_CLASS) 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 1845 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 1852 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

z/OS Inquire Storage Class (Response)

The response to the Inquire Storage Class (MQCMD_INQUIRE_STG_CLASS) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

z/OS Inquire Storage Class Names

The Inquire Storage Class Names (MQCMD_INQUIRE_STG_CLASS_NAMES) command inquires a list of storage class names that match the generic storage class name specified.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

z/OS Inquire Storage Class Names (Response)

The response to the Inquire Storage Class Names (MQCMD_INQUIRE_STG_CLASS_NAMES) command consists of the response header followed by a parameter structure giving zero or more names that match the specified namelist name.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 QSG 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.

Inquire Subscription

The Inquire Subscription (MQCMD_INQUIRE_SUBSCRIPTION) command inquires about the attributes of a subscription.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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

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.

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 recreate the subscription by using **MQDOPT_DEFINED**.

Inquire Subscription (Response)

The response to the Inquire Subscription (MQCMD_INQUIRE_SUBSCRIPTION) command consists of the response header followed by the *SubId* and *SubName* structures, and the requested combination of attribute parameter structures (where applicable).

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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 *ApplIdentityData* 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.

Inquire Subscription Status

The Inquire Subscription Status (MQCMD_INQUIRE_SUB_STATUS) command inquires about the status of a subscription.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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

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.

***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.

Inquire Subscription Status (Response)

The response to the Inquire Subscription Status (MQCMD_INQUIRE_SUB_STATUS) command consists of the response header followed by the *SubId* and *SubName* structures, and the requested combination of attribute parameter structures (where applicable).

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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: MQIA_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.

z/OS Inquire System

The Inquire System (MQCMD_INQUIRE_SYSTEM) command returns general system parameters and information.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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_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):
CheckpointCount, *ClusterCacheType*, *CodedCharSetId*, *CommandUserId*, *ConnSwap*, *DB2BlobTasks*, *DB2Name*, *DB2Tasks*, *DSGName*, *Exclmsg*, *ExitInterval*, *ExitTasks*, *MULCCapture*, *OTMADruExit*, *OTMAGroup*, *OTMAInterval*, *OTMAMember*, *OTMSTpipePrefix*, *QIndexDefer*, *QSGName*, *RESLEVELAudit*, *RoutingCode*, *Service*, *SMFAccounting*, *SMFStatistics*, *SMFInterval*, *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.

ConnSwap (MQCFIN)
Specifies whether jobs that are issuing certain MQ API calls are swappable or non-swappable (parameter identifier: MQIACF_CONNECTION_SWAP).

This value can be either MQSYSP_YES or MQSYSP_NO.

Attention: From IBM MQ 9.0, this keyword has no effect.

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.

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.

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.

Inquire Topic

The Inquire Topic (MQCMD_INQUIRE_TOPIC) command inquires about the attributes of existing IBM MQ administrative topic objects

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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

***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.

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.

***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 1845 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. 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 1852 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.

Inquire Topic (Response)

The response to the Inquire Topic (MQCMD_INQUIRE_TOPIC) 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).

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

Always returned:

TopicName , *TopicType* , z/OS *QSGDisposition*

Returned if requested:

AlterationDate , *AlterationTime* , *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.

***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 and 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 cannote 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 behvaiour 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.

Inquire Topic Names

The Inquire Topic Names (MQCMD_INQUIRE_TOPIC_NAMES) command inquires a list of administrative topic names that match the generic topic name specified.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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

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 as MQQSGD_Q_MGR or MQQSGD_COPY. MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

Inquire Topic Names (Response)

The response to the Inquire Topic Names (MQCMD_INQUIRE_TOPIC_NAMES) command consists of the response header followed by a parameter structure giving zero or more names that match the specified administrative topic name.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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 QSG 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.

Inquire Topic Status

The Inquire Topic Status (MQCMD_INQUIRE_TOPIC_STATUS) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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: 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 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 "Inquire Topic Status (Response)" on page 1762. 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.

Inquire Topic Status (Response)

The response of the Inquire topic (MQCMD_INQUIRE_TOPIC_STATUS) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

Always returned:
TopicString

Returned if requested and StatusType is MQIACF_TOPIC_STATUS:
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)

***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.

z/OS Inquire Usage

The Inquire Usage (MQCMD_INQUIRE_USAGE) command inquires about the current state of a page set, or information about the log data sets.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

***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.

z/OS Inquire Usage (Response)

The response to the Inquire Usage (MQCMD_INQUIRE_USAGE) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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, *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*

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.

***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. For further information about suspended page sets, see message CSQP059E.

***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

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).

z/OS Move Queue

The Move Queue (MQCMD_MOVE_Q) command moves all the messages from one local queue to another.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

Ping Channel

The Ping Channel (MQCMD_PING_CHANNEL) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 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

***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 93 on page 1776

Table 93. ChannelDisposition and CommandScope for PING CHANNEL

ChannelDisposition	***CommandScope* blank or local-qmgr**	***CommandScope* qmgr-name**	***CommandScope(*)***
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

Table 93. ChannelDisposition and CommandScope for PING CHANNEL (continued)			
ChannelDisposition	***CommandScope* blank or local-qmgr**	***CommandScope* qmgr-name**	***CommandScope(*)***
MQCHLD_SHARED	Ping a shared channel on the most suitable queue manager in the group MQCHLD_SHARED might automatically generate a command using *CommandScope* 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	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 1333.

***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.

MQRCCF_CHANNEL_TYPE_ERROR
Channel type not valid.

MQRCCF_CONFIGURATION_ERROR
Configuration error.

MQRCCF_CONNECTION_CLOSED
Connection closed.

MQRCCF_CONNECTION_REFUSED
Connection refused.

MQRCCF_DATA_TOO_LARGE
Data too large.

MQRCCF_ENTRY_ERROR
Connection name not valid.

MQRCCF_HOST_NOT_AVAILABLE
Remote system not available.

MQRCCF_NO_COMMS_MANAGER
Communications manager not available.

MQRCCF_PING_DATA_COMPARE_ERROR
Ping Channel command failed.

MQRCCF_PING_DATA_COUNT_ERROR
Data count not valid.

MQRCCF_PING_ERROR
Ping error.

MQRCCF_RECEIVE_FAILED
Receive failed.

MQRCCF_RECEIVED_DATA_ERROR
Received data error.

MQRCCF_REMOTE_QM_TERMINATING
Remote queue manager terminating.

MQRCCF_REMOTE_QM_UNAVAILABLE
Remote queue manager not available.

MQRCCF_SEND_FAILED
Send failed.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

MQRCCF_TERMINATED_BY_SEC_EXIT
Channel terminated by security exit.

MQRCCF_UNKNOWN_REMOTE_CHANNEL
Remote channel not known.

MQRCCF_USER_EXIT_NOT_AVAILABLE
User exit not available.

Ping Queue Manager

The Ping Queue Manager (MQCMD_PING_Q_MGR) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux systems	Windows	z/OS
✓	✓	✓	✓	

Required parameters:
None

Optional parameters:
None

Purge Channel

The Purge Channel (MQCMD_PURGE_CHANNEL) command stops and purges an IBM MQ telemetry channel.

HP Integrity NonStop Server	IBM i	AIX and Linux	Windows	z/OS
		✓	✓	

This command can only be issued to an MQTT channel type.

Purging a telemetry channel disconnects all the MQTT clients connect to it, cleans up the state of the MQTT clients, and stops the telemetry 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.

***ChannelType* (MQCFIN)**
Channel type. This parameter must follow immediately after the **ChannelName** parameter z/OS on all platforms except z/OS, and the value must be MQTT.

Optional parameters

***ClientIdentifier* (MQCFST)**
Client identifier. The client identifier is a 23-byte string that identifies an IBM MQ Telemetry Transport 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 Recover CF Structure

The Recover CF Structure (MQCMD_RECOVER_CF_STRUC) command initiates recovery of CF application structures.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

Refresh Cluster

The Refresh Cluster (MQCMD_REFRESH_CLUSTER) command discards all locally held cluster information, including any auto-defined channels that are not in doubt, and forces the repository to be rebuilt.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

***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, it is 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

Refresh Queue Manager

Use the Refresh Queue Manager (MQCMD_REFRESH_Q_MGR) command to perform special operations on queue managers.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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: z/OS 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.

z/OS 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)

z/OS

***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:

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. V 8.0.0.4 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.

Refresh Security

The Refresh Security (MQCMD_REFRESH_SECURITY) command refreshes the list of authorizations held internally by the authorization service component.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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.

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 distributed platforms 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 SSL channels currently running, as follows:

- Sender, server, and cluster-sender channels using SSL are allowed to complete the current batch. In general, they then run the SSL handshake again with the refreshed view of the SSL key repository. However, you must manually restart a requester-server channel on which the server definition has no CONNAME parameter.
- All other channel types using SSL 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 SSL handshake uses the refreshed view of the contents of the SSL 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.

z/OS Reset CF Structure

The Reset coupling facility (CF) Structure (MQCMD_RESET_CF_STRUC) command modifies the status of a specific application structure.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

Reset Channel

The Reset Channel (MQCMD_RESET_CHANNEL) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 94 on page 1788

Table 94. ChannelDisposition and CommandScope for RESET CHANNEL

ChannelDisposition	***CommandScope* blank or local-qmgr**	***CommandScope* qmgr-name**
MQCHLD_PRIVATE	Reset private channel on the local queue manager	Reset private channel on the named queue manager
MQCHLD_SHARED	Reset a shared channel on all active queue managers. MQCHLD_SHARED might automatically generate a command using *CommandScope* 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

***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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

Reset Cluster

The Reset Cluster (MQCMD_RESET_CLUSTER) command forces a queue manager to leave a cluster.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

***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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_ACTION_VALUE_ERROR
Value not valid.

Reset Queue Manager

Use the Reset Queue Manager (MQCMD_RESET_Q_MGR) command as part of your backup and recovery procedures on AIX, HP-UX, Linux, Solaris, IBM i, and Windows.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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.

Required parameters

***Action* (MQCFIN)**
Action (parameter identifier: MQIACF_ACTION).

Specifies the action to take place.

The value can be any of the following values:

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 Compaq NSK, or 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 Compaq NSK, or z/OS.

MQACT_PUBSUB
Requests a publish/subscribe reset. This value requires that one of the optional parameters, ChildName or ParentName, is specified.

Optional parameters

***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.

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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRC_RESOURCE_PROBLEM
Insufficient system resources available.

Reset Queue Statistics

The Reset Queue Statistics (MQCMD_RESET_Q_STATS) 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).

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

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 1333.

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, it is necessary to enable queue manager performance events to use this command. For more details, see the PerformanceEvent property in the “Change Queue Manager” on page 1424 command.

Reset Queue Statistics (Response)

The response to the Reset Queue Statistics (MQCMD_RESET_Q_STATS) command consists of the response header followed by the *QName* structure and the attribute parameter structures shown in the following sections.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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 Reset SMDS

The Reset SMDS (MQCMD_RESET_SMDS) command modifies the availability or status information relating to one or more shared message data sets associated with a specific application structure

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

Resolve Channel

The Resolve Channel (MQCMD_RESOLVE_CHANNEL) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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.

***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 95 on page 1796

Table 95. ChannelDisposition and CommandScope for RESOLVE CHANNEL

ChannelDisposition	***CommandScope* blank or local-qmgr**	***CommandScope* qmgr-name**
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 *CommandScope* 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 1333.

***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 Resume Queue Manager

The Resume Queue Manager (MQCMD_RESUME_Q_MGR) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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:

MQQMFAC_DB2
Resumes normal activity with Db2.

MQQMFAC_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.

Resume Queue Manager Cluster

The Resume Queue Manager Cluster (MQCMD_RESUME_Q_MGR_CLUSTER) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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.

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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_CLUSTER_NAME_CONFLICT
Cluster name conflict.

z/OS Reverify Security

The Reverify Security (MQCMD_REVERIFY_SECURITY) to set a reverification flag for all specified users. The user is reverified the next time that security is checked for that user.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

z/OS Set Archive

Use the Set Archive (MQCMD_SET_ARCHIVE) to dynamically change certain archive system parameter values initially set by your system parameter module at queue manager startup.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

See the Administering IBM MQ for z/OS for information about 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 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.

***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 hhmmsst.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 hhmmsst.A nnnnnnn
```

Where *arcpfxi* is the data set name prefix specified by *ArchivePrefix1* or *ArchivePrefix2*. *arcpfxi* can have up to 17 characters.

Set Authority Record

The Set Authority Record (MQCMD_SET_AUTH_REC) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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 1333.

***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.

Set Channel Authentication Record

The Set Channel Authentication Record (MQCMD_SET_CHLAUTH_REC) command sets the allowed partner details and mappings to MCAUSER for a channel or set of channels.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

Syntax diagram

See the syntax diagram in the MQSC "SET CHLAUTH" on page 1273 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 SSL 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	✓	✓	

Parameter	MQACT_ADD or MQACT_REPLACE	MQACT_REMOVE	MQACT_REMOVEALL
CheckClient	✓	✓	
ClntUser	✓	✓	
MCAUser	✓		
QMName	✓	✓	
SSLCertIssuer	✓	✓	
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 **ClntUser**. If the specified type has no current configuration the command still succeeds.

***Address* (MQCFST)**

Attention: Host names can be specified in this parameter, only on queue managers that have IBM MQ 8.0 new functions enabled with OPMODE.

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_ADDESSMAP 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 SSL or TLS peer name, with different addresses. See "Generic IP addresses" on page 1280 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)**

Attention: This parameter is valid only on queue managers that have IBM MQ 8.0 new functions enabled with OPMODE.

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 MQCONNX 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 SSL 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)**

Attention: This parameter is valid only on queue managers that have IBM MQ 8.0 new functions enabled with OPMODE.

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 UNIX 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
- z/OS 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 1333.

***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

z/OS Set Log

Use the Set Log (MQCMD_SET_LOG) command to dynamically change certain log system parameter values initially set by your system parameter module at queue manager startup.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

Required parameters:
ParameterType

Optional parameters (if the value of *ParameterType* is MQSYSP_TYPE_SET:
CommandScope ,*DeallocateInterval* ,*LogCompression* ,*MaxArchiveLog* , *MaxConcurrentOffloads* ,*MaxReadTapeUnits* ,*OutputBufferCount*

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_SETindicates 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 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.

***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.

z/OS 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.

Set Policy

The Set Policy (MQCMD_CHANGE_PROT_POLICY) command sets the protection policy.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

Important: You must have an IBM MQ 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.

Syntax diagram

See the syntax diagram in the MQSC "SET POLICY" on page 1284 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:

MQESE_SIGN_ALG_NONE
No digital signature algorithm specified. This is the default value.

MQESE_SIGN_ALG_MD5
MD5 digital signature algorithm specified.

MQESE_SIGN_ALG_SHA1
SHA1 digital signature algorithm specified.

MQESE_SIGN_ALG_256
SHA256 digital signature algorithm specified.

MQESE_SIGN_ALG_384
SHA384 digital signature algorithm specified.

MQESE_SIGN_ALG_512
SHA512 digital signature algorithm specified.

***EncAlg* (MQCFIN)**
Specifies the encryption algorithm (parameter identifier: MQIA_ENCRYPTION_ALGORITHM). The following values are valid:

MQESE_ENC_ALG_NONE
No encryption algorithm specified. This is the default value.

MQESE_ENC_ALG_RC2
RC2 encryption algorithm specified.

MQESE_ENC_ALG_DES
DES encryption algorithm specified.

MQESE_ENC_ALG_3DES
3DES encryption algorithm specified.

MQESE_ENC_ALG_AES128
AES128 encryption algorithm specified.

MQESE_ENC_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:

MQESE_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.

MQESE_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.

***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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_POLICY_TYPE_ERROR
Policy type not valid.

z/OS Set System

Use the Set System (MQCMD_SET_SYSTEM) command to dynamically change certain general system parameter values initially set from your system parameter module at queue manager startup.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

Required parameters:
ParameterType

Optional parameters (if the value of *ParameterType* is MQSYSP_TYPE_SET:
CheckpointCount, *CommandScope*, *Exclmsg*, *MaxConnects*, *MaxConnectsBackground*, *MaxConnectsForeground*, *Service*, *SMFInterval*, *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.

***SMFInterval* (MQCFIN)**
The default time, in minutes, between each gathering of statistics (parameter identifier: MQIACF_SYSP_SMF_INTERVAL).

Specify a value in the range zero through 1440.

If you specify a value of zero, statistics data and accounting data are both 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.

Start Channel

The Start Channel (MQCMD_START_CHANNEL) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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 96 on page 1819

Table 96. ChannelDisposition and CommandScope for START CHANNEL

ChannelDisposition	***CommandScope* blank or local-qmgr**	***CommandScope* qmgr-name**	***CommandScope(*)***
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
MQCHLD_SHARED	For channels of *ChannelType* MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, start as a shared channel on the most suitable queue manager in the group. For a shared channel of *ChannelType* MQCHT_RECEIVER and MQCHT_SVRCONN, start the channel on all active queue managers. For a shared channel of *ChannelType* MQCHT_CLUSSDR and MQCHT_CLUSRCVR, this option is not permitted. MQCHLD_SHARED might automatically generate a command using *CommandScope* 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	Not permitted

Table 96. ChannelDisposition and CommandScope for START CHANNEL (continued)			
ChannelDisposition	***CommandScope*** **blank or local-qmgr**	***CommandScope*** **qmgr-name**	***CommandScope(*)***
MQCHLD_FIXSHARED	For a shared channel of *ChannelType* MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, with a nonblank *ConnectionName* , start as a shared channel on the local queue manager.	For a shared channel of *ChannelType* MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, with a nonblank *ConnectionName* , start as 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 1333.

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.

Start Channel (MQTT)

The Start Channel (MQCMD_START_CHANNEL) 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 1333.

***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.

Start Channel Initiator

The Start Channel Initiator (MQCMD_START_CHANNEL_INIT) command starts an IBM MQ channel initiator.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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 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.

***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 1333.

***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.

Start Channel Listener

The Start Channel Listener (MQCMD_START_CHANNEL_LISTENER) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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.

On platforms other than z/OS, this parameter is invalid.

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 1333.

***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.

Start Service

The Start Service (MQCMD_START_SERVICE) command starts an existing IBM MQ service definition.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

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 1333.

***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.

z/OS Start SMDS Connection

Use the Start SMDS Connection (MQCMD_INQUIRE_SMDSCONN) command 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

Stop Channel

The Stop Channel (MQCMD_STOP_CHANNEL) command stops an IBM MQ channel.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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

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 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 97 on page 1827

Table 97. ChannelDisposition and CommandScope for STOP CHANNEL			
ChannelDisposition	***CommandScope* blank or local-qmgr**	***CommandScope* qmgr-name**	***CommandScope(*)***
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
MQCHLD_SHARED	For channels of *ChannelType* MQCHT_RECEIVER or MQCHT_SVRCONN, stop as shared channel on all active queue managers. For channels of *ChannelType* 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. MQCHLD_SHARED might automatically generate a command using *CommandScope* 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	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.

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.

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 platforms other than z/OS, 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
z/OS On z/OS, MQMODE_TERMINATE is synonymous with FORCE. On distributed platforms, stop the channel immediately; the thread or process of the channel is terminated.

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.

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 1333.

***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.

Stop Channel (MQTT)

The Stop Channel (MQCMD_STOP_CHANNEL) command stops an IBM 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 IBM 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 1333.

***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.

z/OS Stop Channel Initiator

The Stop Channel Initiator (MQCMD_STOP_CHANNEL_INIT) command stops an IBM MQ channel initiator.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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.

Stop Channel Listener

The Stop Channel Listener (MQCMD_STOP_CHANNEL_LISTENER) command stops an IBM MQ listener.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	✓

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

z/OS

***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.

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.

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 1333.

***Reason* (MQLONG)**
The value can be any of the following values:

MQRCCF_LISTENER_STOPPED
Listener not running.

Stop Connection

The Stop Connection (MQCMD_STOP_CONNECTION) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

Stop Service

The Stop Service (MQCMD_STOP_SERVICE) command stops an existing IBM MQ service definition that is running.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
	✓	✓	✓	

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.

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 1333.

***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.

z/OS Stop SMDS Connection

Use the Stop SMDS Connection (MQCMD_STOP_SMDSCONN) command 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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 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 Suspend Queue Manager

The Suspend Queue Manager (MQCMD_SUSPEND_Q_MGR) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
				✓

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:

MQQMFAC_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.

MQQMFAC_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 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.

Suspend Queue Manager Cluster

The Suspend Queue Manager Cluster (MQCMD_SUSPEND_Q_MGR_CLUSTER) 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.

HP Integrity NonStop Server	IBM i	UNIX and Linux	Windows	z/OS
✓	✓	✓	✓	✓

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: 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.

***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 1333.

***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 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 form:

- PCF header (MQCFH) structure (described in topic "MQCFH - PCF header" on page 1837), 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 1840)
  - PCF byte string parameter (MQCFBS, see topic "MQCFBS - PCF byte string parameter" on page 1843)
  - PCF integer filter parameter (MQCFIF, see topic "MQCFIF - PCF integer filter parameter" on page 1845)
  - PCF integer list parameter (MQCFIL, see topic "MQCFIL - PCF integer list parameter" on page 1848)
  - PCF integer parameter (MQCFIN, see topic "MQCFIN - PCF integer parameter" on page 1850)
  - PCF string filter parameter (MQCFSF, see topic "MQCFSF - PCF string filter parameter" on page 1852)
  - PCF string list parameter (MQCFSL, see topic "MQCFSL - PCF string list parameter" on page 1856)
  - PCF string parameter (MQCFST, see topic "MQCFST - PCF string parameter" on page 1859)

How the structures are shown

The 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.

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 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 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 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 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 1327 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-STRUCLENGTH    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_STRUCLENGTH               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
  StrucLength 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 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.

z/OS

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  StrucLength;        /* 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(MQCFT_BYTE_STRING_FILTER),  /* Structure type */
   3 StrucLength  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
                            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  FBFPRM                 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 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 1837 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-STRUCLENGTH 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 StrucLength 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_STRUCLENGTH              DS   F        Structure length
MQCFBS_PARAMETER                DS   F        Parameter identifier
MQCFBS_STRINGLENGTH             DS   F        Length of string
                                ORG  MQCFBS
MQCFBS_AREA                     DS   CL(MQCFBS_LENGTH)
```

Visual Basic language declaration (Windows only)

```
Type MQCFBS
  Type As Long         ' Structure type
  StrucLength 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 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.

z/OS

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
                                ORG  MQCFIF
MQCFIF_AREA                     DS   CL(MQCFIF_LENGTH)
```

Visual Basic language declaration (Windows only)

```
Type MQCFIF
  Type As Long         ' Structure type
  StrucLength 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 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 1837 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
                                ORG  MQCFIL
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 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 1837 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-STRUCLENGTH 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_STRUCLENGTH              DS   F        Structure length
MQCFIN_PARAMETER                DS   F        Parameter identifier
MQCFIN_VALUE                    DS   F        Parameter value
MQCFIN_LENGTH                   EQU  *-MQCFIN Length of structure
                                ORG  MQCFIN
MQCFIN_AREA                     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 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.

z/OS

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: 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.

***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-STRUCLENGTH 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_STRUCLENGTH            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
  StrucLength 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 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 “Usage notes” on page 1836 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 1837 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  StrucLength;     /* 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 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 “Usage notes” on page 1836 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:

MQCFT_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 1837 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-STRUCLENGTH    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 StrucLength    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_STRUCLENGTH              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
```

```
                                ORG  MQCFST
MQCFST_AREA                     DS   CL(MQCFST_LENGTH)
```

Visual Basic language declaration (Windows only)

```
Type MQCFST
  Type As Long            ' Structure type
  StrucLength 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  STTYP                  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

The compiled program, written in C language, in the example uses IBM MQ for Windows. It 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 recreate 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, "SAVEQMGR.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  */
/* nameof 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
                       );


/* Setup 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;

/* Setup 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   = MQ_Q_NAME_LENGTH;
memset( pPCFString->String, ' ', MQ_Q_NAME_LENGTH );
memcpy( pPCFString->String, "*", 1 );

/* Setup parameter block */
pPCFInteger->Type        = MQCFT_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 == MQOO_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, "       TRIGDPTH(%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 );


sprintf( ParmBuffer, "       RETINTVL(%d) +\n", DefnLQ.RetentionInterval );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "       BOTHRESH(%d) +\n", DefnLQ.BackoutThreshold );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "       BOQNAME('%s') +\n", DefnLQ.BackoutReqQName );
fputs( ParmBuffer, fp );

if ( DefnLQ.Scope == MQSCO_Q_MGR ) {
   sprintf( ParmBuffer, "       SCOPE(QMGR) +\n" );
   fputs( ParmBuffer, fp );
} else {
   sprintf( ParmBuffer, "       SCOPE(CELL) +\n" );
   fputs( ParmBuffer, fp );
} /* endif */

sprintf( ParmBuffer, "       QDEPTHHI(%d) +\n", DefnLQ.QDepthHighLimit );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "       QDEPTHLO(%d) +\n", DefnLQ.QDepthLowLimit );
fputs( ParmBuffer, fp );

if ( DefnLQ.QDepthMaxEvent == MQEVR_ENABLED ) {
   sprintf( ParmBuffer, "       QDPMAXEV(ENABLED) +\n" );
   fputs( ParmBuffer, fp );
} else {
   sprintf( ParmBuffer, "       QDPMAXEV(DISABLED) +\n" );
   fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.QDepthHighEvent == MQEVR_ENABLED ) {
   sprintf( ParmBuffer, "       QDPHIEV(ENABLED) +\n" );
   fputs( ParmBuffer, fp );
} else {
   sprintf( ParmBuffer, "       QDPHIEV(DISABLED) +\n" );
   fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.QDepthLowEvent == MQEVR_ENABLED ) {
   sprintf( ParmBuffer, "       QDPLOEV(ENABLED) +\n" );
   fputs( ParmBuffer, fp );
} else {
   sprintf( ParmBuffer, "       QDPLOEV(DISABLED) +\n" );
   fputs( ParmBuffer, fp );
```

```
   } /* endif */

   sprintf( ParmBuffer, "       QSVCINT(%d) +\n", DefnLQ.QServiceInterval );
   fputs( ParmBuffer, fp );

   switch ( DefnLQ.QServiceIntervalEvent ) {
   case MQQSIE_OK:
      sprintf( ParmBuffer, "       QSVCIEV(OK)\n" );
      fputs( ParmBuffer, fp );
      break;
   case MQQSIE_NONE:
      sprintf( ParmBuffer, "       QSVCIEV(NONE)\n" );
      fputs( ParmBuffer, fp );
      break;
   case MQQSIE_HIGH:
      sprintf( ParmBuffer, "       QSVCIEV(HIGH)\n" );
      fputs( ParmBuffer, fp );
      break;
   } /* endswitch */

   sprintf( 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 nonblank 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;

   /* setup 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 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 );
}
}
```

z/OS Using the IBM MQ utilities for z/OS

Reference information about the syntax, and usage of the various IBM MQ utility programs.

An overview of the IBM MQ utilities

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:

Table 98 on page 1873
Table 99 on page 1873
Table 100 on page 1874
Table 102 on page 1874
Table 103 on page 1875 summarizes what you can do with these utilities.

Table 98. The IBM MQ CSQUTIL utility program: Managing page sets

Purpose	Function	See topic
Format VSAM data sets as IBM MQ page sets.	FORMAT	"Formatting page sets (FORMAT)" on page 1879
Control recovery processing used for IBM MQ page sets.	FORMAT	"Formatting page sets (FORMAT)" on page 1879
Extract page set information.	PAGEINFO	"Page set information (PAGEINFO)" on page 1882
Copy IBM MQ page sets.	COPYPAGE	"Expanding a page set (COPYPAGE)" on page 1883
Copy IBM MQ page sets and reset the log information.	RESETPAGE	"Copying a page set and resetting the log (RESETPAGE)" on page 1885

Table 99. The IBM MQ CSQUTIL utility program: Issuing commands

Purpose	Function	See topic
Issue IBM MQ commands.	COMMAND	"Using the COMMAND function of CSQUTIL" on page 1887
Produce a set of DEFINE, ALTER or DELETE commands for objects.	COMMAND	Making a list of DEFINE commands
Produce a client channel definition file.	COMMAND	Making a client channel definition file
Produce a set of DEFINE commands for objects (offline).	SDEFS	"Producing a list of IBM MQ define commands (SDEFS)" on page 1894

Table 100. The IBM MQ CSQUTIL utility program: Managing queues		
Purpose	**Function**	**See topic**
Copy contents of a queue to a data set.	COPY	"Copying queues into a data set while the queue manager is running (COPY)" on page 1897
Copy contents of a queue to a data set (offline).	SCOPY	"Copying queues into a data set while the queue manager is not running (SCOPY)" on page 1899
Delete contents of a queue.	EMPTY	"Emptying a queue of all messages (EMPTY)" on page 1902
Restore contents of a queue.	LOAD	"Restoring messages from a data set to a queue (LOAD)" on page 1903

Table 101. The IBM MQ CSQUTIL utility program: Migrating CSQXPARM		
Purpose	**Function**	**See topic**
Produce an ALTER QMGR command from a channel initiator parameter module.	XPARM	"Migrating a channel initiator parameter module (XPARM)" on page 1907

Table 102. The IBM MQ CSQJU003 Change log inventory utility		
Purpose	**Function**	**See topic**
Add active or archive log data sets.	NEWLOG	"Adding information about a data set to the BSDS (NEWLOG)" on page 1911
Delete active or archive log data sets.	DELETE	"Deleting information about a data set from the BSDS (DELETE)" on page 1914
Supply passwords for archive logs.	ARCHIVE	"Supplying a password for archive log data sets (ARCHIVE)" on page 1914

Table 102. The IBM MQ CSQJU003 Change log inventory utility (continued)		
Purpose	**Function**	**See topic**
Control the next restart of the queue manager.	CRESTART	"Controlling the next restart (CRESTART)" on page 1915
Set checkpoint records.	CHECKPT	"Setting checkpoint records (CHECKPT)" on page 1916
Update the highest written log RBA.	HIGHRBA	"Updating the highest written log RBA (HIGHRBA)" on page 1917

Table 103. The remaining IBM MQ utilities		
Name	**Purpose**	**See topic**
CSQJU004 (Print log map utility)	List information about the log.	"The print log map utility (CSQJU004)" on page 1918
CSQ1LOGP (Log print utility)	Print the log. Extract log records into sequential files.	"The log print utility (CSQ1LOGP)" on page 1919
CSQ5PQSG (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.	"The queue-sharing group utility (CSQ5PQSG)" on page 1929
CSQJUFMT (Active log preformat utility)	Preformat log data sets Preformat Shared Message Data Sets (SMDS)	"The active log preformat utility (CSQJUFMT)" on page 1933
CSQUDLQH (Dead-letter queue handler utility)	Process messages on the dead-letter queue.	"The dead-letter queue handler utility (CSQUDLQH)" on page 1934
CSQUCVX (Data conversion exit utility)	Generate data conversion exit routines.	Writing a data-conversion exit program for IBM MQ for z/OS

These utilities are located in the thlqual.SCSQAUTH or thlqual.SCSQLOAD IBM MQ load libraries. Concatenate the appropriate IBM MQ language load library thlqual.SCSQANLx (where x is the language letter) in the STEPLIB with the thlqual.SCSQAUTH and thlqual.SCSQLOAD. The utility control statements are available only in U.S. English. In some cases, the Db2 library db2qual.SDSNLOAD is also needed.

Syntax diagrams

The syntax for a command and its options is presented in the form of a syntax diagram called a railroad diagram.

Railroad diagrams are a visual format suitable for sighted users; see, "How to read railroad diagrams" on page 7. It tells you what options you can supply with the command, how to enter them, indicates relationships between different options, and sometimes different values of an option.

IBM MQ utility program (CSQUTIL)

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

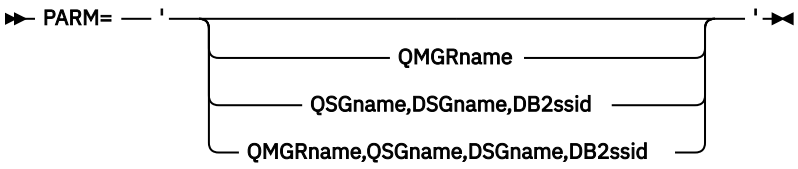
Use this topic to understand how to invoke CSQUTIL, the format of its parameters, and its return codes.

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 1. 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 1 on page 1877 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) TOQLOCAL(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

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)

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.

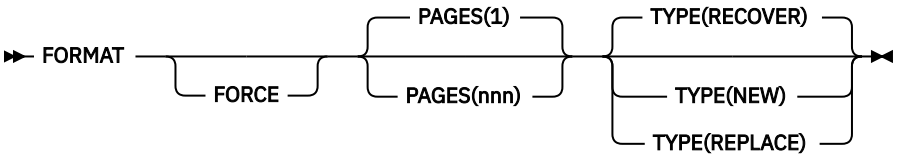
In summary:

- to reinstate a page set with no data, use FORMAT with the TYPE(NEW) option

- to reinstate a page set with old data, use FORMAT with the TYPE(REPLACE) option
- to reinstate 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

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.

Page set management (FORMAT)



- 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 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.

Example

Figure 2 on page 1882 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 2. Sample JCL for the FORMAT function of CSQUTIL

Figure 3 on page 1882 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 3. 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)

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)

Keywords and parameters

There are no keywords or parameters.

Example

In Figure 4 on page 1883, 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 4. 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 *DFSMS/MVS Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* 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)

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.

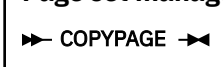
Each page set is copied to a destination data set that must be formatted as a page set. Copying to a smaller page set is not supported.

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)



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.

```
//JOBLIB  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 pageset
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
DELETE 'VICY.MQ38.PAGE01.NEW' CLUSTER
DEFINE CLUSTER (NAME('VICY.MQ38.PAGE01.NEW') +
MODEL('VICY.MQ38.PAGE01') +
DATACLAS(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 5. 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 5 on page 1885.
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 *DFSMS/MVS Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* 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)

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.

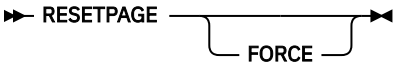
In this situation, use the RESETPAGE function on **all** the page sets of the affected queue manager. You must also create new BSDS and log data sets.

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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.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 6. 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 6 on page 1887.
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.

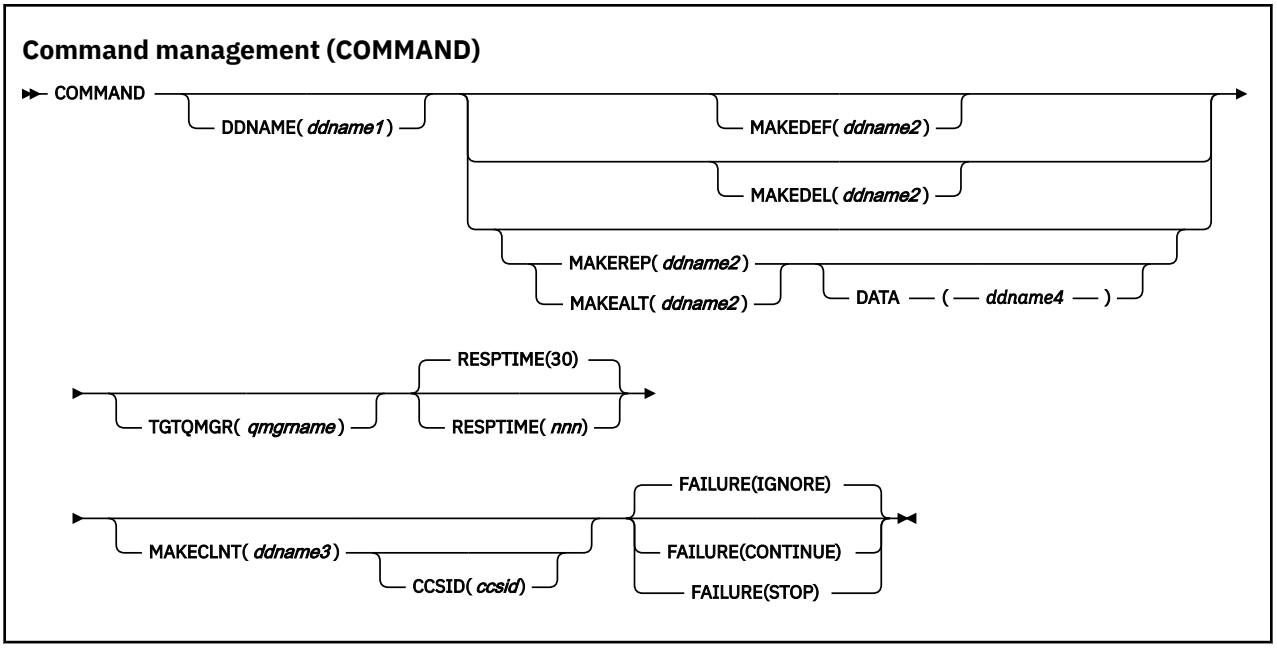
Using the COMMAND function of CSQUTIL

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 recreate 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 recreate 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: The MAKECLNT utility is now stabilized at the IBM WebSphere MQ 7.1 level. You should use the **runmqsc** command using the **-n** option; see “runmqsc” on page 116 for further information.

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 1890
- "Making a list of DEFINE commands" on page 1890
- "Making a list of ALTER commands" on page 1891
- "Making a client channel definition file" on page 1892

Issuing commands

In Figure 7 on page 1890, 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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.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 TGTQMGR(CSQ2) RESPTIME(10)
/*
```

Figure 7. Sample JCL for issuing IBM MQ commands using CSQUTIL

Making a list of DEFINE commands

In Figure 8 on page 1891, 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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.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 8. Sample JCL for using the MAKEDEF option of the COMMAND function

Making a list of ALTER commands

In Figure 9 on page 1891, 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=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.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 9. Sample JCL for using the MAKEALT option of the COMMAND function

Making a client channel definition file

In Figure 10 on page 1892, 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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.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 10. 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 "Building command scripts" on page 628 for more information about the rules for building IBM MQ commands.
2. The output from a "DISPLAY QMGR" on page 1122 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 initiatoi 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 8 on page 1891.
   - 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.

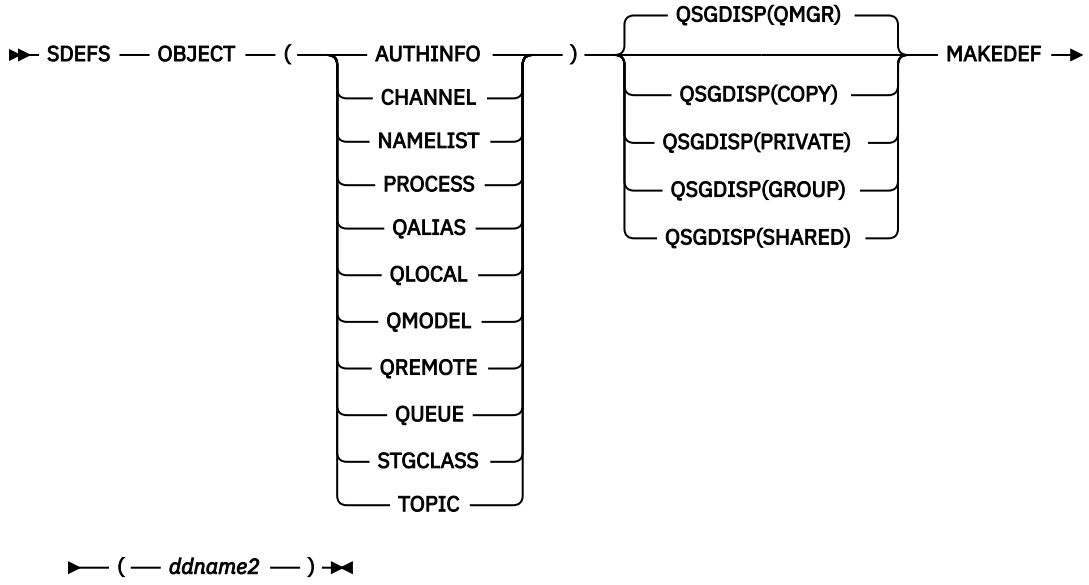
Producing a list of IBM MQ define commands (SDEFS)

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 recreate 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 recreate definitions for a queue manager that is not currently running. This makes the CSQUTIL SDEFS option more appropriate for recovery scenarios.

Command management (SDEFS)



- 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 104 on page 1895.

Table 104. SDEFS QSGDISP parameters and their actions

QSGDISP parameter	What the SDEFS utility does
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.
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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.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 11. Sample JCL for the SDEFS function of CSQUTIL

```
//SDEFS   EXEC  PGM=CSQUTIL,PARM='Qsgname,Dsgname,Db2name'
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//         DD   DISP=SHR,DSN=db2qual.SDSNLOAD
//OUTPUT1  DD   DISP=OLD,DSN=MY.COMMANDS(DEFS)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
SDEFS OBJECT(QLOCAL) QSGDISP(SHARED) MAKEDEF(OUTPUT1)
/*
```

Figure 12. 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=thlqual.SCSQAUTH
//         DD DISP=SHR,DSN=thlqual.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(QUEUE)
//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(QUEUE) MAKEDEF(OUTPUT9)
SDEFS OBJECT(STGCLASS) MAKEDEF(OUTPUT0)
SDEFS OBJECT(TOPIC) MAKEDEF(OUTPUTA)
/*
```

Figure 13. 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 reference

IBM MQ utility program (CSQUTIL)

Copying queues into a data set while the queue manager is running (COPY)

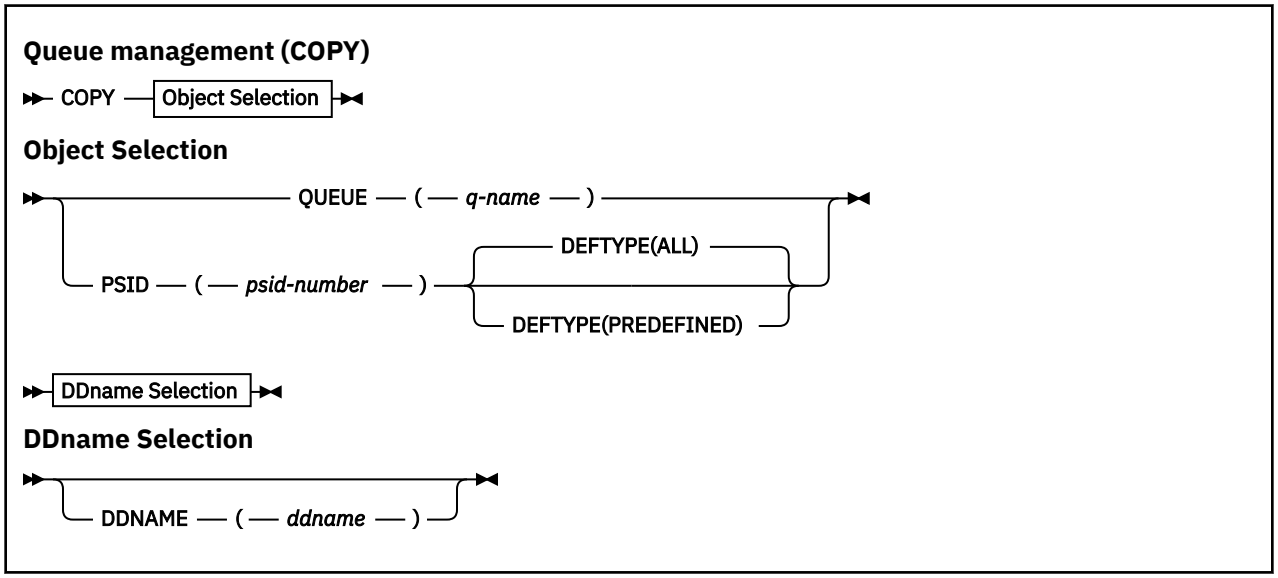
You can use the COPY function of CSQUTIL to copy queued messages to a sequential data set while the queue manager is running, without destroying any messages in the original queues.

The scope of the COPY 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 appropriate queues.

Note:

1. If you want to copy the object definitions from the named page set, use COPYPAGE.
2. If you want to copy messages to a data set when the queue manager is stopped, use SCOPY.
3. For information about how to avoid problems with duplicate messages if this function fails, see Syncpoints in IBM MQ for z/OS applications.
4. An alternative approach to the COPY function is to use the "dmpmqmsg" on page 46 utility which is more flexible in many cases.



- 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.

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.

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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.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 14. 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.

Copying queues into a data set while the queue manager is not running (SCOPY)

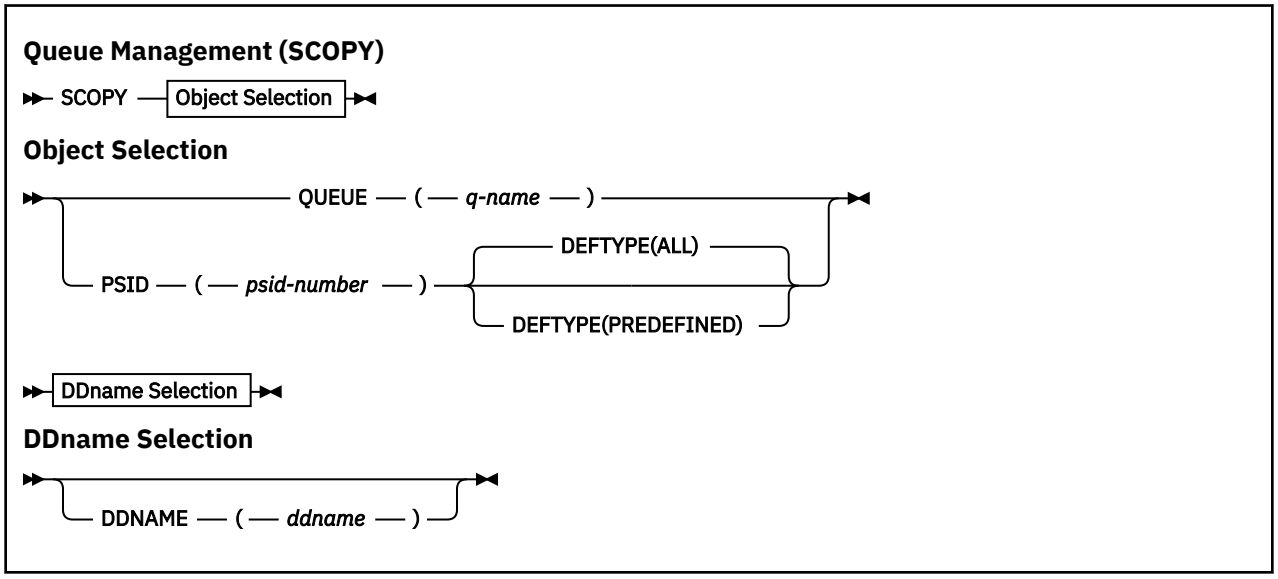
You can use the SCOPY function of CSQUTIL to copy queued messages to a sequential data set when the queue manager is not running, without destroying any messages in the original queues.

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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.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)
//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 15. 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.

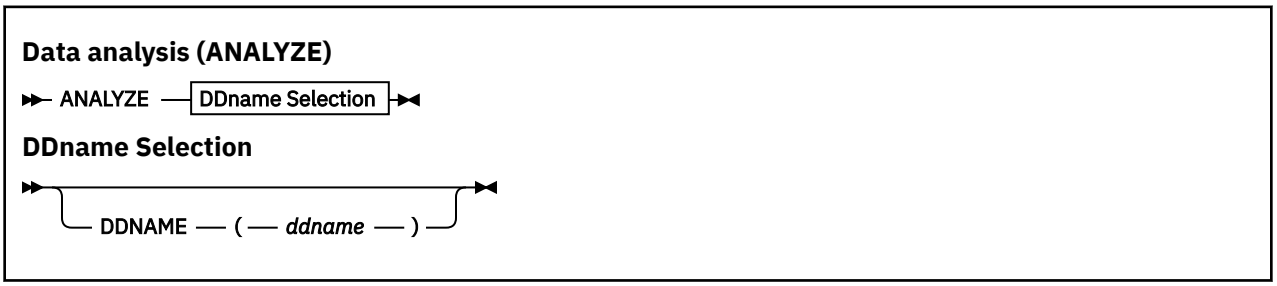
5. An SCOPY PSID function is considered successful only if it successfully copies all the queues on the page set. If an empty queue is processed, data indicating this is written to the sequential data set. If the page set has no queues, the SCOPY function fails, and no data is written to the data set.
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

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 1901
- "Example" on page 1902
- "Usage notes" on page 1902

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=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUTA DD DSN=MY.UTILITY.OUTPUTA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ANALYZE DDNAME(OUTPUTA)
```

Figure 16. 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)

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 1878 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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
EMPTY QUEUE(SPARE)
EMPTY PSID(66)
/*
```

Figure 17. 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 page 1877
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)

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.

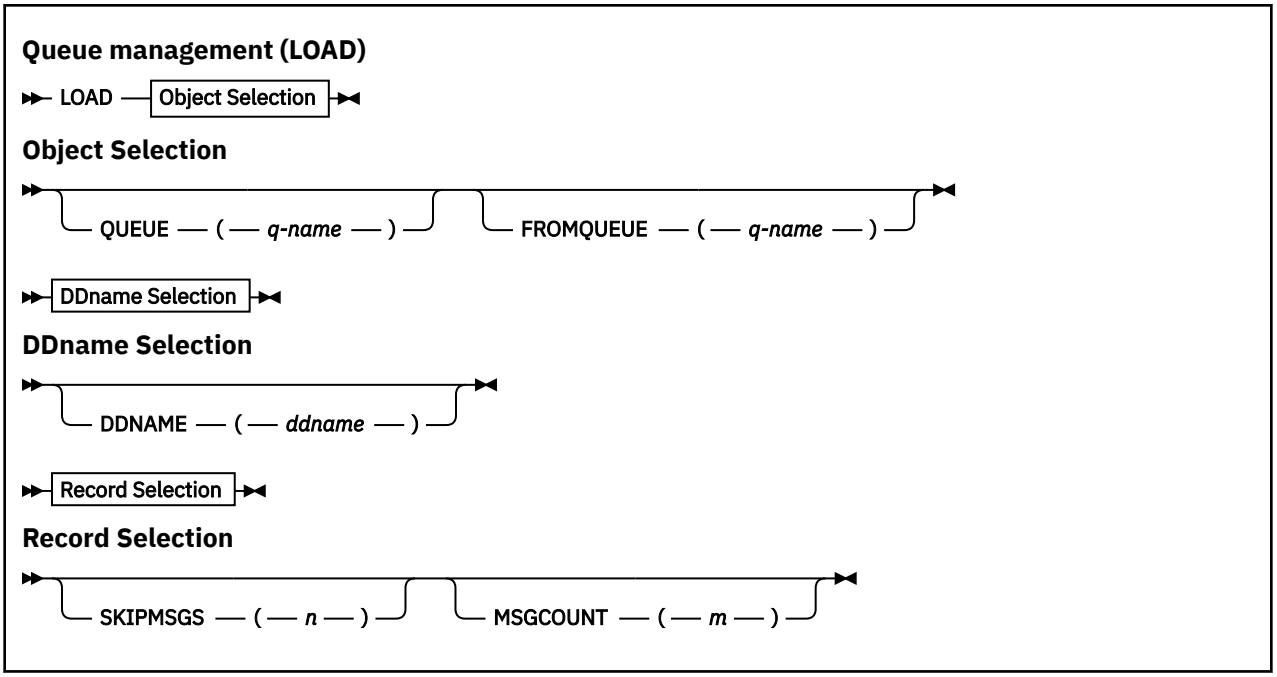
The data set can contain messages from one queue only if it was created by COPY or SCOPY QUEUE, or from a number of queues if it was created by COPY PSID or several successive COPY or SCOPY QUEUE operations. Messages are restored to queues with the same name as those from which they were copied. You 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 1878 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” on page 46 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=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.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 18. 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 dataset 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)

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.

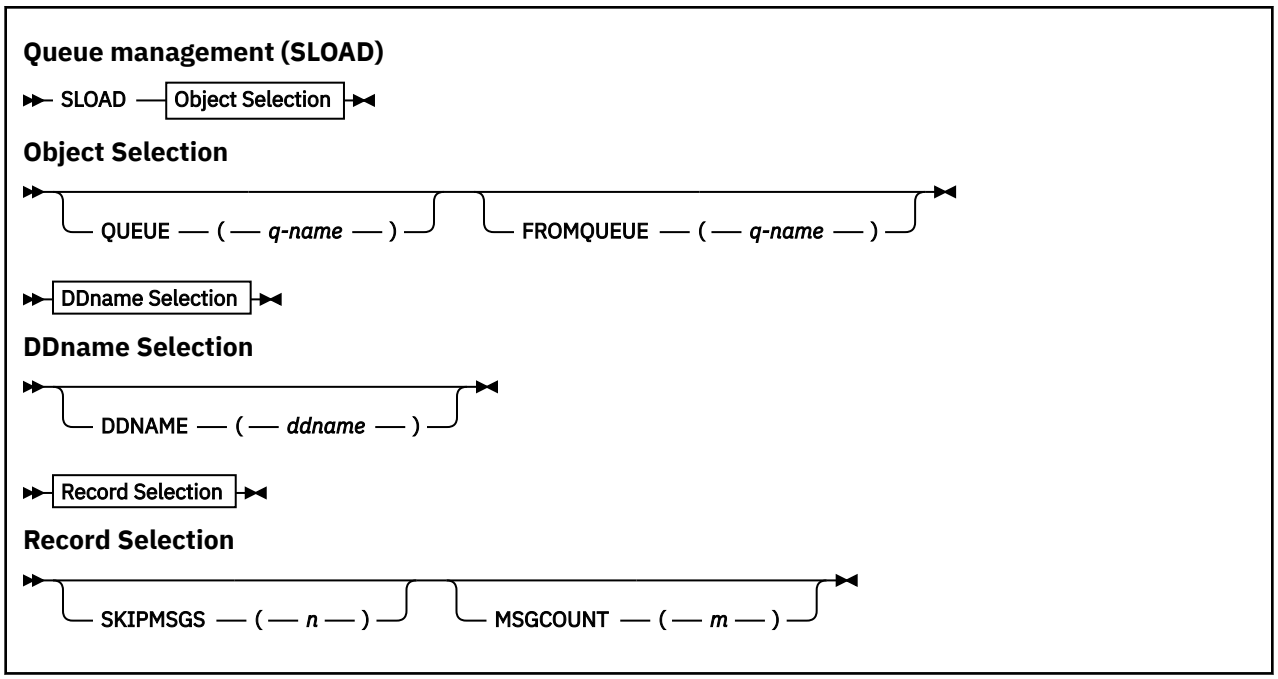
To use SLOAD the queue manager must be running.

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 1878 for information about how to avoid problems with duplicate messages if this function fails.



- "Keywords and parameters" on page 1906
- "Example" on page 1907
- "Usage notes" on page 1907

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.

MSGCOUNT(*m*)
Specifies that only *m* messages are to be read from the data set and loaded to the queue.

If you omit MSGCOUNT(*m*) the number of messages read is unlimited.

Example

```
//SLOAD EXEC PGM=CSQUTIL,PARM=('CSQ1'),REGION=0M
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUTA DD DSN=MY.UTILITY.OUTPUTA,DISP=SHR
//CSQUINP DD DSN=MY.UTILITY.COPYA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SLOAD DDNAME(OUTPUTA)
SLOAD QUEUE(TOQ) FROMQUEUE(QUEUEA) SKIPMSGS(55)
/*
```

Figure 19. Sample JCL for the CSQUTIL SLOAD 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.
- SLOAD DDNAME(OUTPUTA) - Reloads the first queue from the input data set MY.UTILITY.OUTPUTA. The name of the queue loaded is the same as the queue name from which the data was copied.
- SLOAD QUEUE(TOQ) FROMQUEUE(QUEUEA) SKIPMSGS(55) - Reloads the messages that were copied from the queue QUEUEA (from the input data set MY.UTILITY.COPYA). The messages are reloaded to the queue called TOQ. In processing the messages in QUEUEA, the first 55 messages are ignored, and loading starts from the 56th message.

Usage notes

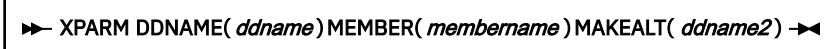
1. To use the SLOAD function, the queues or page sets involved must not be in use when the function is invoked.
2. If SLOAD fails, or is forced to take a syncpoint, no further CSQUTIL functions are attempted.
3. CSQUTIL uses MQPMO_SET_ALL_CONTEXT to ensure that the message descriptor fields remain the 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.

Migrating a channel initiator parameter module (XPARM)

You can use the XPARM function of CSQUTIL to generate ALTER QMGR command that can be used to migrate to Version 7.0.

In versions of IBM MQ for z/OS before Version 6.0, you could tailor the channel initiator by creating a channel initiator parameter load module. In Version 7.0, you do it by setting queue manager attributes. To make it easier to migrate to Version 7.0, this command generates an ALTER QMGR command from a pre-Version 6.0 channel initiator parameter module.

Migration (XPARM)



Keywords and parameters

DDNAME(*ddname*)
Specifies that an ALTER QMGR command is to be generated from a channel initiator parameter module in this data set.

MEMBER(*membername*)
Specifies the name of the channel initiator parameter module in the data set specified by DDNAME(ddname2).

MAKEALT(*ddname2*)
Specifies the DDname that identifies the output data set in which the ALTER command is 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 CSQINP2 initialization input data sets.

Example

```
//MIGRATE1 EXEC PGM=CSQUTIL
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQXPARM DD DISP=SHR,DSN=user.loadlib
//SYSPRINT DD SYSOUT=*
//ALTQMGR  DD DISP=OLD,DSN=user.commands(ALTQMGR)
//SYSIN DD *
XPARM DDNAME(CSQXPARM) MEMBER(MQ3AXPRM) MAKEALT(ALTQMGR)
/*
```

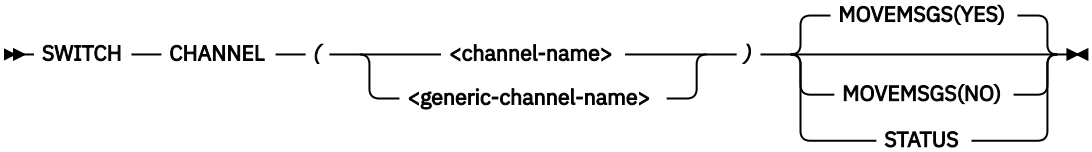
Figure 20. Sample JCL for the CSQUTIL XPARM function

Switch the transmission queue associated with cluster-sender channels (SWITCH)

You can use the SWITCH function of CSQUTIL to switch or query the transmission queue associated with cluster-sender channels.

To use the SWITCH function the queue manager must be running.

Switch transmission queue (SWITCH)



- 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=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
SWITCH CHANNEL(CLUSTER.*) STATUS
/*
```

Figure 21. 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=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
SWITCH CHANNEL(CLUSTER.TO.QM1)
/*
```

Figure 22. 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.
6. You cannot use the SWITCH function to query the status of, or switch the transmission queue for, a cluster-sender channel that has not been started since version 8 new function has been enabled using the OPMODE parameter in the CSQ6SYSP macro.

Related tasks

Clustering: Switching cluster transmission queues

z/OS The change log inventory utility (CSQJU003)

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” on page 900 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

Use this topic to understand how to invoke the CSQJU003 utility.

The utility runs as a z/OS batch program. Figure 23 on page 1911 gives an example of the JCL required.

```
//JU003   EXEC  PGM=CSQJU003
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD  SYSOUT=*,DCB=BLKSIZE=629
//SYSUT1   DD  DISP=SHR,DSN=bsds.dsname
//SYSIN    DD  *
  NEWLOG DSNAME=CSQREPAL.A0001187,COPY1VOL=CSQV04,UNIT=SYSDA,
  STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=YES,PASSWORD=PASSWRD
/*
```

Figure 23. 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.

z/OS Adding information about a data set to the BSDS (NEWLOG)

You can use the NEWLOG function of CSQJU003 to add information about a data set to BSDS.

The NEWLOG function declares one of the following data sets:

- A VSAM data set that is available for use as an active log data set.

Use the keywords DSNAME, COPY1, COPY2, and PASSWORD.

- An active log data set that is replacing one that encountered an I/O error.

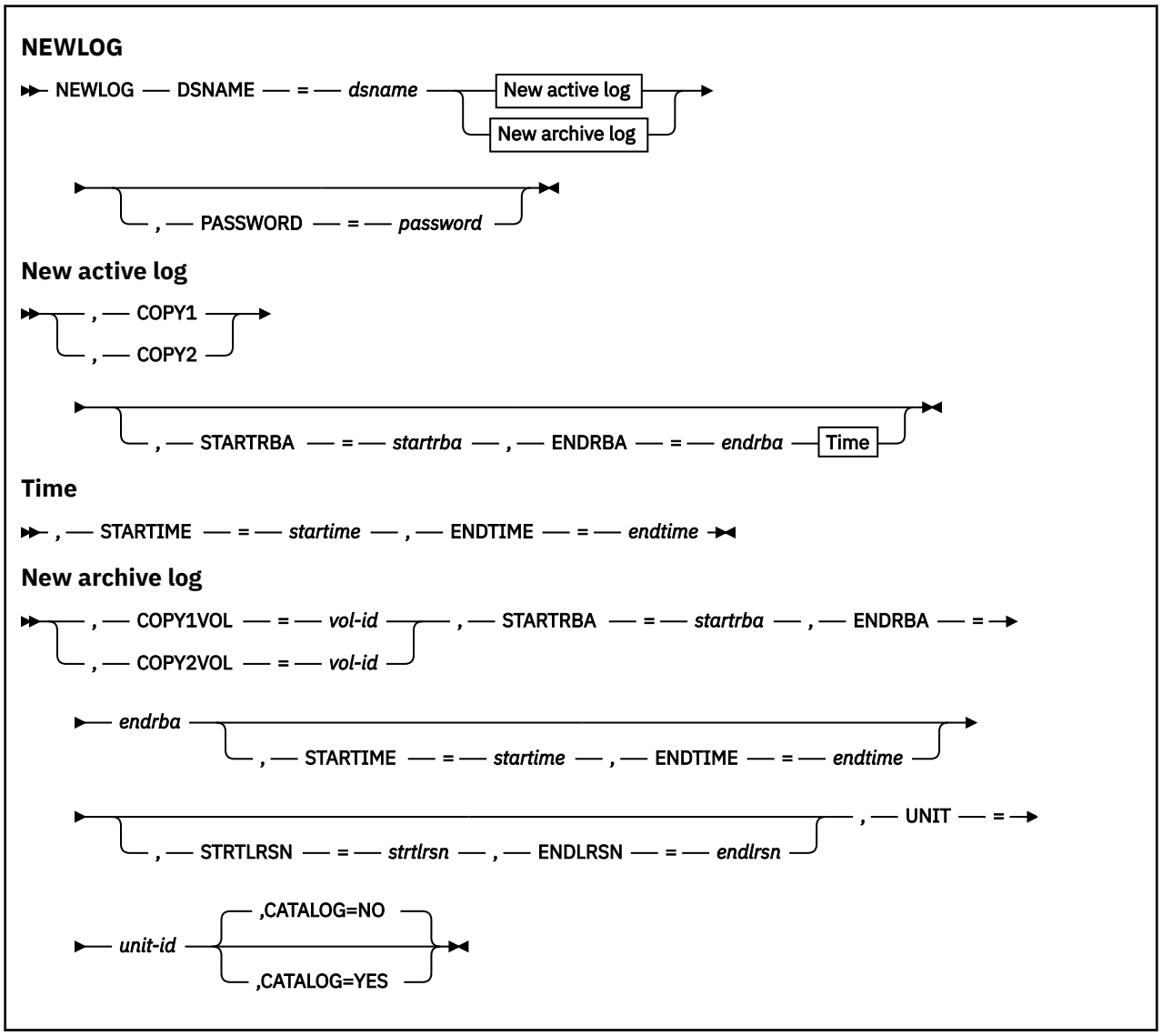
  Use the keywords DSNAME, COPY1, COPY2, STARTRBA, ENDRBA, and PASSWORD.

- An archive log data set volume.

  Use the keywords DSNAME, COPY1VOL, COPY2VOL, STARTRBA, ENDRBA, STRTLRSN, ENDLRSN, UNIT, CATALOG, and PASSWORD.

  In a queue-sharing group environment, you should always supply LRSN information. Run the print log map utility (CSQJU004) to find RBAs and LRSNs to use for archive log data sets.

A maximum of 310 data sets can be defined for each log copy, either by this NEWLOG function or the MQSC DEFINE LOG command.



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 (& * + - . ; ' /).

We recommend that you 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 FFFFFFFFF000 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 FFFFFFFFFFFF cannot be specified for a version 1 format BSDS.

STARTIME= ***startime***
Start time of the RBA in the BSDS. This is an optional field. The time stamp format (with valid values in parentheses) is yyyydddhhmmsst, 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.

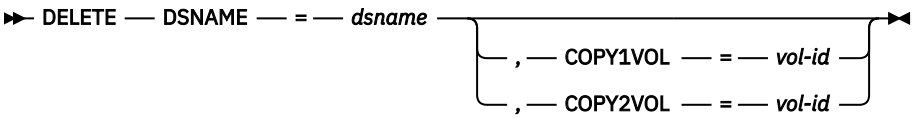
IBM MQ requires that all archive log data sets on DASD be cataloged. Select CATALOG=YES if the archive log data set is on DASD.

Deleting information about a data set from the BSDS (DELETE)

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.

DELETE



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.

Supplying a password for archive log data sets (ARCHIVE)

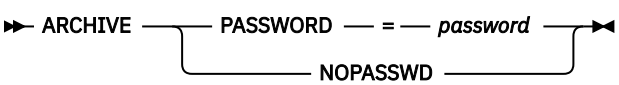
You can use the ARCHIVE function of CSQJU003 to assign a password to all archive data sets created after this operation.

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.

ARCHIVE



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.

Controlling the next restart (CRESTART)

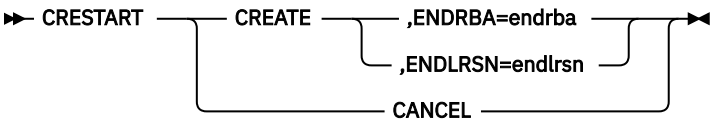
You can use the CRESTART function of CSQJU003 to control the next restart of the queue manager, either by creating a new conditional restart control record or by canceling the one currently active.

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.

CRESTART



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 FFFFFFFFF000 cannot be specified for a version 1 format BSDS.

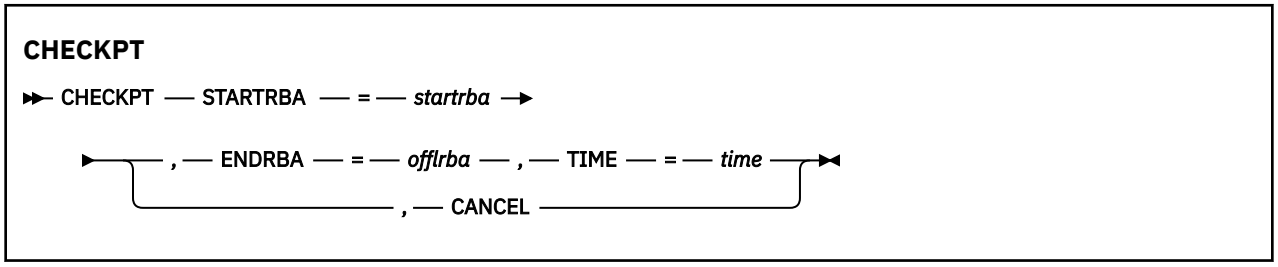
ENDLRSN= ***endlrsn***
Gives the LRSN of the last log record to be used during restart (the point at which the log is to be truncated). Any log information in the bootstrap data set and the active logs with an LRSN greater than endlrsn is discarded.

Setting checkpoint records (CHECKPT)

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.



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 FFFFFFFFFFFF 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 FFFFFFFFFFFF 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 yyyydddhhmmsst, 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.

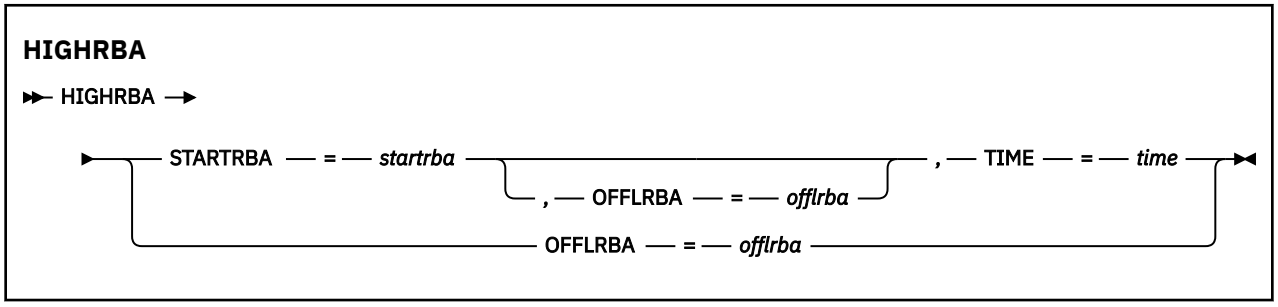
CANCEL
Deletes the checkpoint queue record containing a starting RBA that matches the RBA specified by STARTRBA.

Updating the highest written log RBA (HIGHRBA)

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 FFFFFFFFFFFF 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 yyyydddhhmmsst, 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.

The print log map utility (CSQJU004)

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=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=bsds.dsname
```

Figure 24. 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.

The log print utility (CSQ1LOGP)

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 25 on page 1920, Figure 26 on page 1921, Figure 27 on page 1921 and Figure 28 on page 1921.

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 1921 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).

ACTIVEn
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.

CSQOBJS
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).

```
//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 *
* extract records for pageset 3. Produce both summary and detail reports
PAGESET(3)
SUMMARY(YES)
/*
```

Figure 25. 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 26. 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 27. Sample JCL to invoke the CSQ1LOGP utility using archive log data sets

```
//PRTLOG  EXEC PGM=CSQ1LOGP
...
//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)
//CSQOBJS   DD DSN=objects.dataset,DISP=(NEW,CATLG)
```

Figure 28. Sample JCL showing additional statements for the EXTRACT keyword

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 A00000000000; 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 FFFFFFFFFFFF (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 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 1924. 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.

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.
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 CSQOBJS 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 29. 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

Sample	Description
thlqual.SCSQLOAD(CSQ4LOGS)	Sample C program to: • Report on the UOW activity and object defines and alters • Optionally, replay messages.
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
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:

AAAAAAAA
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:

Offset		Type	Length	Name	Description
Dec	Hex				
0	0	Character	21	csrecorddate	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	csqueuename	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 object name or queue name 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	csdmcp	Non-shared queue message key. Blank if a shared queue.

Offset		Type	Length	Name	Description
175	AF	Character	8	csverb	Activity: **ALTER** the object was changed **DEFINE** the object was created **MQGET** the message was got **MQPUT** the message was put **EXPIRE** the message expired **ABORT2** the message was backed out **PHASE1** the first phase of two-phase commit **PHASE2** the second phase of two-phase commit, or the only phase of one phase commit
183	B7	Character	1	cscmitstatus	Status of unit of work: **B** backed out **C** committed **I** inflight
184	B8	Character	1	csshunt	Shunted indicator: **S** shunted record **N** not shunted
185	B9	Character	8	cslogrba	RBA of log record
193	C1	Character	8	csshuntrba	RBA of shunted log record
201	C9	Character	1	csuowscope	UOW scope in hexadecimal: **01** local **02** shared
202	CA	Integer	4	lsegment	The segment number of the data, starting from 1.
206	CE		Variable		Data part

Offset		Type	Length	Name	Description
206	CE	Character	1	csbora	If csverb is ALTER, indicates whether the data is the 'before' or 'after' copy of the object. **B** before **A** after
207	CF	Character	Variable	csvardata	Message or object data. Length as given in ldatalen.

The queue-sharing group utility (CSQ5PQSG)

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 30 on page 1929 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=thlqual.SCSQANLE,DISP=SHR
//         DD DSN=thlqual.SCSQAUTH,DISP=SHR
//         DD DSN=db2qual.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Figure 30. Sample JCL to invoke the CSQ5PQSG utility

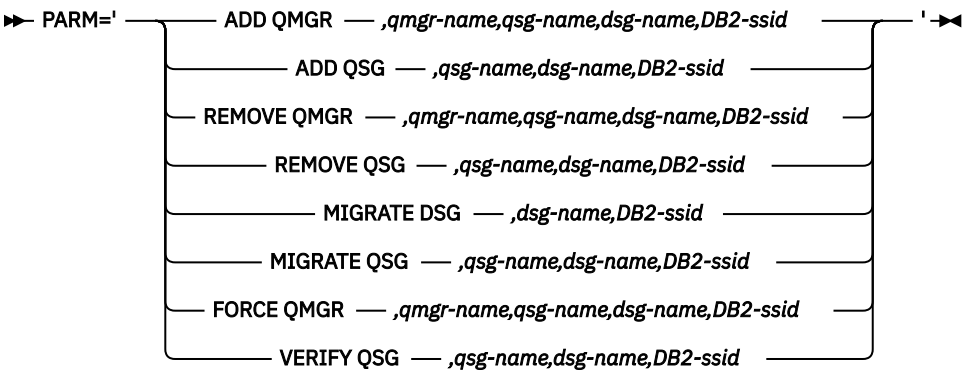
Data definition statements

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



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
Check that the installation is ready to migrate its Db2 data-sharing group table definitions from IBM WebSphere MQ 7.0, or Version 7.1 to IBM MQ 8.0.

It verifies that:

- All the queue managers in the data-sharing group have applied the migration & coexistence PTF and have since been started.
- The data-sharing group has not already been migrated.

dsg-name
The Db2 data-sharing group name

DB2-ssid
The Db2 subsystem ID

This function does not actually do the migration, which involves several steps. It is included as part of the sample migration job CSQ4570T and CSQ4571T in thlqual SCSQPROC.

Migration requires that a migration PTF is installed on **all** queue managers in the data-sharing group.

MIGRATE QSG

Check that the installation is ready to migrate its Db2 queue-sharing group table definitions from IBM WebSphere MQ 7.0, or Version 7.1, to IBM MQ 8.0

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.

It verifies that:

- All the queue managers in the queue-sharing group have applied the migration & coexistence PTF and have since been started.

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 actually do the migration, which involves several steps. It is included as part of the sample migration job CSQ4570T and CSQ4571T in thlqual SCSQPROC.

Migration requires that a migration PTF is installed on **all** queue managers in the queue-sharing group.

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=thlqual.SCSQANLE,DISP=SHR
//         DD DSN=thlqual.SCSQAUTH,DISP=SHR
//         DD DSN=db2qual.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Figure 31. Using the queue-sharing group utility to add a queue manager into a queue-sharing group

The active log preformat utility (CSQJUFMT)

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 CSQJUFMT utility formats a single active log dataset. 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 datasets, 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.

```
//JOBLIB    DD   DISP=SHR,DSN=thlqual.SCSQANLE
//          DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//*
//JUFMT11   EXEC PGM=CSQJUFMT
//SYSPRINT  DD   SYSOUT=*
//SYSUT1    DD   DISP=OLD,DSN=hlq.LOGCOPY1.DS01
//*
//JUFMT21   EXEC PGM=CSQJUFMT
//SYSPRINT  DD   SYSOUT=*
//SYSUT1    DD   DISP=OLD,DSN=hlq.LOGCOPY2.DS01
```

Figure 32. 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.

The dead-letter queue handler utility (CSQUDLQH)

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.

Invoking the DLQ handler

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 33. 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 34. 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 page 1936.

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=thlqual.SCSQAUTH,DISP=SHR
//         DD DSN=thlqual.SCSQLOAD,DISP=SHR
//         DD DSN=thlqual.SCSQANLE,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSIN    DD *
INPUTQM(CSQ2) INPUTQ('CSQ2.DEAD.QUEUE')
ACTION(RETRY)
/*
```

Figure 35. Sample JCL to invoke the CSQUDLQH utility

The DLQ handler rules table

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 1935.
- 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 page 1939 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* |' ')
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* |' ')
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)
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 |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)

The DLQ handler is controlled with a series of pattern-matching and action keywords described here.

Figure 36 on page 1937 shows an example rule from a DLQ handler rules table.

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
 ACTION (RETRY) RETRY (3)
```

Figure 36. 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* | *)
The *ApplIdentityData* value of the message on the DLQ, specified in the message descriptor, MQMD.

APPLNAME (*PutApplName* | *)
The name of the application that issued the MQPUT or MQPUT1 call, as specified in the *PutApplName* field of the message descriptor, MQMD, of the message on the DLQ.

APPLTYPE (*PutApplType* | *)
The *PutApplType* value specified in the message descriptor, MQMD, of the message on the DLQ.

DESTQ (*QueueName* | *)
The name of the message queue for which the message is destined.

DESTQM (*QueueManagerName* | *)
The queue manager name for the message queue for which the message is destined.

FEEDBACK (*Feedback* | *)
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* | *)
The name that the sender of the message uses to describe the format of the message data.

MSGTYPE (*MsgType* | *)
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* | *)
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* | *)
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* | *)
The reply-to queue name specified in the message descriptor, MQMD, of the message on the DLQ.

REPLYQM (*QueueManagerName* | *)
The queue manager name of the reply-to queue specified in the REPLYQ keyword.

USERID (*UserIdentifier* | *)
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 |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|' ')
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 |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 |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)
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

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 nonblank character on a line to indicate that the rule continues from the first nonblank character in the next line. Use the minus sign (-) as the last nonblank 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

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

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)
```

The BSDS conversion utility (CSQJUCNV)

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, 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.

A version 2 BSDS can be used only by queue managers that have Version 8.0 new functions enabled with OPMODE. If the queue manager is in a queue-sharing group, all queue managers in the queue-sharing group must either have been started with OPMODE(NEWFUNC,800), or added into the queue-sharing group at Version 8.0, before the BSDS can be converted to version 2.

If the parameters provided specify that the queue manager is in a queue-sharing group, the utility checks that the queue managers are at the correct level, before allowing the conversion of the BSDS to proceed. For queue managers that are not in a queue-sharing group,the utility does not check that the queue manager has been started in Version 8.0 new function mode.

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. See Planning your logging environment, and the associated topics, for 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.
4. If you are using queue sharing groups, for each user ID that might use the CSQJUCNV utility GRANTs are needed for Db2 plans.

   Run CSQ45GEX in hlq.SCSQPROC to grant execute authority for the Db2 plans before you use CSQJUCNV, or CSQJUCNV receives SQL error -981 and REASON code 00C12219. See Preparing to migrate a single IBM MQ for z/OS queue manager for more information on CSQ45GEX.

- "Invoking the CSQJUCNV utility" on page 1944
- "Syntax, keywords, and parameters" on page 1944
- "Data definition (DD) statements" on page 1945

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.

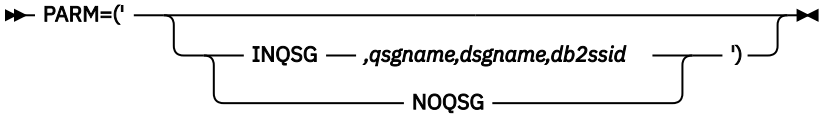
```
//CONVERT EXEC PGM=CSQJUCNV,REGION=32M,
//             PARM=('INQSG,qsgname,dsgname,db2ssid')
//STEPLIB  DD DSN=thlqual.SCSQAUTH,DISP=SHR
//         DD DSN=thlqual.SCSQANLE,DISP=SHR
//         DD DSN=db2qual.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=hlq.BSDS01,DISP=SHR
//SYSUT2   DD DSN=hlq.BSDS02,DISP=SHR
//SYSUT3   DD DSN=newhlq.BSDS01,DISP=OLD
//SYSUT4   DD DSN=newhlq.BSDS02,DISP=OLD
```

Figure 37. Sample JCL to invoke the CSQJUCNV utility

Sample JCL to run the utility is also provided in `thlqual.SCSQPROC(CSQ4BCNV)`.

Syntax, keywords, and parameters

BSDS conversion utility



PARM

This field must contain one of the following parameters to indicate whether the queue manager is a member of a queue-sharing group or not, followed by any function-specific parameters described in the following text:

INQSG
The queue manager that owns the BSDS is a member of a queue-sharing group. Specifying this parameter causes the utility to verify that all members of the queue-sharing group have either been started with OPMODE(NEWFUNC,800), or been added into the queue-sharing group at Version 8.0.

See Implementing the larger log Relative Byte Address for details on how you perform this task.

The utility terminates with a nonzero reason code, without writing anything to the output BSDS, if this condition is not met.

qsgname
The queue-sharing group name

dsgname
The Db2 data-sharing group name

db2ssid
The Db2 subsystem ID

NOQSG
The queue manager that owns the BSDS is not a member of a queue-sharing group.
The BSDS is converted, regardless of whether the queue manager has been started with OPMODE(NEWFUNC,800).

Attention: Do not specify this parameter for a queue manager that is a member of a queue-sharing group.

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.

The message security policy utility (CSQ0UTIL)

The IBM MQ 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 CSQ0UTIL 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=CSQ0UTIL,
//         PARM='ENVAR("_CEE_ENVFILE_S=DD:ENVARS") /'
//STEPLIB  DD DSN=thlqual.SCSQANLE,DISP=SHR
//         DD DSN=thlqual.SCSQAUTH,DISP=SHR
//ENVARS   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 CSQ0UTIL.

To use this utility you need connect authority to the queue manager and access to the queue SYSTEM.PROTECTION.POLICY.QUEUE. If command events have been enabled for the queue manager you 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

dspmqspl

Use the **dspmqspl** command to display a list of all policies and details of a named policy.

Syntax

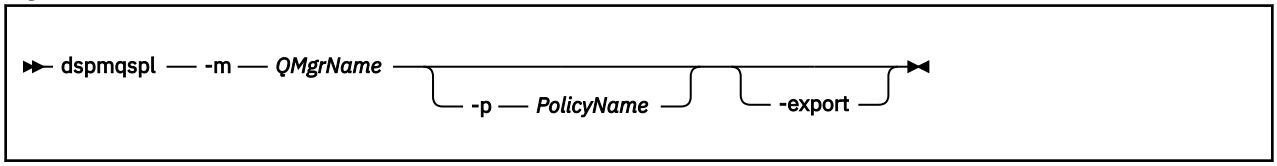


Table 105. dspmqspl command flags.

Command flag	Explanation
-m	Queue manager name (**mandatory**).
-p	Policy name.
-export	Adding this flag generates output which can easily be applied to a different queue manager.

Related reference
"SET POLICY" on page 1284

Use the MQSC command SET POLICY to set a security policy.

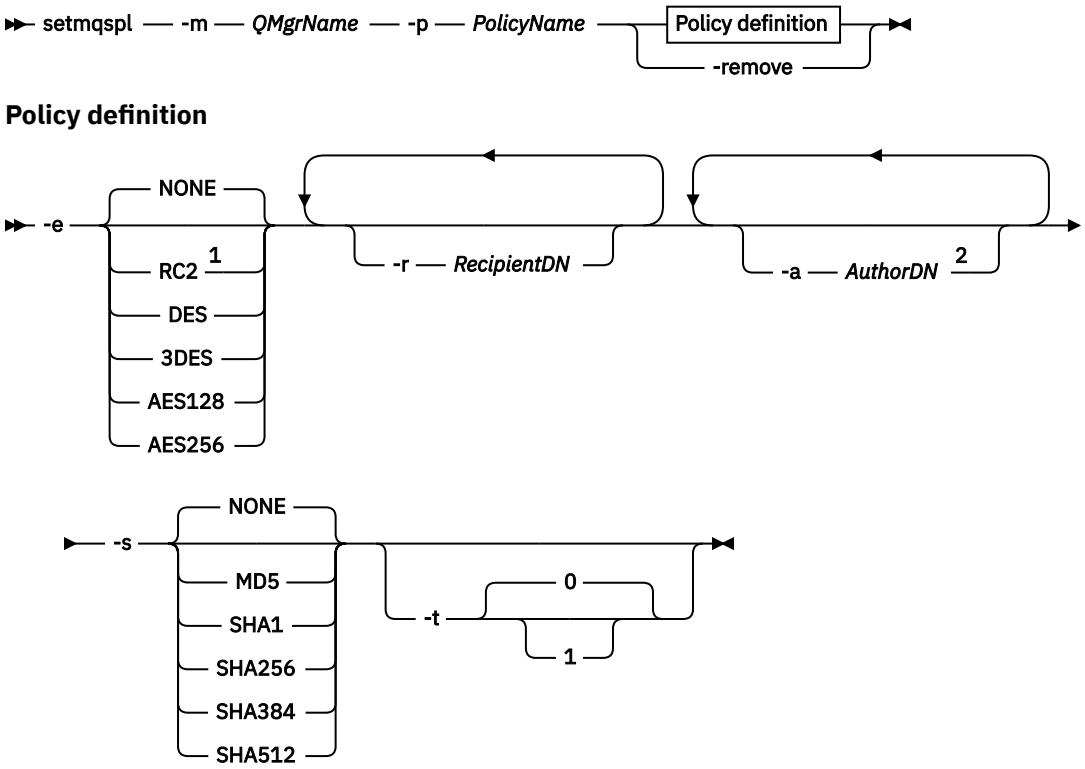
"DISPLAY POLICY" on page 1113
Use the MQSC command DISPLAY POLICY to display a security policy.

"setmqspl" on page 144
Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy.

setmqspl

Use the **setmqspl** command to define a new security policy, replace an already existing one, or remove an existing policy.

Syntax



Notes:

[1] If an encryption algorithm is selected, a recipient DN must also be provided.
[2] If an author DN is provided, a signing algorithm must also be selected.

Table 106. `setmqspl` command flags.

Command flag	Explanation
`-m`	Queue manager name. This flag is mandatory for all actions on security policies.
`-p`	Policy name. Set the policy name to the name of the queue you want the policy to apply to.

Table 106. `setmqspl` command flags. (continued)

Command flag	Explanation
`-s`	Digital signature algorithm. IBM MQ Advanced Message Security supports the following values: MD5, SHA1, SHA256, SHA384, and SHA512. All must be in uppercase. The default value is NONE **Important:** • For the SHA384 and SHA512 cryptographic hash functions, keys used for signing must be longer than 768 bits. • The name, or names, of the encryption algorithms must be in uppercase. • The signature algorithm cannot be NONE if the encryption algorithm is different from NONE.
`-e`	Digital encryption algorithm. IBM MQ Advanced Message Security supports the following encryption algorithms: RC2, DES, 3DES, AES128, AES256. The default value is NONE. **Important:** • The name of the encryption algorithm must be provided in uppercase • z/OS On z/OS encryption algorithm RC2 is not supported for confidentiality policies.
`-r`	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 **`-r`** flag. **Important:** • DN attribute names must be in uppercase. • Commas must be used as a name separators. • To avoid command interpreter errors, place quotation marks around the DNs. For example: `-r "CN=alice, O=ibm, C=US"`

Table 106. `setmqspl` command flags. (continued)	
Command flag	**Explanation**
`-a`	Signature DN that is validated during message retrieval. Only messages signed by a user with a DN provided are accepted during the retrieval. Signature DNs can be specified only if the signature algorithm is different from NONE. Multiple authors can be included. Each author needs to have a separate `-a` flag. **Important:** DN attribute name must be in uppercase.
`-t`	Toleration flag that indicates whether a policy that is associated with a queue can be ignored when an attempt to retrieve a message from the queue involves a message with no security policy set. Valid values include: • **0 (default)** Toleration flag off. • **1** Toleration flag on. 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.
`-remove`	Delete policy. If specified, only the `-p` flag remains valid.

Related reference

"SET POLICY" on page 1284
Use the MQSC command SET POLICY to set a security policy.

"DISPLAY POLICY" on page 1113
Use the MQSC command DISPLAY POLICY to display a security policy.

"dspmqspl" on page 73
Use the **dspmqspl** command to display a list of all policies and details of a named policy.

IBM MQ Administration Interface

Reference information for the IBM MQ Administration Interface (MQAI).

Related tasks

Using the MQAI to simplify the use of PCFs

MQAI calls

Reference information for MQAI calls.

A list of reference information for the MQAI.

There are two types of selector: *user selector* and *system selector*. These are described in "MQAI Selectors" on page 2028.

There are three types of call:

- Data-bag manipulation calls for configuring data bags:
  - "mqAddBag" on page 1951
  - "mqAddByteString" on page 1952
  - "mqAddByteStringFilter" on page 1954
  - "mqAddInquiry" on page 1956
  - "mqAddInteger" on page 1958
  - "mqAddInteger64" on page 1959
  - "mqAddIntegerFilter" on page 1961
  - "mqAddString" on page 1962
  - "mqAddStringFilter" on page 1964
  - "mqClearBag" on page 1970
  - "mqCountItems" on page 1971
  - "mqCreateBag" on page 1972
  - "mqDeleteBag" on page 1975
  - "mqDeleteItem" on page 1976
  - "mqInquireBag" on page 1984
  - "mqInquireByteString" on page 1986
  - "mqInquireByteStringFilter" on page 1988
  - "mqInquireInteger" on page 1991
  - "mqInquireInteger64" on page 1993
  - "mqInquireIntegerFilter" on page 1995
  - "mqInquireItemInfo" on page 1997
  - "mqInquireString" on page 2000
  - "mqInquireStringFilter" on page 2003
  - "mqSetByteString" on page 2008
  - "mqSetByteStringFilter" on page 2011
  - "mqSetInteger" on page 2013
  - "mqSetInteger64" on page 2015
  - "mqSetIntegerFilter" on page 2018
  - "mqSetString" on page 2020
  - "mqSetStringFilter" on page 2023
  - "mqTruncateBag" on page 2026
- Command calls for sending and receiving administration commands and PCF messages:
  - "mqBagToBuffer" on page 1966
  - "mqBufferToBag" on page 1968
  - "mqExecute" on page 1978
  - "mqGetBag" on page 1982
  - "mqPutBag" on page 2007
- Utility calls for handling blank-padded and null-terminated strings:
  - "mqPad" on page 2005
  - "mqTrim" on page 2025

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.

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'
```

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'
```

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).

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'
```

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'
```

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'
```

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'
```

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'
```

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 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_INHIBITIED
- 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_INHIBITIED
  - 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'
```

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'
```

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'
```

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'
```

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, Bag, 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.

***Bag* (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'
```

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 (nonnull) 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 (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 *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'
```

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'
```

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 (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 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'
```

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'
```

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'
```

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'
```

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 */
MQLONG   Selector;     /* Selector */
MQLONG   ItemIndex;    /* Item index */
MQLONG   BufferLength; /* Buffer length */
PMQBYTE  Buffer;       /* Buffer containing string */
MQLONG   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'
```

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'
```

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'
```

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'
```

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; MQRC_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, MQRC_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:

MQRC_HBAG_ERROR
Bag handle not valid.

MQRC_ITEM_COUNT_ERROR
ItemCount parameter not valid (value exceeds the number of user data items in the bag).

MQRC_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 in 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.

Example code

Here are some example uses of the mqExecute call.

The example shown in figure Figure 38 on page 2030 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)
```

Figure 38. Using mqExecute to create a local queue

The example shown in figure Figure 39 on page 2030 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)
```

Figure 39. Using mqExecute to inquire about queue attributes

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 Introduction to the IBM MQ Administration Interface (MQAI).

For sample programs, see Examples of using the MQAI.

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.

© Copyright IBM Corp. 2007, 2025

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®, 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. 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 (http://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: