

MQSeries



# Command Reference



MQSeries



# Command Reference

**Note!**

Before using this information and the product it supports, be sure to read the general information under Appendix I, "Notices" on page 209.

**Tenth edition (February 1998)**

This edition applies to the following products:

- MQSeries for AIX Version 5
- MQSeries for AS/400 Version 4 Release 2
- MQSeries for AT&T GIS UNIX Version 2 Release 2
- MQSeries for Digital OpenVMS AXP Version 2 Release 2
- MQSeries for Digital OpenVMS VAX Version 2 Release 2
- MQSeries for HP-UX Version 5
- MQSeries for MVS/ESA Version 1 Release 2
- MQSeries for OS/2 Warp Version 5
- MQSeries for SINIX and DC/OSx Version 2 Release 2
- MQSeries for SunOS Version 2 Release 2
- MQSeries for Sun Solaris Version 5
- MQSeries for Tandem NonStop Kernel Version 2 Release 2
- MQSeries for Windows NT Version 5
- MQSeries for Windows Version 2 Release 0
- MQSeries for Windows Version 2 Release 1

and to any subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page titled "Sending your comments to IBM". If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories,  
Information Development,  
Mail Point 095,  
Hursley Park,  
Winchester,  
Hampshire,  
England,  
SO21 2JN

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 1993, 1998. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About this book</b> . . . . .	vii
Who this book is for . . . . .	viii
How to use this book . . . . .	viii
MQSeries publications . . . . .	viii
MQSeries cross-platform publications . . . . .	viii
MQSeries platform-specific publications . . . . .	x
MQSeries Level 1 product publications . . . . .	xii
Softcopy books . . . . .	xii
MQSeries information available on the Internet . . . . .	xiii
<b>Summary of changes</b> . . . . .	xv
Changes for this edition . . . . .	xv
Changes for the ninth edition . . . . .	xv
<b>Chapter 1. Using MQSeries Commands</b> . . . . .	1
Rules for using MQSeries commands . . . . .	1
Rules for naming MQSeries objects . . . . .	5
How to read syntax diagrams . . . . .	8
<b>Chapter 2. The MQSeries commands</b> . . . . .	9
ALTER CHANNEL . . . . .	11
ALTER NAMELIST . . . . .	28
ALTER PROCESS . . . . .	29
ALTER QALIAS . . . . .	31
ALTER QLOCAL . . . . .	33
ALTER QMGR . . . . .	41
ALTER QMODEL . . . . .	45
ALTER QREMOTE . . . . .	53
ALTER SECURITY . . . . .	56
ALTER STGCLASS . . . . .	57
ALTER TRACE . . . . .	59
ARCHIVE LOG . . . . .	60
CLEAR QLOCAL . . . . .	62
DEFINE BUFFPOOL . . . . .	63
DEFINE CHANNEL . . . . .	64
DEFINE MAXSMSGS . . . . .	82
DEFINE NAMELIST . . . . .	83
DEFINE PROCESS . . . . .	85
DEFINE PSID . . . . .	88
DEFINE QALIAS . . . . .	89
DEFINE QLOCAL . . . . .	92
DEFINE QMODEL . . . . .	100
DEFINE QREMOTE . . . . .	108
DEFINE STGCLASS . . . . .	112
DELETE CHANNEL . . . . .	114
DELETE NAMELIST . . . . .	115
DELETE PROCESS . . . . .	116
DELETE QALIAS . . . . .	117
DELETE QLOCAL . . . . .	118
DELETE QMODEL . . . . .	119

DELETE QREMOTE	120
DELETE STGCLASS	121
DISPLAY CHANNEL	122
DISPLAY CHSTATUS	126
DISPLAY CMDSERV	133
DISPLAY DQM	134
DISPLAY MAXSMSGS	135
DISPLAY NAMELIST	136
DISPLAY PROCESS	137
DISPLAY QMGR	138
DISPLAY QUEUE	141
DISPLAY SECURITY	146
DISPLAY STGCLASS	147
DISPLAY THREAD	148
DISPLAY TRACE	149
DISPLAY USAGE	151
PING CHANNEL	152
PING QMGR	153
RECOVER BSDS	154
REFRESH SECURITY	155
RESET CHANNEL	156
RESET TPIPE	157
RESOLVE CHANNEL	159
RESOLVE INDOUBT	160
RVERIFY SECURITY	161
START CHANNEL	162
START CHINIT	163
START CMDSERV	164
START LISTENER	165
START QMGR	166
START TRACE	167
STOP CHANNEL	171
STOP CHINIT	172
STOP CMDSERV	173
STOP LISTENER	174
STOP QMGR	175
STOP TRACE	176
<b>Appendix A. Command summary</b>	<b>179</b>
<b>Appendix B. How to issue MQSC commands on Digital OpenVMS</b>	<b>185</b>
Using the runmqsc command	185
<b>Appendix C. How to issue MQSC commands on MVS/ESA</b>	<b>189</b>
Directing the command to the correct queue manager	190
<b>Appendix D. How to issue MQSC commands on OS/2 Warp</b>	<b>191</b>
Using the runmqsc command	191
<b>Appendix E. How to issue MQSC commands on OS/400</b>	<b>195</b>
Example OS/400 MQSeries command file and report	195
<b>Appendix F. How to issue MQSC commands on Tandem NSK</b>	<b>197</b>
Using the runmqsc command	197

<b>Appendix G. How to issue MQSC commands on UNIX systems</b> . . . . .	201
Using the runmqsc command . . . . .	201
<b>Appendix H. How to issue MQSC commands on Windows NT</b> . . . . .	205
Using the runmqsc command . . . . .	205
<b>Appendix I. Notices</b> . . . . .	209
Trademarks . . . . .	209
<b>Index</b> . . . . .	211

---

## Figures

1. Example command input file for Digital OpenVMS . . . . .	186
2. Example report file from Digital OpenVMS . . . . .	187
3. Example command input file for OS/2 Warp . . . . .	192
4. Example report file from OS/2 Warp . . . . .	193
5. Example command input file for OS/400 . . . . .	195
6. Example report file from OS/400 . . . . .	196
7. Example command input file for Tandem NSK . . . . .	198
8. Example report file from Tandem NSK . . . . .	199
9. Example command input file for UNIX systems . . . . .	202
10. Example report file from UNIX systems . . . . .	203
11. Example command input file for Windows NT . . . . .	206
12. Example report file from Windows NT . . . . .	207

## Figures



---

## About this book

This book describes the MQSeries commands (MQSC), which system operators and administrators can use to manage queue managers on the following MQSeries platforms:

- Digital OpenVMS
- MVS/ESA
- OS/2 Warp
- OS/400
- Tandem NSK
- UNIX operating systems
- Windows NT

The commands are listed in alphabetic order in Chapter 2, “The MQSeries commands” on page 9. A matrix at the start of each command description identifies platforms on which the command is valid. Table 2 on page 9 gives a list of all MQSC commands.

MQSeries for Windows supports a subset of the MQSC commands. This subset is shown in Table 2 on page 9. For a description of the syntax of each command, see the *MQSeries for Windows Command Reference* (this is an online book shipped with MQSeries for Windows).

For information about building commands on your platform, see the following sections:

- Appendix B, “How to issue MQSC commands on Digital OpenVMS” on page 185
- Appendix C, “How to issue MQSC commands on MVS/ESA” on page 189
- Appendix D, “How to issue MQSC commands on OS/2 Warp” on page 191
- Appendix E, “How to issue MQSC commands on OS/400” on page 195
- Appendix F, “How to issue MQSC commands on Tandem NSK” on page 197
- Appendix G, “How to issue MQSC commands on UNIX systems” on page 201
- Appendix H, “How to issue MQSC commands on Windows NT” on page 205

The term “UNIX systems” is used to denote the following UNIX operating systems:

- AIX
- AT&T\*\* GIS UNIX<sup>1</sup>
- HP-UX\*\*
- SINIX\*\* and DC/OSx\*\*
- SunOS\*\*
- Sun Solaris\*\*

---

<sup>1</sup> This platform has become NCR UNIX SVR4 MP-RAS, R3.0

### Who this book is for

This book is intended for system programmers, system administrators, and system operators.

---

### How to use this book

First read those sections of the MQSeries *Administration Guide*, *System Management Guide*, or *System Administration* book for your platform that relate to the task you want to perform. When you have decided which commands you need to use, check their syntax in this book.

The syntax of the MQSeries commands is represented in *syntax diagrams*. How to read these diagrams is explained in “How to read syntax diagrams” on page 8. The parameters for each command are listed in the following order in the syntax diagrams:

- Parameters that are required are listed first, in alphabetic order.
- Parameters that are optional follow, again in alphabetic order.

---

### MQSeries publications

This section describes the documentation available for all current MQSeries products.

### MQSeries cross-platform publications

Most of these publications, which are sometimes referred to as the MQSeries “family” books, apply to all MQSeries Level 2 products. The latest MQSeries Level 2 products are:

- MQSeries for AIX V5.0
- MQSeries for AS/400 V4R2
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Digital OpenVMS V2.2
- MQSeries for HP-UX V5.0
- MQSeries for MVS/ESA V1.2
- MQSeries for OS/2 Warp V5.0
- MQSeries for SINIX and DC/OSx V2.2
- MQSeries for SunOS V2.2
- MQSeries for Sun Solaris V5.0
- MQSeries for Tandem NonStop Kernel V2.2
- MQSeries Three Tier
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT V5.0

Any exceptions to this general rule are indicated. (Publications that support the MQSeries Level 1 products are listed in “MQSeries Level 1 product publications” on page xii. For a functional comparison of the Level 1 and Level 2 MQSeries products, see the *MQSeries Planning Guide*.)

**MQSeries Brochure**

The *MQSeries Brochure*, G511-1908, gives a brief introduction to the benefits of MQSeries. It is intended to support the purchasing decision, and describes some authentic customer use of MQSeries.

**MQSeries: An Introduction to Messaging and Queuing**

*MQSeries: An Introduction to Messaging and Queuing*, GC33-0805, describes briefly what MQSeries is, how it works, and how it can solve some classic interoperability problems. This book is intended for a more technical audience than the *MQSeries Brochure*.

**MQSeries Planning Guide**

The *MQSeries Planning Guide*, GC33-1349, describes some key MQSeries concepts, identifies items that need to be considered before MQSeries is installed, including storage requirements, backup and recovery, security, and migration from earlier releases, and specifies hardware and software requirements for every MQSeries platform.

**MQSeries Intercommunication**

The *MQSeries Intercommunication* book, SC33-1872, defines the concepts of distributed queuing and explains how to set up a distributed queuing network in a variety of MQSeries environments. In particular, it demonstrates how to (1) configure communications to and from a representative sample of MQSeries products, (2) create required MQSeries objects, and (3) create and configure MQSeries channels. The use of channel exits is also described.

**MQSeries Clients**

The *MQSeries Clients* book, GC33-1632, describes how to install, configure, use, and manage MQSeries client systems.

**MQSeries System Administration**

The *MQSeries System Administration* book, SC33-1873, supports day-to-day management of local and remote MQSeries objects. It includes topics such as security, recovery and restart, transactional support, problem determination, the dead-letter queue handler, and the MQSeries links for Lotus Notes\*\*. It also includes the syntax of the MQSeries control commands.

This book applies to the following MQSeries products only:

- MQSeries for AIX V5.0
- MQSeries for HP-UX V5.0
- MQSeries for OS/2 Warp V5.0
- MQSeries for Sun Solaris V5.0
- MQSeries for Windows NT V5.0

**MQSeries Command Reference**

The *MQSeries Command Reference*, SC33-1369, contains the syntax of the MQSC commands, which are used by MQSeries system operators and administrators to manage MQSeries objects.

**MQSeries Programmable System Management**

The *MQSeries Programmable System Management* book, SC33-1482, provides both reference and guidance information for users of MQSeries events, programmable command formats (PCFs), and installable services.

## MQSeries publications

### MQSeries Messages

The *MQSeries Messages* book, GC33-1876, which describes “AMQ” messages issued by MQSeries, applies to these MQSeries products only:

- MQSeries for AIX V5.0
- MQSeries for HP-UX V5.0
- MQSeries for OS/2 Warp V5.0
- MQSeries for Sun Solaris V5.0
- MQSeries for Windows NT V5.0
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1

This book is available in softcopy only.

### MQSeries Application Programming Guide

The *MQSeries Application Programming Guide*, SC33-0807, provides guidance information for users of the message queue interface (MQI). It describes how to design, write, and build an MQSeries application. It also includes full descriptions of the sample programs supplied with MQSeries.

### MQSeries Application Programming Reference

The *MQSeries Application Programming Reference*, SC33-1673, provides comprehensive reference information for users of the MQI. It includes: data-type descriptions; MQI call syntax; attributes of MQSeries objects; return codes; constants; and code-page conversion tables.

### MQSeries Application Programming Reference Summary

The *MQSeries Application Programming Reference Summary*, SX33-6095, summarizes the information in the *MQSeries Application Programming Reference* manual.

### MQSeries Using C++

*MQSeries Using C++*, SC33-1877, provides both guidance and reference information for users of the MQSeries C++ programming-language binding to the MQI. MQSeries C++ is supported by V5.0 of MQSeries for AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, and by MQSeries clients supplied with those products and installed in the following environments:

- AIX
- HP-UX
- OS/2
- Sun Solaris
- Windows NT
- Windows 3.1
- Windows 95

MQSeries C++ is also supported by MQSeries for AS/400 V4R2.

## MQSeries platform-specific publications

Each MQSeries product is documented in at least one platform-specific publication, in addition to the MQSeries family books.

### MQSeries for AIX

*MQSeries for AIX V5.0 Quick Beginnings*, GC33-1867

**MQSeries for AS/400**

*MQSeries for AS/400 Version 4 Release 2 Licensed Program Specifications, GC33-1958*

*MQSeries for AS/400 Version 4 Release 2 Administration Guide, GC33-1956*

*MQSeries for AS/400 Version 4 Release 2 Application Programming Reference (RPG), SC33-1957*

**MQSeries for AT&T GIS UNIX**

*MQSeries for AT&T GIS UNIX Version 2.2 System Management Guide, SC33-1642*

**MQSeries for Digital OpenVMS**

*MQSeries for Digital OpenVMS Version 2.2 System Management Guide, GC33-1791*

**MQSeries for HP-UX**

*MQSeries for HP-UX V5.0 Quick Beginnings, GC33-1869*

**MQSeries for MVS/ESA**

*MQSeries for MVS/ESA Version 1 Release 2 Licensed Program Specifications, GC33-1350*

*MQSeries for MVS/ESA Version 1 Release 2 Program Directory*

*MQSeries for MVS/ESA Version 1 Release 2 System Management Guide, SC33-0806*

*MQSeries for MVS/ESA Version 1 Release 2 Messages and Codes, GC33-0819*

*MQSeries for MVS/ESA Version 1 Release 2 Problem Determination Guide, GC33-0808*

**MQSeries for OS/2 Warp**

*MQSeries for OS/2 Warp V5.0 Quick Beginnings, GC33-1868*

**MQSeries link for R/3**

*MQSeries link for R/3 Version 1.0 User's Guide, GC33-1934*

**MQSeries for SINIX and DC/OSx**

*MQSeries for SINIX and DC/OSx Version 2.2 System Management Guide, GC33-1768*

**MQSeries for SunOS**

*MQSeries for SunOS Version 2.2 System Management Guide, GC33-1772*

**MQSeries for Sun Solaris**

*MQSeries for Sun Solaris V5.0 Quick Beginnings, GC33-1870*

**MQSeries for Tandem NonStop Kernel**

*MQSeries for Tandem NonStop Kernel Version 2.2 System Management Guide, GC33-1893*

## MQSeries publications

### MQSeries Three Tier

*MQSeries Three Tier Administration Guide*, SC33-1451  
*MQSeries Three Tier Reference Summary*, SX33-6098  
*MQSeries Three Tier Application Design*, SC33-1636  
*MQSeries Three Tier Application Programming*, SC33-1452

### MQSeries for Windows

*MQSeries for Windows Version 2.0 User's Guide*, GC33-1822  
*MQSeries for Windows Version 2.1 User's Guide*, GC33-1965

### MQSeries for Windows NT

*MQSeries for Windows NT V5.0 Quick Beginnings*, GC33-1871

## MQSeries Level 1 product publications

For information about the MQSeries Level 1 products, see the following publications:

*MQSeries: Concepts and Architecture*, GC33-1141  
*MQSeries Version 1 Products for UNIX Operating Systems Messages and Codes*, SC33-1754  
*MQSeries for SCO UNIX Version 1.4 User's Guide*, SC33-1378  
*MQSeries for UnixWare Version 1.4.1 User's Guide*, SC33-1379  
*MQSeries for VSE/ESA Version 1 Release 4 Licensed Program Specifications*, GC33-1483  
*MQSeries for VSE/ESA Version 1 Release 4 User's Guide*, SC33-1142

## Softcopy books

Most of the MQSeries books are supplied in both hardcopy and softcopy formats.

### BookManager format

The MQSeries library is supplied in IBM BookManager format on a variety of online library collection kits, including the *Transaction Processing and Data* collection kit, SK2T-0730. You can view the softcopy books in IBM BookManager format using the following IBM licensed programs:

BookManager READ/2  
BookManager READ/6000  
BookManager READ/DOS  
BookManager READ/MVS  
BookManager READ/VM  
BookManager READ for Windows

### PostScript format

The MQSeries library is provided in PostScript (.PS) format with many MQSeries products, including all MQSeries V5.0 products. Books in PostScript format can be printed on a PostScript printer or viewed with a suitable viewer.

### HTML format

The MQSeries documentation is provided in HTML format with these MQSeries products:

- MQSeries for AIX V5.0
- MQSeries for HP-UX V5.0
- MQSeries for OS/2 Warp V5.0
- MQSeries for Sun Solaris V5.0
- MQSeries for Windows NT V5.0

The MQSeries books are also available from the MQSeries product family Web site:

<http://www.software.ibm.com/ts/mqseries/>

### Information Presentation Facility (IPF) format

In the OS/2 environment, the MQSeries documentation is supplied in IBM IPF format on the MQSeries product CD-ROM.

### Windows Help format

The *MQSeries for Windows User's Guide* is provided in Windows Help format with MQSeries for Windows Version 2.0 and MQSeries for Windows Version 2.1.

---

## MQSeries information available on the Internet

### MQSeries web site

The MQSeries product family Web site is at:

<http://www.software.ibm.com/ts/mqseries/>

By following links from this Web site you can:

- Obtain latest information about the MQSeries product family.
- Access the MQSeries books in HTML format.
- Download MQSeries SupportPacs.





---

## Summary of changes

This section lists the changes that have been made to this book since previous editions. Changes for this edition are marked with vertical bars in the left-hand margin.

---

### Changes for this edition

Changes for edition number SC33-1369-09 include:

- The book has been updated to contain information about the following changes to the MQSeries for AS/400 Version 4.2 product:
  - You can now specify more than one message, send, and receive exit.
  - You can now use distribution lists.
  - You can now request a channel heartbeat.
  - You can now define fast channels for nonpersistent messages.
  - You can now write exit programs to define receiver and client-connection channels automatically.
- The book has been updated to contain information about MQSeries for Tandem NonStop Kernel Version 2.2.
- The rules for reading syntax diagrams have changed. These are described in “How to read syntax diagrams” on page 8.

---

### Changes for the ninth edition

Changes for edition number SC33-1369-08 include:

- The book has been updated to contain information about the following new releases of MQSeries products:
  - MQSeries for AIX Version 5
  - MQSeries for HP-UX Version 5
  - MQSeries for MVS/ESA Version 1.2
  - MQSeries for OS/2 Warp Version 5
  - MQSeries for OS/400 Version 3.7
  - MQSeries for Sun Solaris Version 5
  - MQSeries for Windows NT Version 5
- The following products have been added to the MQSeries family:
  - MQSeries for Digital OpenVMS
  - MQSeries for Windows
- The maximum message length has been increased to 100 MB for some platforms.
- The following new commands have been added:
  - DISPLAY QALIAS
  - DISPLAY QLOCAL
  - DISPLAY QMODEL
  - DISPLAY QREMOTE

## Summary of changes

- The following objects have been changed for some platforms:
  - Channels
    - You can now specify more than one message, send, and receive exit.
    - You can now use SPX as your transport protocol.
    - You can now request a channel heartbeat.
    - You can now define fast channels for nonpersistent messages.
  - Queues
    - You can now build an index to expedite **MQGET** operations on queues (MVS/ESA).
    - You can now use distribution lists (AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT).
    - The process attribute is now optional for transmission queues (AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT).
  - Queue managers
    - You can now write exit programs to define receiver and client-connection channels automatically.
- The rules for entering commands have been changed, and online help is available. This is described in:
  - Appendix D, “How to issue MQSC commands on OS/2 Warp” on page 191
  - Appendix G, “How to issue MQSC commands on UNIX systems” on page 201
  - Appendix H, “How to issue MQSC commands on Windows NT” on page 205

---

## Chapter 1. Using MQSeries Commands

MQSeries commands (MQSC) provide a uniform method of issuing human readable commands across MQSeries platforms. For information about *programmable command format* (PCF) commands (not available on MVS/ESA) see the *MQSeries Programmable System Management* manual.

This chapter discusses:

- “Rules for using MQSeries commands”
- “Rules for naming MQSeries objects” on page 5
- “How to read syntax diagrams” on page 8

The general format of the commands is shown in Chapter 2, “The MQSeries commands” on page 9.

For information about how to issue the commands on your MQSeries platform, see:

- Appendix B, “How to issue MQSC commands on Digital OpenVMS” on page 185
- Appendix C, “How to issue MQSC commands on MVS/ESA” on page 189
- Appendix D, “How to issue MQSC commands on OS/2 Warp” on page 191
- Appendix E, “How to issue MQSC commands on OS/400” on page 195
- Appendix F, “How to issue MQSC commands on Tandem NSK” on page 197
- Appendix G, “How to issue MQSC commands on UNIX systems” on page 201
- Appendix H, “How to issue MQSC commands on Windows NT” on page 205

For information about using MQSC commands on MQSeries for Windows, see the *MQSeries for Windows User's Guide*.

---

### Rules for using MQSeries commands

You should observe the following rules when using MQSeries commands:

- Each command starts with a primary keyword (a verb), and this is followed by a secondary keyword (a noun). This is then followed by the name or generic name of the object (in parentheses) if there is one, which there is on most commands. Following that, keywords can occur in any order; if a keyword has a corresponding value, the value must occur directly after the keyword to which it relates.

**Note:** On MVS/ESA, the secondary keyword does not have to be second.

- Keywords, parentheses, and values can be separated by any number of blanks and commas. A comma shown in the syntax diagrams can always be replaced by one or more blanks. There must be at least one blank immediately preceding each keyword (after the primary keyword) except on MVS/ESA.
- Any number of blanks can occur at the beginning or end of the command, and between keywords, punctuation, and values. For example, the following command is valid:

```
ALTER QLOCAL ('Account' )          TRIGDPTH ( 1)
```

Blanks within a pair of quotation marks are significant.

## Rules for using commands

There must be at least one blank immediately preceding each keyword (after the primary keyword) except on MVS/ESA.

- Additional commas can appear anywhere where blanks are allowed and are treated as if they were blanks (unless, of course, they are inside quoted strings).
- Repeated keywords are not allowed. Repeating a keyword with its 'NO' version, as in REPLACE NOREPLACE, is also not allowed.
- Strings that contain blanks, lowercase characters or special characters other than:
  - Period (.)
  - Forward slash (/)
  - Underscore (\_)
  - Percent sign (%)

must be enclosed in single quotation marks, unless they are:

- Issued from the MQSeries for MVS/ESA operations and control panels
- Generic names ending with an asterisk (on OS/400 these must be enclosed in single quotation marks)
- A single asterisk (for example, TRACE(\*)) (on OS/400 these must be enclosed in single quotation marks)
- A range specification containing a colon (for example, CLASS(01:03))

If the string itself contains a quotation mark, the quotation mark is represented by two single quotation marks. Lowercase characters not contained within quotation marks are folded to uppercase.

- A string containing no characters (that is, two single quotation marks with no space in between) is not valid.
- A left parenthesis followed by a right parenthesis, with no significant information in between, for example

NAME ( )

is not valid except where specifically noted.

- Keywords are not case sensitive – AltER, alter, and ALTER are all acceptable. Names that are not contained within quotation marks are converted to uppercase.
- Synonyms are defined for some keywords. For example, DEF is always a synonym for DEFINE, so DEF QLOCAL is valid. Synonyms are not, however, just minimum strings; DEFI is not a valid synonym for DEFINE.

**Note:** There is no synonym for the DELETE keyword. This is to avoid accidental deletion of objects when using DEF, the synonym for DEFINE.

## Characters with special meanings

The following characters have special meaning when you build MQSC commands:

	Blanks are used as separators. Multiple blanks are equivalent to a single blank, except in strings that have quotation marks (') round them.
,	Commas are used as separators. Multiple commas are equivalent to a single comma, except in strings that have quotation marks (') round them.
'	A single quotation mark indicates the beginning or end of a string. MQSeries leaves all characters that have quotation marks round them exactly as they are entered. The containing quotation marks are not included when calculating the length of the string.
''	Two quotation marks together inside a string are treated by MQSeries as one quotation mark, and the string is not terminated. The double quotation marks are treated as one character when calculating the length of the string.
(	An open parenthesis indicates the beginning of a parameter list.
)	A close parenthesis indicates the end of a parameter list.
:	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 whose names 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

If you build the commands into a script, as when using:

- The CSQINP1, CSQINP2, and CSQINPX initialization data sets or the CSQUTIL batch utility on MVS/ESA
- The STRMQMMQSC command on OS/400
- The runmqsc command on Digital OpenVMS, OS/2 Warp, Tandem NSK, UNIX systems, and Windows NT

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 MVS/ESA, 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, OS/2 Warp, Sun Solaris, and Windows NT, each line can be of any length up to the maximum allowed for your platform.
  - On other UNIX systems, Digital OpenVMS, and OS/400, each line can be of any length up to and including 80 characters.
  - On Tandem NSK, each line can be of any length up to and including 72 characters.

## Rules for using commands

- 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 keyword (except on MVS/ESA where this is not necessary).

Either of these can occur within a keyword, data value, or quoted string. 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 the *MQSeries Programmable System Management* manual.)

- + and - values used at the ends of lines are discarded when the command is reassembled into a single string.
- On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT you can use a semicolon character (;) to terminate a command, even if you have entered a + character at the end of the previous line. You can also use the semicolon in the same way on MVS/ESA 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.

---

## Rules for naming MQSeries objects

MQSeries queue, process, namelist, channel, and storage class objects exist in separate object *name spaces*, and so objects from each type can all have the same name. However, an object cannot have the same name as any other object in the same name space. (For example, a local queue cannot have the same name as a model queue.) Names in MQSeries are case sensitive; however, you should remember that lowercase characters that are not contained within quotation marks are folded to uppercase.

The character set that can be used for naming all MQSeries objects is as follows:

- Uppercase A–Z
- Lowercase a–z (however, on systems using EBCDIC Katakana you cannot use lowercase characters, and there are also restrictions on the use of lowercase letters for MVS console support)
- Numerics 0–9
- Period (.)
- Forward slash (/)
- Underscore (\_)
- Percent sign (%). The percent sign (%) is a special character to RACF. If you are using RACF as the external security manager for MQSeries for MVS/ESA, you should not use % in object names. If you do, these names are not included in any security checks when RACF generic profiles are used.

### Notes:

1. Leading or embedded blanks are not allowed.
2. You should avoid using names with leading or trailing underscores, because they cannot be handled by the MQSeries for MVS/ESA operations and control panels.
3. Any name that is less than the full field length can be padded to the right with blanks. All short names that are returned by the queue manager are always padded to the right with blanks.
4. Any structure to the names (for example, the use of the period or underscore) is not significant to the queue manager.

### Queue names

Queues can have names up to 48 characters long.

#### Reserved queue names

Names that start with "SYSTEM." are reserved for queues defined by the queue manager. You can use the ALTER or DEFINE REPLACE commands to change these queue definitions to suit your installation. On MVS/ESA, you cannot delete any of the reserved queues, except for those with names starting SYSTEM.ADMIN. The following names are defined for MQSeries:

SYSTEM.ADMIN.CHANNEL.EVENT	Queue for channel events
SYSTEM.ADMIN.COMMAND.QUEUE	Queue to which PCF command messages are sent (not for MVS/ESA)
SYSTEM.ADMIN.PERFM.EVENT	Queue for performance events
SYSTEM.ADMIN.QMGR.EVENT	Queue for queue-manager events
SYSTEM.CHANNEL.COMMAND	Queue used for distributed queuing on MVS/ESA using CICS
SYSTEM.CHANNEL.INITQ	Queue used for distributed queuing (without CICS on MVS/ESA)
SYSTEM.CHANNEL.REPLY.INFO	Queue used for distributed queuing on MVS/ESA without CICS
SYSTEM.CHANNEL.SEQNO	Queue used for distributed queuing on MVS/ESA using CICS
SYSTEM.CHANNEL.SYNCQ	Queue used for distributed queuing (without CICS on MVS/ESA)
SYSTEM.CICS.INITIATION.QUEUE	Queue used for triggering (not for MVS/ESA)
SYSTEM.COMMAND.INPUT	Queue to which command messages are sent on MVS/ESA
SYSTEM.COMMAND.REPLY.MODEL	Model queue definition for command replies (for MVS/ESA)
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter queue (not for MVS/ESA)
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue definition
SYSTEM.DEFAULT.INITIATION.QUEUE	Queue used for distributed queuing (not for MVS/ESA)
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue definition
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue definition
SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue definition
SYSTEM.MQSC.REPLY.QUEUE	Model queue definition for MQSC command replies (not for MVS/ESA)



## Other object names

Processes and namelists can have names up to 48 bytes long. Channels can have names up to 20 bytes long. Storage classes can have names up to 8 bytes long.

### Reserved object names

Names that start with "SYSTEM." are reserved for objects defined by the queue manager. You can use the ALTER or DEFINE REPLACE commands to change these object definitions to suit your installation. On MVS/ESA, you cannot delete any of the reserved objects. The following names are defined for MQSeries:

SYSTEM.AUTO.RECEIVER	Default receiver channel for auto definition (not for AT&T GIS UNIX, Digital OpenVMS, MVS/ESA, SINIX and DC/OSx, SunOS, or Tandem NSK)
SYSTEM.AUTO.SVRCONN	Default server-connection channel for auto definition (not for AT&T GIS UNIX, Digital OpenVMS, MVS/ESA, SINIX and DC/OSx, SunOS, or Tandem NSK)
SYSTEM.DEF.CLNTCONN	Default client-connection channel definition
SYSTEM.DEF.RECEIVER	Default receiver channel definition
SYSTEM.DEF.REQUESTER	Default requester channel definition
SYSTEM.DEF.SENDER	Default sender channel definition
SYSTEM.DEF.SERVER	Default server channel definition
SYSTEM.DEF.SVRCONN	Default server-connection channel definition
SYSTEM.DEFAULT.NAMELIST	Default namelist definition (MVS/ESA only)
SYSTEM.DEFAULT.PROCESS	Default process definition
SYSTEMST	Default storage class definition (MVS/ESA only)

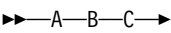
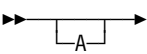
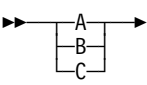
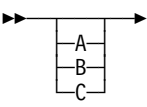
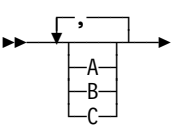
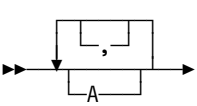
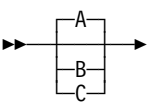
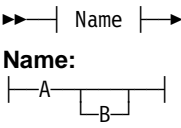
## How to read syntax diagrams

This chapter contains syntax diagrams (sometimes referred to as “railroad” diagrams).

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

Other conventions used in syntax diagrams are:

Table 1. How to read syntax diagrams

Convention	Meaning
	You must specify values A, B, and C. Required values are shown on the main line of a syntax diagram.
	You may specify value A. Optional values are shown below the main line of a syntax diagram.
	Values A, B, and C are alternatives, one of which you must specify.
	Values A, B, and C are alternatives, one of which you may specify.
	You may 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 may specify value A multiple times. The separator in this example is optional.
	Values A, B, and C are alternatives, one of which you may specify. If you specify none of the values shown, the default A (the value shown above the main line) is used.
	The syntax fragment Name is shown separately from the main syntax diagram.
Punctuation and uppercase values	Specify exactly as shown.
Lowercase values (for example, name)	Supply your own text in place of the <i>name</i> variable.

## Chapter 2. The MQSeries commands

This chapter describes all the MQSeries commands (MQSC) that can be issued by operators and administrators. Table 2 shows which commands can be issued on each MQSeries platform:

Command	Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT	Windows <sup>1</sup>
ALTER CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	√
ALTER NAMELIST		√						
ALTER PROCESS	√	√	√	√	√	√	√	
ALTER QALIAS	√	√	√	√	√	√	√	√
ALTER QLOCAL	√	√	√	√	√	√	√	√
ALTER QMGR	√	√	√	√	√	√	√	√
ALTER QMODEL	√	√	√	√	√	√	√	√
ALTER QREMOTE	√	√	√	√	√	√	√	√
ALTER SECURITY		√						
ALTER STGCLASS		√						
ALTER TRACE		√						
ARCHIVE LOG		√						
CLEAR QLOCAL	√		√	√	√	√	√	√
DEFINE BUFFPOOL		√						
DEFINE CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	√
DEFINE MAXSMGS	√ <sup>3</sup>	√	√ <sup>3</sup>	√ <sup>3</sup>	√ <sup>3</sup>	√ <sup>3</sup>	√ <sup>3</sup>	
DEFINE NAMELIST		√						
DEFINE PROCESS	√	√	√	√	√	√	√	
DEFINE PSID		√						
DEFINE QALIAS	√	√	√	√	√	√	√	√
DEFINE QLOCAL	√	√	√	√	√	√	√	√
DEFINE QMODEL	√	√	√	√	√	√	√	√
DEFINE QREMOTE	√	√	√	√	√	√	√	√
DEFINE STGCLASS		√						
DELETE CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	√
DELETE NAMELIST		√						
DELETE PROCESS	√	√	√	√	√	√	√	
DELETE QALIAS	√	√	√	√	√	√	√	√
DELETE QLOCAL	√	√	√	√	√	√	√	√
DELETE QMODEL	√	√	√	√	√	√	√	√
DELETE QREMOTE	√	√	√	√	√	√	√	√
DELETE STGCLASS		√						
DISPLAY CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	√ <sup>4</sup>
DISPLAY CHSTATUS	√	√ <sup>2</sup>	√	√	√	√	√	
DISPLAY CMDSERV		√						
DISPLAY DQM		√ <sup>2</sup>						
DISPLAY MAXSMGS	√ <sup>3</sup>	√	√ <sup>3</sup>	√ <sup>3</sup>	√ <sup>3</sup>	√ <sup>3</sup>	√ <sup>3</sup>	
DISPLAY NAMELIST		√						
DISPLAY PROCESS	√	√	√	√	√	√	√	
DISPLAY QALIAS <sup>5</sup>		√	√	√		√ <sup>6</sup>	√	
DISPLAY QLOCAL <sup>5</sup>		√	√	√		√ <sup>6</sup>	√	
DISPLAY QMGR	√	√	√	√	√	√	√	√ <sup>4</sup>
DISPLAY QMODEL <sup>5</sup>		√	√	√		√ <sup>6</sup>	√	

## MQSeries commands

Table 2 (Page 2 of 2). MQSeries operator and administrator commands

Command	Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT	Windows <sup>1</sup>
DISPLAY QREMOTE <sup>5</sup>		√	√	√		√ <sup>6</sup>	√	
DISPLAY QUEUE	√	√	√	√	√	√	√	√ <sup>4</sup>
DISPLAY SECURITY		√						
DISPLAY STGCLASS		√						
DISPLAY THREAD		√						
DISPLAY TRACE		√						
DISPLAY USAGE		√						
PING CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	
PING QMGR	√		√	√	√	√	√	
RECOVER BSDS		√						
REFRESH SECURITY		√						
RESET CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	√
RESET TPIPE		√						
RESOLVE CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	√
RESOLVE INDOUBT		√						
RVERIFY SECURITY		√						
START CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	√
START CHINIT	√	√ <sup>2</sup>	√	√		√	√	
START CMDSERV		√						
START LISTENER		√ <sup>2</sup>	√	√			√	
START QMGR		√						
START TRACE		√						
STOP CHANNEL	√	√ <sup>2</sup>	√	√	√	√	√	√
STOP CHINIT		√ <sup>2</sup>						
STOP CMDSERV		√						
STOP LISTENER		√ <sup>2</sup>						
STOP QMGR		√						
STOP TRACE		√						

**Notes:**

1. For a description of the syntax for commands on this platform, see the online *MQSeries for Windows Command Reference*.
2. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, the equivalent function is available using the DQM panels (the CKMC transaction). See the *MQSeries Intercommunication* manual for information about this.
3. This command is valid only on MVS/ESA. For other platforms, use the MAXUMSGS keyword of the ALTER QMGR command instead.
4. MQSeries for Windows Version 2.1 only.
5. See "DISPLAY QUEUE" on page 141.
6. AIX, HP-UX, and Sun Solaris only.

---

**ALTER CHANNEL**

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

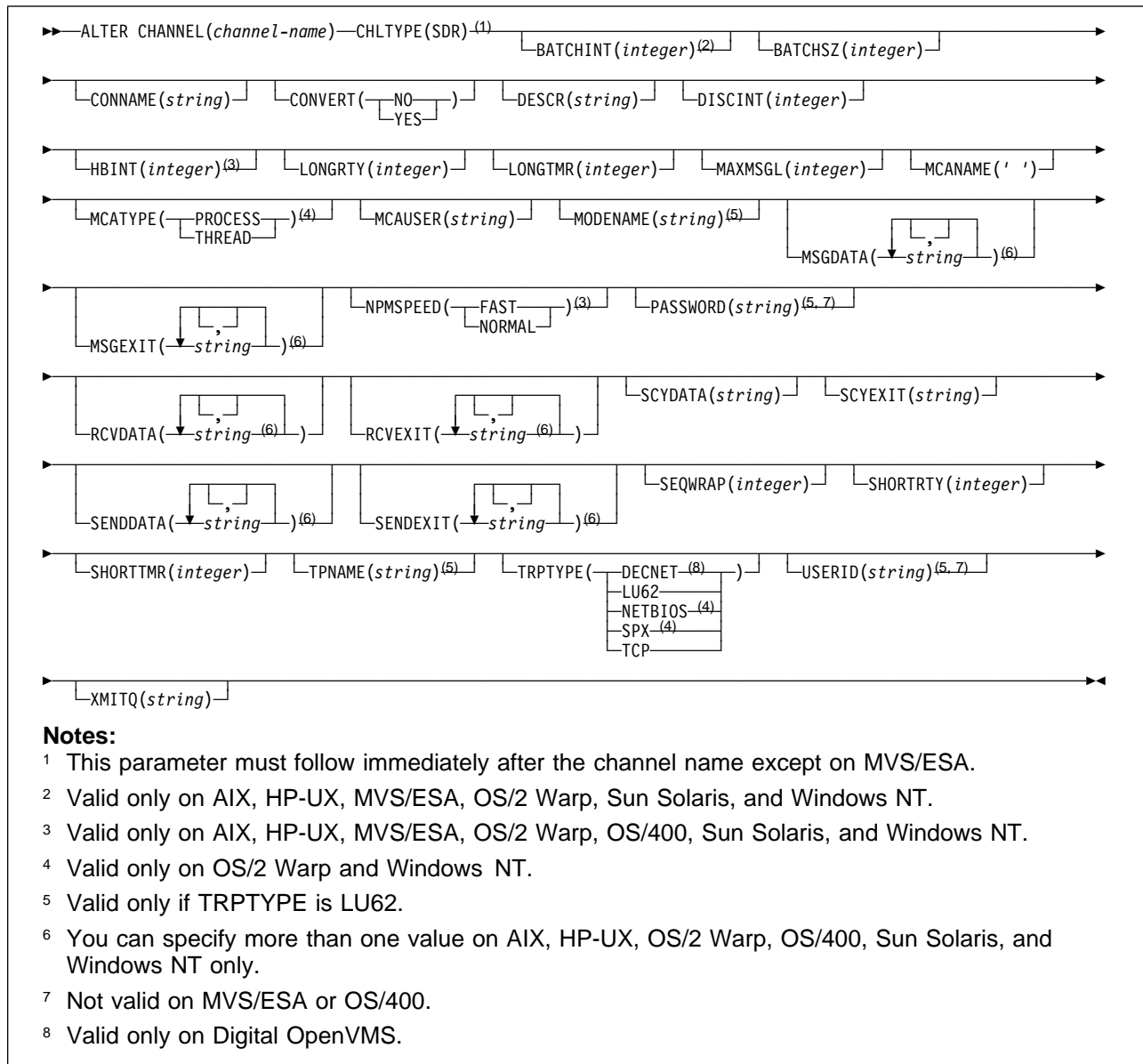
Use ALTER CHANNEL to alter the attributes of a channel.

**Notes:**

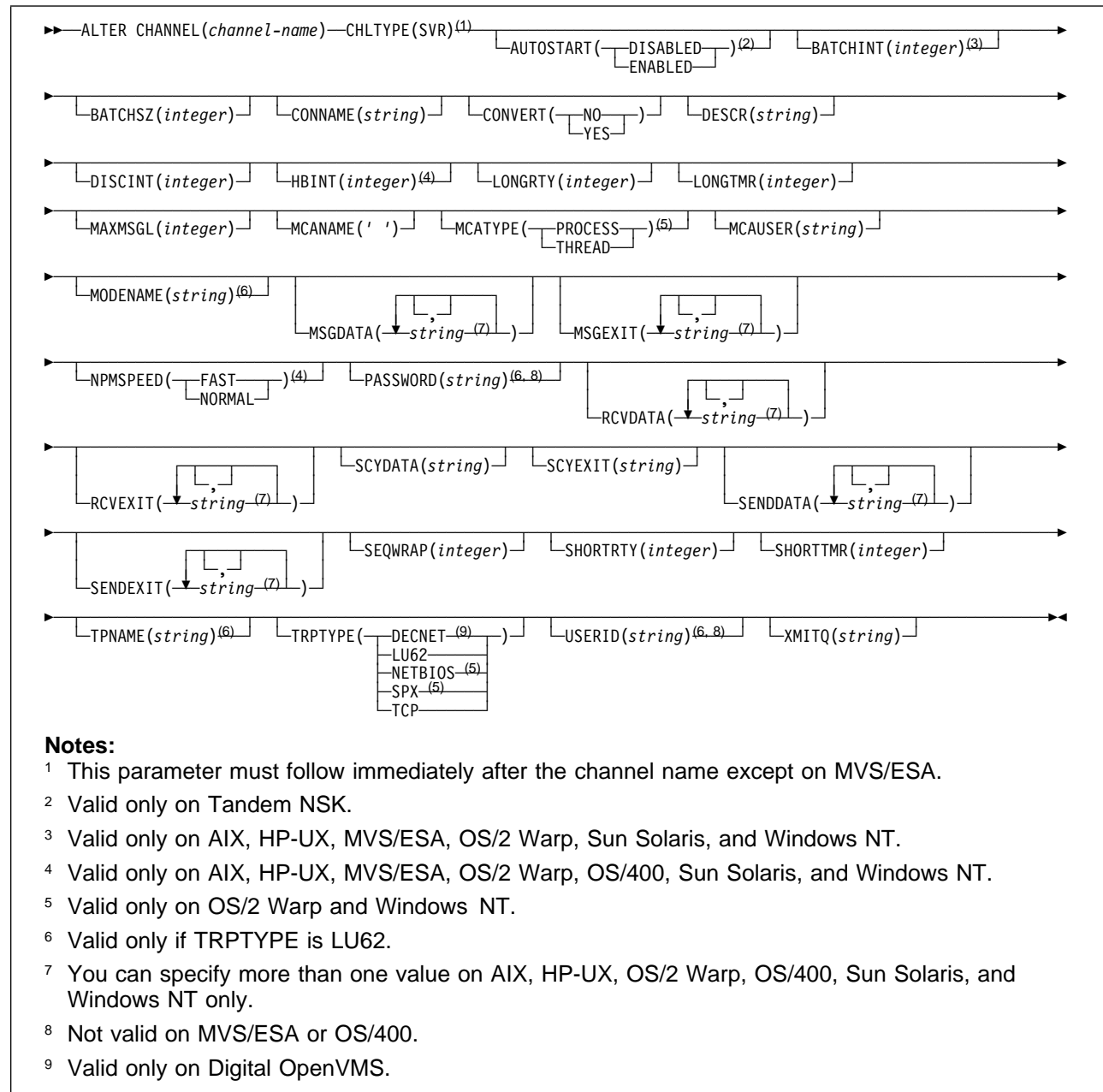
1. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.
2. There is a separate syntax diagram for each of the six types of channel.

**Synonym:** ALT CHL

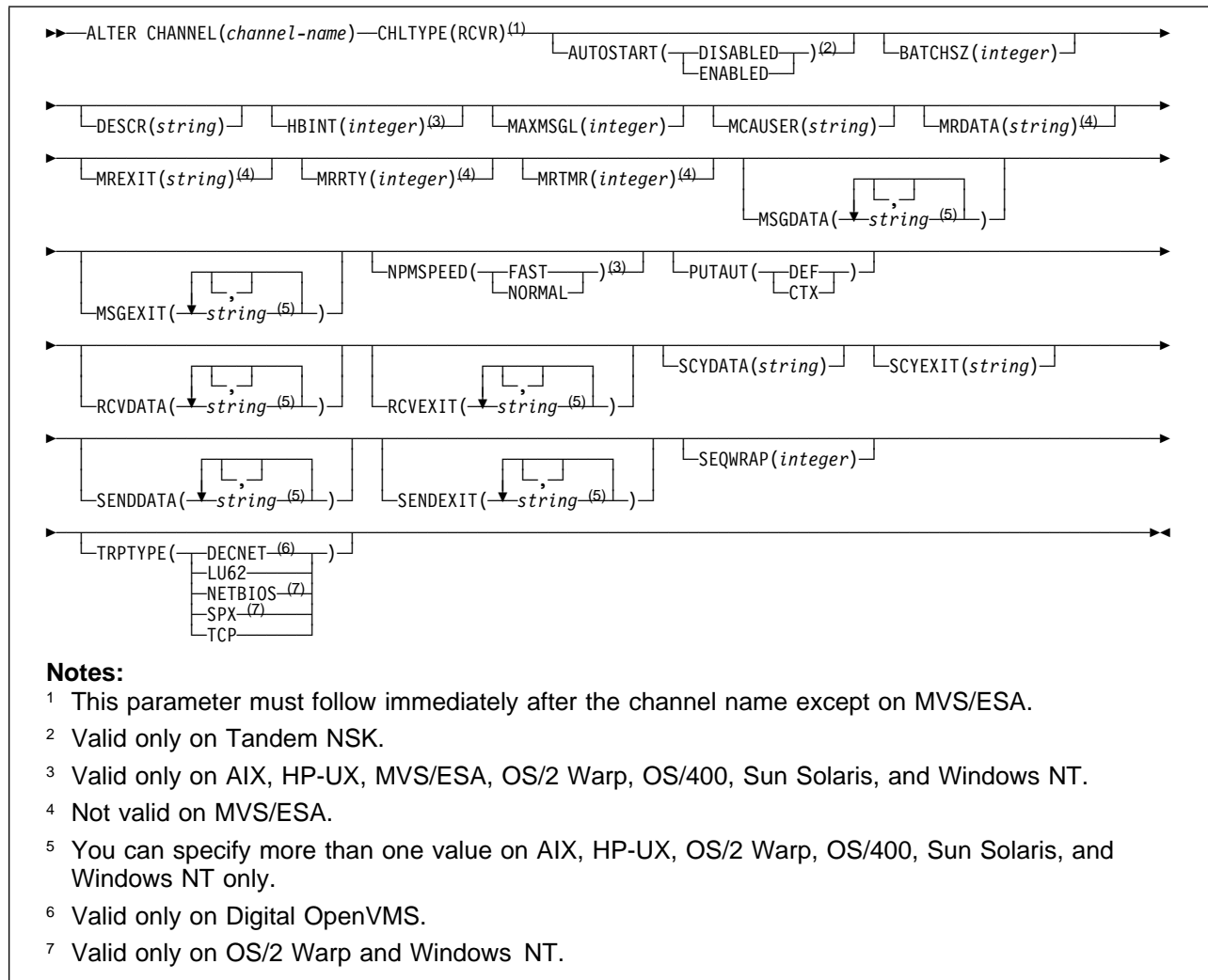
Sender channel



## Server channel

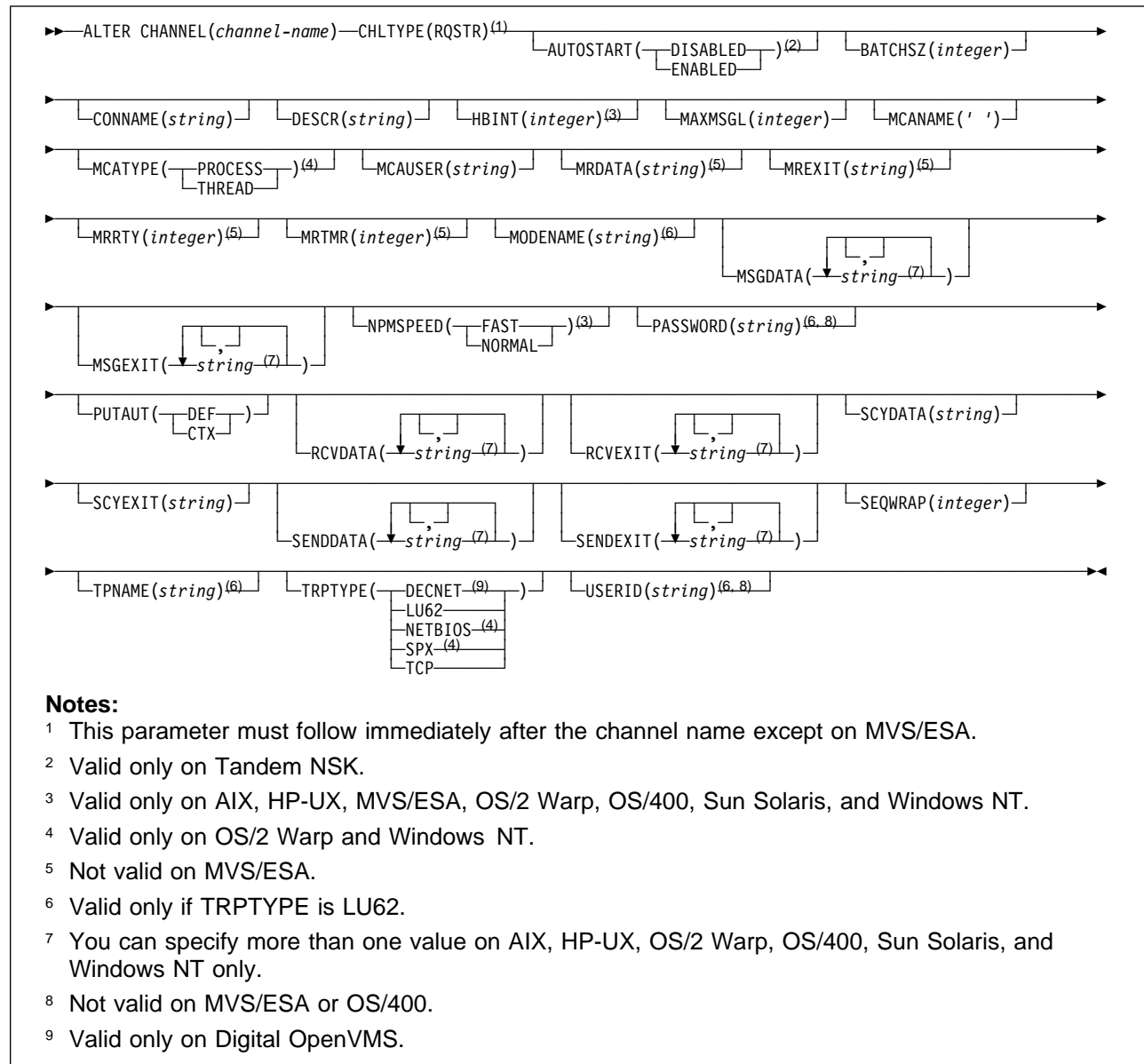


## Receiver channel

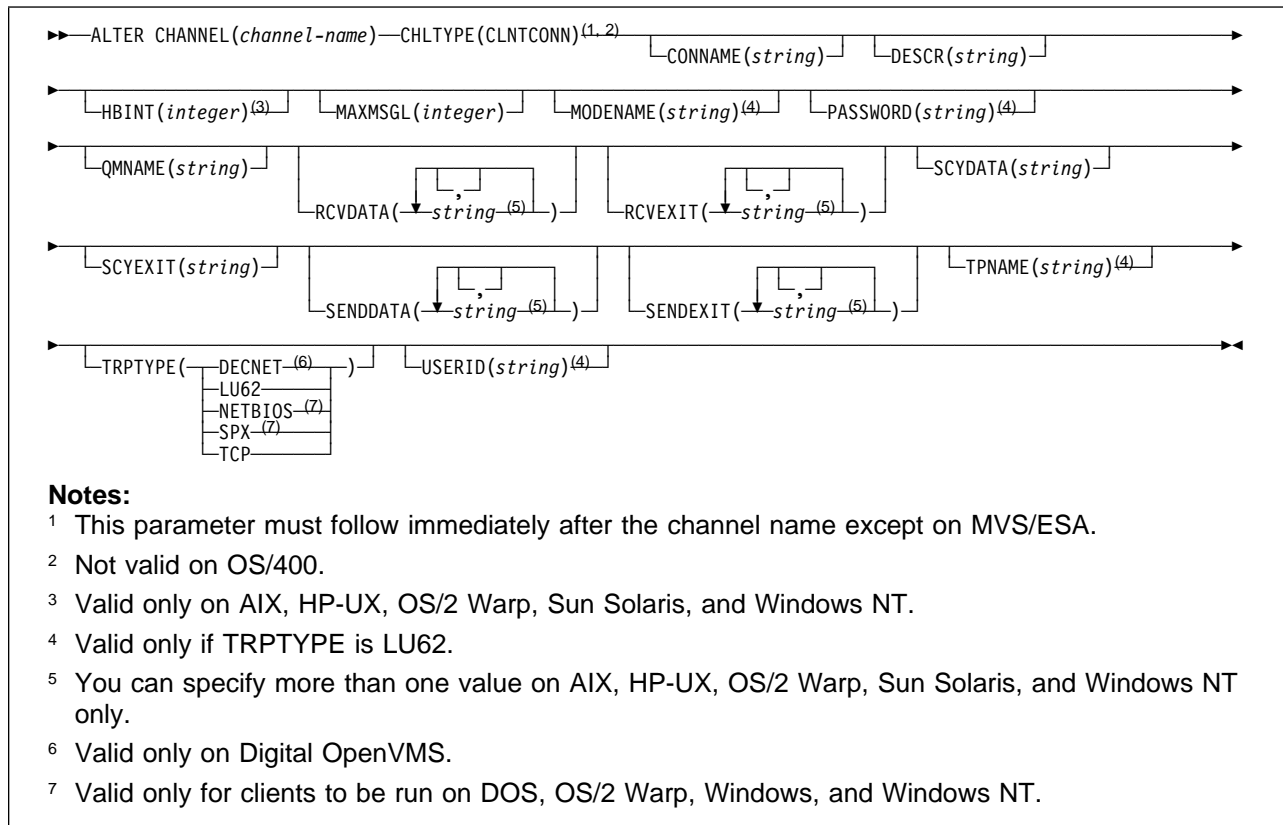




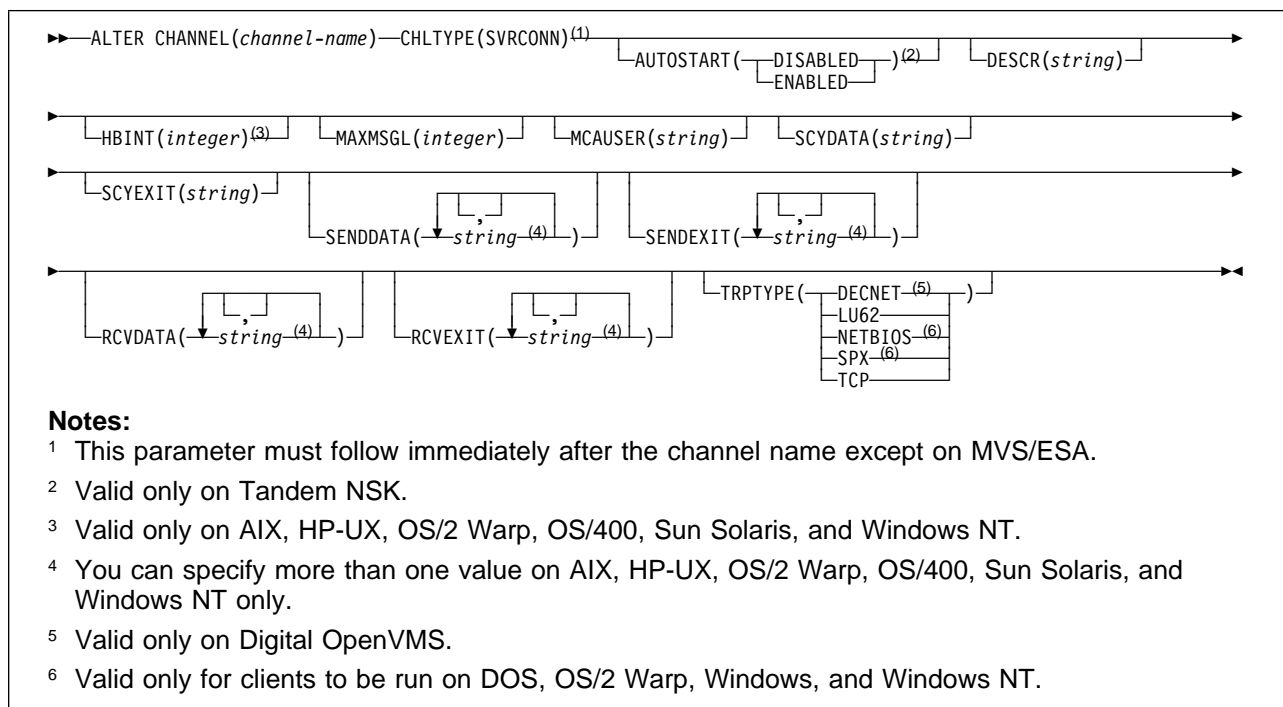
## Requester channel



### Client-connection channel



### Server-connection channel



## Keyword and parameter descriptions

Attributes specified override the current values. Attributes that you do not specify are unchanged. Some attributes depend on the type of the channel – see the CHLTYPE parameter.

Parameters are optional unless the description states that they are required.

*(channel-name)*

The name of the channel definition. This is required.

The name must be defined to the local queue manager. The maximum length of the string is 20 characters, and the string must contain only valid characters; see “Rules for naming MQSeries objects” on page 5.

### AUTOSTART

Specifies whether an LU 6.2 responder process for the channel will be started at queue manager startup.

**ENABLED** The responder is started

**DISABLED** The responder is not started (this is the default)

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR, RQSTR, SVR, and SVRCONN. It is supported only on Tandem NSK.

### BATCHINT(*integer*)

The minimum amount of time, in milliseconds, that a channel will keep a batch open.

The batch is terminated by whichever of the following occurs first:

- BATCHSZ messages have been sent, or
- The transmission queue is empty and BATCHINT is exceeded

The default value is zero, 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 greater than or equal to zero, and less than or equal to 999 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR. It is valid only on AIX, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT.

### BATCHSZ(*integer*)

The maximum number of messages that can be sent through a channel before taking a checkpoint.

The maximum batch size actually used is the lowest of the following:

- The BATCHSZ of the sending channel
- The BATCHSZ of the receiving channel
- The maximum number of uncommitted messages allowed at the sending queue manager
- The maximum number of uncommitted messages allowed at the receiving queue manager

The maximum number of uncommitted messages is specified by the MAXUMSGS parameter of the ALTER QMGR command, or the DEFINE MAXSMMSG command on MVS/ESA.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, RCVR, or RQSTR.

The value must be greater than zero, and less than or equal to 9999.

## ALTER CHANNEL

### CHLTYPE

Channel type. This is required, and must be of the same type as the existing channel. It must follow immediately after the (*channel-name*) parameter on all platforms except MVS/ESA.

<b>SDR</b>	Sender channel
<b>SVR</b>	Server channel
<b>RCVR</b>	Receiver channel
<b>RQSTR</b>	Requester channel
<b>CLNTCONN</b>	Client-connection channel (not valid on OS/400)
<b>SVRCONN</b>	Server-connection channel

### CONNNAME(*string*)

Connection name.

The connection name of the partner. (The maximum length is 48 characters on MVS/ESA, and 264 characters on other platforms.) The type of name depends on the transport type (TRPTYPE) to be used:

#### DECnet Phase IV

The DECnet node name and the DECnet object name, in the form:

```
CONNNAME('node_name(object_name)')
```

### LU 6.2

- On Digital OpenVMS this is the gateway node, access name, and the tpname that is used by SNA to invoke the remote program. The format of this information is as follows:  

```
CONNNAME('gateway_node.access_name(tpname)')
```
  - On MVS/ESA this is the symbolic destination name for the partner LU, as defined in the side information data set.
  - On OS/2 Warp it is the fully-qualified name of the partner Logical Unit.
  - On OS/400, Windows NT, and UNIX systems (except SunOS), this is the name of the CPI-C communications side object.
  - On SunOS, this is the name of the gateway on which the queue manager is running.
  - On Tandem NSK, the value of this depends on whether SNAX or ICE is used as the communications protocol:
    - If SNAX is used:
      - For sender, requester, and fully qualified server channels, this is the process name of the SNAX/APC process, the name of the local LU, and the name of the partner LU on the remote machine, for example:  

```
CONNNAME('çPPPP.LOCALLU.REMOTELU')
```
      - For receiver and non fully qualified server channels, this is the process name of the SNAX/APC process and the name of the local LU, for example:  

```
CONNNAME('çPPPP.LOCALLU')
```
- The name of the local LU can be an asterisk (\*), indicating any name.

– If ICE is used:

- For sender, requester, and fully qualified server channels, this is the process name of the ICE process, the ICE open name, the name of the local LU, and the name of the partner LU on the remote machine, for example:

```
CONNAME('¢PPPP.#OPEN.LOCALLU.REMOTELU')
```

For receiver and non fully qualified server channels, this is the process name of the SNAX/APC process, the ICE open name, and the name of the local LU, for example:

```
CONNAME('¢PPPP.#OPEN.LOCALLU')
```

The name of the local LU can be an asterisk (\*), indicating any name.

### 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)')
```

If the socket number is omitted, the MQSeries default value (X'5e86') is assumed.

### TCP/IP

Either the host name, or the network address of the remote machine. This can be followed by an optional port number, enclosed in parentheses.

This parameter is required for channels with a channel type (CHLTYPE) of SDR, RQSTR, or CLNTCONN. It is optional for SVR channels, and is not valid for RCVR or SVRCONN channels.

**Note:** If you are using any of the special characters in your connection name (for example, parentheses) you must enclose the string in single quotes.

### CONVERT

Specifies whether the sending message channel agent should attempt conversion of the application message data, if the receiving message channel agent is unable to perform this conversion.

**NO** No conversion by sender

**YES** Conversion by sender

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### DESCR(*string*)

Plain-text comment. It provides descriptive information about the channel when an operator issues the DISPLAY CHANNEL command.

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.

### 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. The value must be greater than or equal to zero, and less than or equal to 999 999. A value of zero causes the message channel agent to wait indefinitely.

## ALTER CHANNEL

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### HBINT(*integer*)

This parameter has a different interpretation depending upon the channel type, as follows:

- For a channel type of SDR, SVR, RCVR, or RQSTR, this is 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 type of heartbeat is valid only on AIX, HP-UX, MVS/ESA, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

**Note:** You should set this value to be significantly less than the value of DISCINT. MQSeries checks only that it is within the permitted range however.

- For a channel type of SVRCONN or CLNTCONN, this is the time, in seconds, between heartbeat flows passed from the server MCA when that MCA has issued an MQGET with WAIT on behalf of a client application. This allows the server to handle situations where the client connection fails during an MQGET with WAIT. This type of heartbeat is valid only for AIX, HP-UX, OS/2 Warp, OS/400 (SVRCONN only), Sun Solaris, and Windows NT.

The value must be in the range zero through 999 999. A value of zero means that no heartbeat exchange takes place. The value that is used is the larger of the values specified at the sending side and the receiving side.

### LONGRTY(*integer*)

When a sender or server channel is attempting to connect to the remote queue manager, and the count specified by SHORTRTY has been exhausted, this 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 subsequently be restarted with a command (it is not started automatically by the channel initiator), and it then makes only one attempt to connect, because 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.

The value must be greater than or equal to zero, and less than or equal to 999 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### LONGTMR(*integer*)

For long retry attempts, this is the maximum number of seconds to wait before re-attempting 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 greater than or equal to zero, and less than or equal to 999 999 999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999 999; values exceeding this will be treated as 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### MAXMSGL(*integer*)

Specifies the maximum message length that can be transmitted on the channel. This is compared with the value for the partner and the actual maximum used is the lower of the two values.

The value zero means the maximum message length for the queue manager.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the MAXMSGL parameter of the ALTER QMGR command for more information. On other platforms, specify a value greater than or equal to zero, and less than or equal to 4 194 304.

**MCANAME**(*string*)

Message channel agent name.

This is reserved, and if specified must only be set to blanks (maximum length 20 characters).

**MCATYPE**

Specifies whether the message-channel-agent program should run 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

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, or RQSTR. It is supported only on OS/2 Warp and Windows NT.

**MCAUSER**(*string*)

Message channel agent user identifier (maximum length 12 characters).

If *string* is nonblank, it is the user identifier which is to be used by the message channel agent for authorization to access MQSeries 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.

This user identifier can be overridden by one supplied by a channel security exit.

This parameter is not valid for channels with a channel type (CHLTYPE) of CLNTCONN.

**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.

On SunOS, the MODENAME should be set to the *unique\_session\_name* parameter as specified in the Sunlink configuration file. The name can also be nonblank for client connection channels to be used with OS/2 Warp.

On Tandem NSK, this should be set to the SNA mode name.

On other platforms, the mode name can only be set to blanks; the actual name is taken instead from the CPI-C Communications Side Object, or APPC side information data set.

This parameter is not valid for channels with a channel type (CHLTYPE) of RCVR or SVRCONN.

**MRDATA**(*string*)

Channel message-retry exit user data (maximum length 32 characters).

This 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 or RQSTR. It is not supported on MVS/ESA.

**MREXIT**(*string*)

Channel message-retry exit name.

The format and maximum length of the name is the same as for MSGEXIT.

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR or RQSTR. It is not supported on MVS/ESA.

## ALTER CHANNEL

### MRRTY(*integer*)

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. If the exit name is not blank, the value of MRRTY is passed to the exit for the exit's use, but the number of retries performed (if any) is controlled by the exit, and not by this attribute.

The value must be greater than or equal to zero, and less than or equal to 999 999 999. A value of zero means that no retries will be performed.

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR or RQSTR. It is not supported on MVS/ESA.

### MRTMR(*integer*)

The minimum interval of time that must pass before the channel can retry the MQPUT operation. This time interval is in milliseconds.

This attribute 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 for the exit's use, but the retry interval is controlled by the exit, and not by this attribute.

The value must be greater than or equal to zero, and less than or equal to 999 999 999. A value of zero means that the retry will be performed as soon as possible (provided that the value of MRRTY is greater than zero).

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR or RQSTR. It is not supported on MVS/ESA.

### MSGDATA

User data for the channel message exit (maximum length 32 characters).

This data is passed to the channel message exit when it is called.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, you can specify data for more than one exit program by specifying multiple strings separated by commas. On OS/400, 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.

On other platforms you can specify only one string of message exit data for each channel.

### MSGEXIT

Channel message exit name.

On Tandem NSK, there is only one channel user exit program. 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 a text string for these attributes. The maximum length of the string is 128 characters. This string is passed to the exit program, but it is not used to determine the program name.

See the *MQSeries for Tandem NonStop Kernel System Management Guide* for more information about using channel exit programs on Tandem NSK.

On other platforms, 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, OS/2 Warp, Sun Solaris, and Windows NT, 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 OS/400, you can specify the names of up to 10 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 not relevant, because message exits are not invoked for such channels.

The format and maximum length of the name depends on the environment:

- On Digital OpenVMS and UNIX systems, it is of the form

libraryname(functionname)

The maximum length of the string is 128 characters.

- On OS/2 Warp, Windows, and Windows NT, 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 OS/400, it is of the form

progname libname

where *progname* 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 MVS/ESA, it is a load module name, maximum length 8 characters (128 characters are allowed for exit names for client-connection channels).

### NPMSPEED

The class of service for nonpersistent messages on this channel:

**NORMAL** Normal delivery for nonpersistent messages.

**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.

If the sending side and the receiving side do not agree about this attribute, or one does not support it, NORMAL is used.

This parameter is valid only for channels with a CHLTYPE of SDR, SVR, RCVR, or RQSTR. It is valid only on AIX, HP-UX, MVS/ESA, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

### PASSWORD(*string*)

Password (maximum length 12 characters).

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, or CLNTCONN. It is not supported on OS/400 and is only supported on MVS/ESA for client-connection channels.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

## ALTER CHANNEL

### PUTAUT

Specifies whether the user identifier in the context information associated with a message should be used to establish authority to put the message to the destination queue.

**DEF** Default user ID is used  
**CTX** Context user ID is used

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR or RQSTR.

### QMNAME(*string*)

Queue manager name.

For channels with a channel type (CHLTYPE) of CLNTCONN, this is the name of the queue manager to which an application running in the MQI client environment can request connection.

For channels of other types this parameter is not valid.

### RCVDATA

Channel receive exit user data (maximum length 32 characters).

This is passed to the channel receive exit when it is called.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, you can specify data for more than one exit program by specifying multiple strings separated by commas. On OS/400, 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.

On other platforms you can specify only one string of receive exit data for each channel.

### RCVEXIT

Channel receive exit name.

On platforms other than Tandem NSK, 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, OS/2 Warp, Sun Solaris, and Windows NT, 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 OS/400, you can specify the names of up to 10 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

Channel security exit user data (maximum length 32 characters).

This is passed to the channel security exit when it is called.

### SCYEXIT

Channel security exit name.

On platforms other than Tandem NSK, 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 given the opportunity 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.

### SENDDATA

Channel send exit user data (maximum length 32 characters).

This is passed to the channel send exit when it is called.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, you can specify data for more than one exit program by specifying multiple strings separated by commas. On OS/400, 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.

On other platforms you can specify only one string of send exit data for each channel.

### SENDEXIT

Channel send exit name.

On platforms other than Tandem NSK, 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, OS/2 Warp, Sun Solaris, and Windows NT, 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 OS/400, you can specify the names of up to 10 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 non-negotiable and must match in both the local and remote channel definitions.

The value must be greater than or equal to 100, and less than or equal to 999 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, RCVR, or RQSTR.

### SHORTRTY(*integer*)

The maximum number of attempts that are made by a sender or server 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 retry is unlikely to be successful, retries are not attempted.

The value must be greater than or equal to zero, and less than or equal to 999 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

## ALTER CHANNEL

### SHORTTMR(*integer*)

For short retry attempts, this is the maximum number of seconds to wait before re-attempting 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 greater than or equal to zero, and less than or equal to 999 999 999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999 999; values exceeding this will be treated as 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### 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.

On SunOS, the transaction program name should be set to the name to which the listener program (runmqsr) is listening at the remote end.

On Tandem NSK, this should be set to the local TP name. This can be followed by the name of the TP on the remote machine, for example:

```
TPNAME('localtp[.remotetp]')
```

Both names can be up to 16 characters in length.

The name can also be nonblank for client connection channels to be used with OS/2 Warp.

On other platforms, the transaction program name can only be set to blanks; the actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set. On Windows NT SNA Server, and in the side object on MVS/ESA, the TPNAME is folded to upper case.

This parameter is not valid for channels with a channel type (CHLTYPE) of RCVR.

### TRPTYPE

Transport type to be used. No check is made that the correct transport type has been specified if the channel is initiated from the other end.

**DECNET** DECnet Phase IV (supported only on Digital OpenVMS)

**LU62** SNA LU 6.2

**NETBIOS** NetBIOS (supported only on OS/2 Warp, Windows, Windows NT, and DOS; it also applies to MVS/ESA for client-connection channels)

**SPX** Sequenced packet exchange (supported only on OS/2 Warp, Windows, Windows NT, and DOS)

**TCP** Transmission Control Protocol/Internet Protocol (TCP/IP)

### USERID(*string*)

Task user identifier (maximum length 12 characters).

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, or CLNTCONN. It is not supported on OS/400 and is only supported on MVS/ESA for client-connection channels.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

**XMITQ**(*string*)

Transmission queue name.

The name of the queue from which messages are retrieved. See “Rules for naming MQSeries objects” on page 5.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

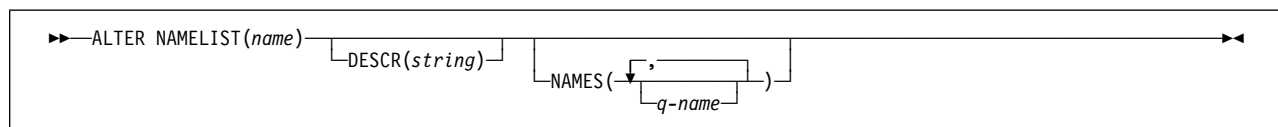
## ALTER NAMELIST

### ALTER NAMELIST

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use ALTER NAMELIST to alter a list of queue names.

**Synonym:** ALT NL



### Keyword and parameter descriptions

A namelist is used to hold a list of queue names.

The attributes you specify override the current values. Attributes that you do not specify are unchanged.

*(name)* Name of the list. This is required. The list must already be defined.

#### DESCR(*string*)

Plain-text comment. This is optional. It provides descriptive information about the namelist when an operator issues the DISPLAY NAMELIST command.

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.

#### NAMES(*q-name, ...*)

List of names of queues. This is optional.

The queues can be of any type, and the list can contain queues of more than one type. In fact, because the objects defined in the namelist do not have to exist at the time the namelist is altered, there is no check that they are even the names of queues, although the characters used for each name are restricted to those that are valid for queue names (see “Rules for naming MQSeries objects” on page 5).

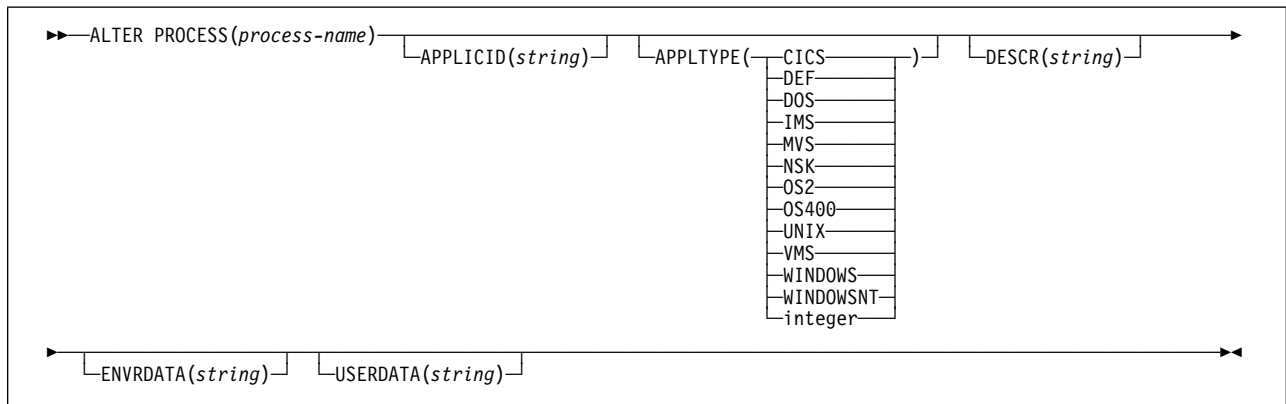
The maximum number of names in the list is 256. An empty list is valid: specify NAMES().

## ALTER PROCESS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
✓	✓	✓	✓	✓	✓	✓

Use ALTER PROCESS to alter the attributes of an existing MQSeries process definition.

**Synonym:** ALT PRO



## Keyword and parameter descriptions

The attributes you specify override the current values. Attributes that you do not specify are unchanged.

*(process-name)*

The name of the MQSeries process definition to be altered (see “Rules for naming MQSeries objects” on page 5). This is required. The name must be defined to the local queue manager. The maximum length is 48 bytes.

**APPLICID***(string)*

The name of the application to be started. This might typically be a fully-qualified file name of an executable object. The maximum length is 256 characters.

For a CICS application this is a CICS transaction ID, and for an IMS application it is an IMS transaction ID.

On MVS/ESA, for distributed queuing using CICS it must be “CKSG”, and for distributed queuing without CICS, it must be “CSQX START”.

**APPLTYPE***(string)*

The type of application to be started. Valid application types are:

<b>CICS</b>	Represents a CICS transaction.
<b>DOS</b>	Represents a DOS application.
<b>IMS</b>	Represents an IMS transaction.
<b>MVS</b>	Represents an MVS application (batch or TSO).
<b>NSK</b>	Represents a Tandem NSK application.
<b>OS2</b>	Represents an OS/2 Warp application.
<b>OS400</b>	Represents an OS/400 application.
<b>UNIX</b>	Represents a UNIX application.
<b>VMS</b>	Represents a Digital OpenVMS application.

## ALTER PROCESS

<b>WINDOWS</b>	Represents a Windows application.
<b>WINDOWSNT</b>	Represents a Windows NT application.
<b>integer</b>	User-defined application type in the range 65 536 through 999 999 999.
<b>DEF</b>	This 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, this is interpreted as the default application type of the server.

Only application types (other than user-defined types) that are supported on the platform at which the command is executed should be used:

- On Digital OpenVMS, VMS is supported
- On MVS/ESA, CICS (default), IMS, MVS, and DEF are supported
- On OS/400, OS400 (default), CICS, and DEF are supported
- On OS/2 Warp, OS2 (default), DOS, WINDOWS, UNIX, CICS, and DEF are supported
- On Tandem NSK, NSK is supported
- On UNIX systems, UNIX (default), OS2, DOS, WINDOWS, CICS, and DEF are supported
- On Windows NT, WINDOWSNT (default), DOS, WINDOWS, OS2, UNIX, CICS, and DEF are supported

### **DESCR**(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY PROCESS command.

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.

### **ENVRDATA**(*string*)

A character string that contains environment information pertaining to the application to be started. The maximum length is 128 characters.

### **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.

For MQSeries message channel agents, the format of this field is a channel name of up to 20 characters. See the *MQSeries Intercommunication* manual for information about what these need as APPLICID.

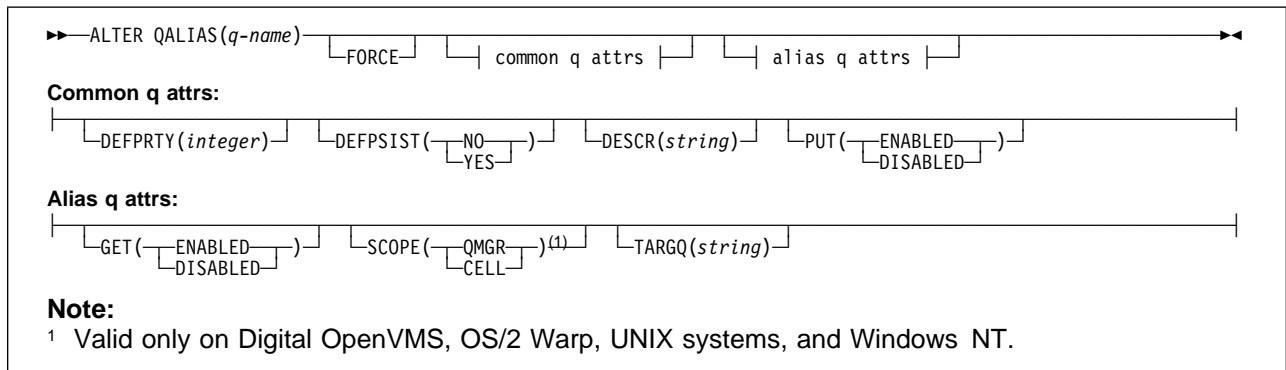


## ALTER QALIAS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use ALTER QALIAS to alter the attributes of an alias queue.

**Synonym:** ALT QA



## Keyword and parameter descriptions

The attributes you specify override the current values. Attributes that you do not specify are unchanged.

*(q-name)*

The name of the alias queue to be altered (see “Rules for naming MQSeries objects” on page 5). The name must be defined to the local queue manager.

**FORCE** Specify this to force completion of the command if both of the following are true:

- The TARGQ keyword is specified
- An application has this alias queue open

If FORCE is not specified in these circumstances, the command is unsuccessful, and no changes are made.

### Common queue attributes

**DEFPRTY***(integer)*

The default priority of messages put on the queue. The value must be greater than or equal to zero, (the lowest priority) and less than or equal to the MAXPRTY queue manager attribute. (MAXPRTY is 9.)

**DEFPSIST**

The default message persistence for this queue:

**NO** Messages on this queue are lost across a restart of the queue manager. This is the default supplied with MQSeries, but your installation might have changed it.

**YES** Messages on this queue survive a restart of the queue manager.

## ALTER QALIAS

### DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY QUEUE command.

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.

**PUT** Whether messages can be put on the queue.

**ENABLED** Suitably authorized applications can add messages to the queue. This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Messages cannot be added to the queue.

This attribute can also be changed using the **MQSET** API call.

### Alias queue attributes

**GET** Whether applications are permitted to get messages from this queue.

**ENABLED** Suitably authorized applications can retrieve messages from the queue. This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Applications cannot retrieve messages from the queue.

This attribute can also be changed using the **MQSET** API call.

### SCOPE

Specifies the scope of the queue definition.

**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. To open the queue for output from some other queue manager, either the name of the owning queue manager must be specified, or the other queue manager must have a local definition of the queue.

**CELL** The queue definition has cell scope. This means that the queue is known to all of the queue managers in the cell, and 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.

This value is valid only if a name service supporting a cell directory (for example, the supplied DCE name service) has been configured.

This attribute is only supported on Digital OpenVMS, OS/2 Warp, Windows NT, and UNIX systems.

### TARGQ(*string*)

The local name of the base queue being aliased. (See "Rules for naming MQSeries objects" on page 5.) The maximum length is 48 characters.

This must be one of the following (although this is not checked until the alias queue is opened by an application):

- A local queue (not a dynamic queue)
- A local definition of a remote queue

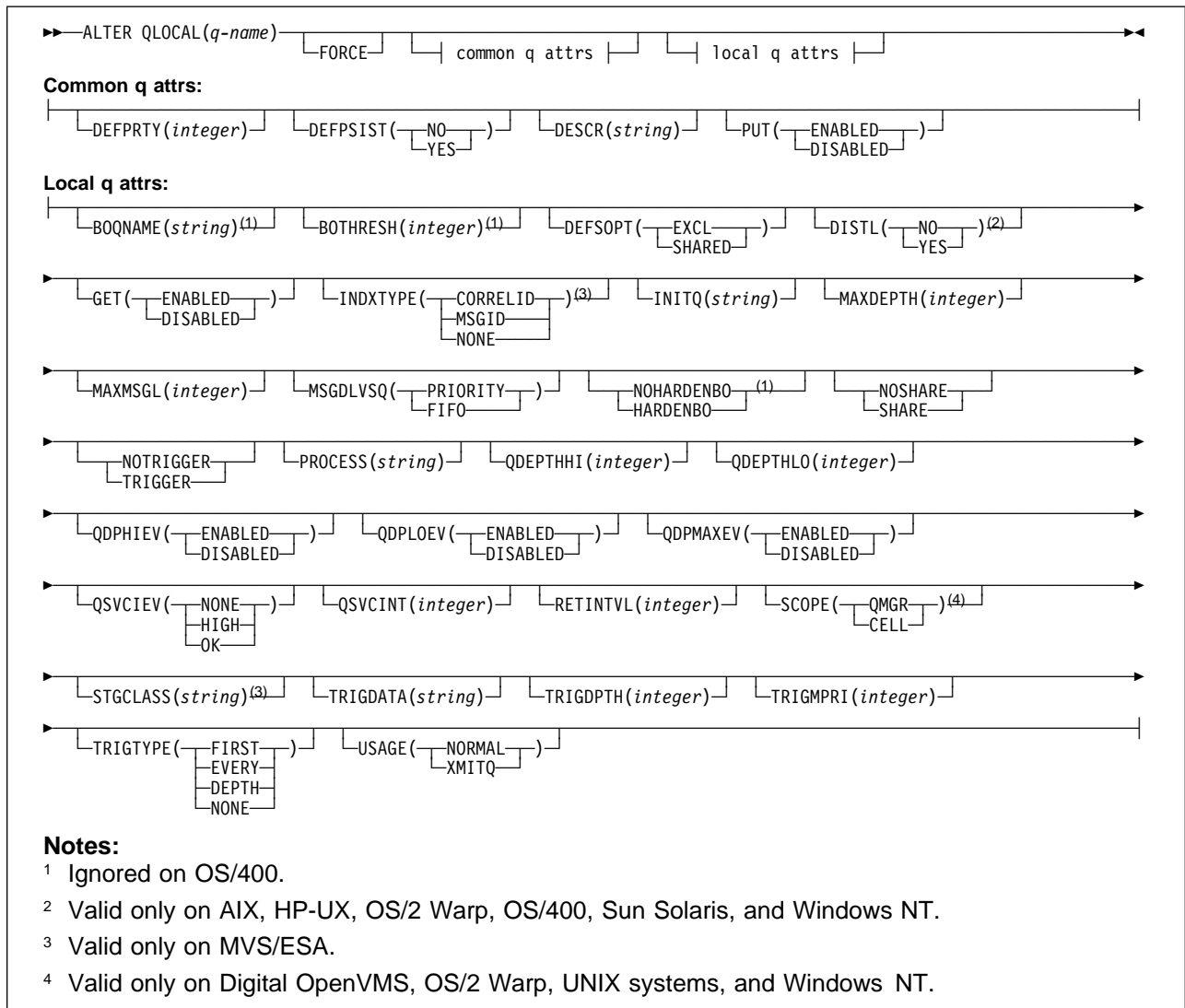
This queue need not be defined until an application process attempts to open the alias queue.

## ALTER QLOCAL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use ALTER QLOCAL to alter the attributes of a local queue.

**Synonym:** ALT QL



## Keyword and parameter descriptions

Attributes specified override the current values. Attributes that you do not specify are unchanged.

(q-name)

The local name of the local queue to be altered (see “Rules for naming MQSeries objects” on page 5). The name must be defined to the local queue manager.

## ALTER QLOCAL

**FORCE** Specify this to force completion of the command if both of the following are true:

- The NOSHARE keyword is specified
- One or more applications have the queue open for input

If FORCE is not specified in these circumstances, the command is unsuccessful.

FORCE is also needed if both of the following are true:

- The USAGE attribute is changed
- Either one or more messages are on the queue, or one or more applications have the queue open

Again, the command is unsuccessful if FORCE is not specified in these circumstances.

Do not change the USAGE attribute while there are messages on the queue; the format of messages changes when they are put on a transmission queue.

### Common queue attributes

#### DEFPRTY(*integer*)

The default priority of messages put on the queue. The value must be greater than or equal to zero, (the lowest priority) and less than or equal to the MAXPRTY queue manager attribute. MAXPRTY can be displayed using the DISPLAY QMGR command. (MAXPRTY is 9.)

#### DEFPSIST

The default message persistence for this queue:

**NO** Messages on this queue are lost across a restart of the queue manager. This is the default supplied with MQSeries, but your installation might have changed it.

**YES** Messages on this queue survive a restart of the queue manager.

#### DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY QUEUE command.

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.

**PUT** Whether messages can be put on the queue.

**ENABLED** Messages can be added to the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Messages cannot be added to the queue.

This attribute can also be changed using the **MQSET** API call.

### Local queue attributes

#### BOQNAME(*string*)

The excessive backout requeue name. Apart from maintaining a value for this attribute, the queue manager takes no action based on its value. On OS/400, the keyword is ignored.

#### BOTHRESH(*integer*)

The backout threshold. Apart from maintaining a value for this attribute, the queue manager takes no action based on its value. On OS/400, the keyword is ignored.

Specify a value greater than or equal to zero, and less than or equal to 999 999 999.

**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

**DISTL** 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 should not normally change this attribute, because it is set by the MCA. However you can set this attribute when defining a transmission queue if the distribution list capability of the destination queue manager is known.

This keyword is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

**GET** Whether applications are to be permitted to get messages from this queue:

**ENABLED** Messages can be retrieved from the queue (by suitably authorized applications).

This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Applications cannot retrieve messages from the queue.

This attribute can also be changed using the **MQSET** API call.

**INDXTYPE**

The type of index maintained by the queue manager to expedite **MQGET** operations on the queue:

**NONE** No index is maintained. Use this when messages are usually 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 this when messages are usually 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 this when messages are usually retrieved using the correlation identifier as a selection criterion on the **MQGET** call with the message identifier set to NULL.

The **INDXTYPE** attribute can be changed at any time, and the change takes effect immediately if all the following conditions are satisfied:

- No applications have the queue open
- The queue is empty
- There are no uncommitted MQPUT or MQGET operations outstanding against the queue

If these conditions are not satisfied, the attribute is changed immediately, but the index is not rebuilt until the next time the queue manager is restarted. The reply sent by the queue manager indicates if this is the case.

This attribute is supported only on MVS/ESA. On other platforms, retrieval optimization might be provided, but it is not controlled by a queue attribute.

**INITQ(string)**

The local name of a local queue (known as the *initiation queue*) on this queue manager, to which trigger messages relating to this queue are written. See "Rules for naming MQSeries objects" on page 5.

## ALTER QLOCAL

### MAXDEPTH(*integer*)

The maximum number of messages allowed on the queue. Specify a value greater than or equal to zero, and less than or equal to:

- 999 999 999 if the queue is on MVS/ESA
- 640 000 if the queue is on any other MQSeries platform

Other factors might still cause the queue to be treated as full, for example, if there is no further DASD space available.

If this value is reduced, any messages that are already on the queue that cause the new maximum to be exceeded remain intact.

### MAXMSGL(*integer*)

The maximum length of messages on this queue. On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the MAXMSGL parameter of the ALTER QMGR command for more information. On other platforms, specify a value greater than or equal to zero, and less than or equal to 4 194 304.

For a transmission queue, this value includes the space required for headers. It is recommended that the value should be at least 4000 bytes larger than the maximum expected length of user data in any message that could be put on a transmission queue.

If this value is reduced, any messages that are already on the queue, whose length exceeds the new maximum, are not affected.

Applications can use this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore, the value should only be reduced if it is known that this will not cause an application to operate incorrectly.

### MSGDLVSEQ

Message delivery sequence:

**PRIORITY** Messages are delivered (in response to **MQGET** API calls) in first-in-first-out (FIFO) order within priority. This is the default supplied with MQSeries, but your installation might have changed it.

**FIFO** Messages are delivered (in response to **MQGET** API calls) in FIFO order. Priority is ignored for messages on this queue.

If the message delivery sequence is changed from PRIORITY to FIFO while there are messages on the queue, the order of the messages already enqueued is not changed. Messages added to the queue subsequently 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 enqueued while the queue was set to FIFO take the default priority.

### NOHARDENBO and HARDENBO

Whether hardening should be used to ensure that the count of the number of times that a message has been backed out is accurate.

**Note:** The count is always hardened on OS/400; the HARDENBO and NOHARDENBO keywords are ignored if specified.

**NOHARDENBO** The count is not hardened. This is the default supplied with MQSeries, but your installation might have changed it.

**HARDENBO** The count is hardened.

**NOSHARE** and **SHARE**

Whether multiple applications can get messages from this queue:

**NOSHARE** A single application instance only can get messages from the queue

**SHARE** More than one application instance can get messages from the queue

**NOTRIGGER** and **TRIGGER**

Whether trigger messages are written to the initiation queue (named by the INITQ attribute) to trigger the application (named by the PROCESS attribute):

**NOTRIGGER** Triggering is not active, and trigger messages are not written to the initiation queue. This is the default supplied with MQSeries, but your installation might have changed it.

**TRIGGER** Triggering is active, and trigger messages are written to the initiation queue.

This attribute can also be changed using the **MQSET** API call.

**PROCESS**(*string*)

The local name of the MQSeries process. This 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 MQSeries objects” on page 5.

The process does not have to be defined 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 gives the name of the channel to be started. This parameter is optional for transmission queues on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT; if you do not specify it, the channel name is taken from the value specified for the TRIGDATA parameter.

**QDEPTHHI**(*integer*)

The threshold against which the queue depth is compared to generate a Queue Depth High event.

This event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the QDPHIEV attribute.

The value is expressed as a percentage of the maximum queue depth (MAXDEPTH attribute), and must be greater than or equal to zero, and less than or equal to 100.

**QDEPTHLO**(*integer*)

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

This event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the QDPLOEV attribute.

The value is expressed as a percentage of the maximum queue depth (MAXDEPTH attribute), and must be greater than or equal to zero, and less than or equal to 100.

**QDPHIEV**

Controls whether Queue Depth High events are generated.

A Queue Depth High event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold (see the QDEPTHHI attribute).

**Note:** The value of this attribute can change implicitly. See the description of the Queue Depth High event in the *MQSeries Programmable System Management* manual for more details.

## ALTER QLOCAL

**ENABLED** Queue Depth High events are generated

**DISABLED** Queue Depth High events are not generated

### QDPLOEV

Controls whether Queue Depth Low events are generated.

A Queue Depth Low event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold (see the QDEPTHLO attribute).

**Note:** The value of this attribute can change implicitly. See the description of the Queue Depth Low event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Depth Low events are generated

**DISABLED** Queue Depth Low events are not generated

### QDPMAXEV

Controls whether Queue Full events are generated.

A Queue Full event indicates that a put to a queue has been rejected because the queue is full, that is, the queue depth has already reached its maximum value.

**Note:** The value of this attribute can change implicitly. See the description of the Queue Full event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Full events are generated

**DISABLED** Queue Full events are not generated

### QSVCI EV

Controls 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 at least the time indicated by the QSVCI NT attribute.

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 QSVCI NT attribute.

**Note:** The value of this attribute can change implicitly. See the description of the Service Interval High and Service Interval OK events in the *MQSeries Programmable System Management* manual for more details.

**HIGH** Service Interval High events are generated

**OK** Service Interval OK events are generated

**NONE** No service interval events are generated

### QSVCI NT(*integer*)

The service interval used for comparison to generate Service Interval High and Service Interval OK events. See the QSVCI EV attribute.

The value is in units of milliseconds, and must be greater than or equal to zero, and less than or equal to 999 999 999.

### RETINTVL(*integer*)

The number of hours (greater than or equal to zero, and less than or equal to 999 999 999) from the queue creation date and time (the date and time at which the queue was defined), after which the queue is no longer needed. The CRDATE and CRTIME can be displayed using DISPLAY QUEUE on page 141.

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 has not expired. It is the user's responsibility to take any required action.



**SCOPE**

Specifies the scope of the queue definition.

**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. To open the queue for output from some other queue manager, either the name of the owning queue manager must be specified, or the other queue manager must have a local definition of the queue.

If the SCOPE attribute of a queue is changed from CELL to QMGR, the entry for the queue is deleted from the cell directory.

**CELL** The queue definition has cell scope. This means that the queue is known to all of the queue managers in the cell, and 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 the SCOPE attribute of a queue is changed from QMGR to 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 SCOPE attribute of a dynamic queue cannot be changed to CELL.

This value is valid only if a name service supporting a cell directory (for example, the supplied DCE name service) has been configured.

This attribute is valid only on Digital OpenVMS, OS/2 Warp, UNIX systems, and Windows NT.

**STGCLASS(string)**

The name of the storage class. This is an installation-defined name.

This attribute is valid only on MVS/ESA. See the *MQSeries for MVS/ESA System Management Guide* for more details. The first character of the name must be uppercase A–Z, and subsequent characters either uppercase A–Z or numeric 0–9.

**Note:** This attribute can be changed only if the queue is empty and closed.

**TRIGDATA(string)**

The data that is inserted in the trigger message. The maximum length of the string is 64 bytes.

For a transmission queue on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, or Windows NT, you can use this parameter to specify the name of the channel to be started.

This attribute 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 greater than zero, and less than or equal to 999 999 999.

This attribute can also be changed using the **MQSET** API call.

**TRIGMPRI(integer)**

The message priority number that will trigger this queue. The value must be greater than or equal to zero, and less than or equal to the MAXPRTY queue manager attribute (see DISPLAY QMGR on page 138 for details).

This attribute can also be changed using the **MQSET** API call.

## ALTER QLOCAL

### TRIGTYPE

Whether and under what conditions a trigger message is written to the initiation queue (named by the INITQ attribute):

**FIRST** Whenever the first message of priority equal to or greater than that specified by the TRIGMPRI attribute of the queue arrives on the queue.

**EVERY** Every time a message arrives on the queue with priority equal to or greater than that specified by the TRIGMPRI attribute of the queue.

**DEPTH** When the number of messages with priority equal to or greater than that specified by TRIGMPRI is equal to the number indicated by the TRIGDPTH attribute.

**NONE** No trigger messages are written.

This attribute can also be changed using the **MQSET** API call.

### USAGE

Queue usage:

**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 until it has been successfully transmitted and stored at the remote queue manager.

## ALTER QMGR

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use ALTER QMGR to alter the queue manager attributes for the local queue manager.

**Synonym:** ALT QMGR

```

▶▶ ALTER QMGR [ qmgr attrs ] [ FORCE ]
Qmgr attrs:
  [ AUTHOREV( [ ENABLED(1) ] [ DISABLED ] ) ] [ CHAD( [ DISABLED ] [ ENABLED ] )(2) ] [ CHADEV( [ DISABLED ] [ ENABLED ] )(2) ]
  [ CHADEXIT(string)(2) ] [ DEADQ(string) ] [ DEFXMITQ(string) ] [ DESCR(string) ]
  [ INHIBTEV( [ ENABLED ] [ DISABLED ] ) ] [ LOCALEV( [ ENABLED ] [ DISABLED ] ) ] [ MAXHANDS(integer) ] [ MAXMSGL(integer)(3) ]
  [ MAXUMSGS(integer)(1) ] [ PERFMEV( [ ENABLED ] [ DISABLED ] ) ] [ REMOTEEV( [ ENABLED ] [ DISABLED ] ) ]
  [ STRSTPEV( [ ENABLED ] [ DISABLED ] ) ] [ TRIGINT(integer) ]

Notes:
1 Not valid on MVS/ESA.
2 Valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
3 Valid only on AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT.

```

## Keyword and parameter descriptions

The attributes you specify override the current values. Attributes that you do not specify are unchanged.

### Notes:

1. If you do not specify any attributes, 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 to force completion of the command if both of the following are true:

- The DEFXMITQ keyword 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 attributes

#### AUTHOREV

Whether authorization (Not Authorized) events are generated:

**ENABLED** Authorization events are generated.

This value is not supported on MVS/ESA.

**DISABLED** Authorization events are not generated. This is the queue manager's initial default value.

**CHAD** 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.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

#### CHADEV

Whether channel auto-definition events are generated.

**DISABLED** Auto-definition events are not generated. This is the queue manager's initial default value.

**ENABLED** Auto-definition events are generated.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

#### CHADEXIT(*string*)

Auto-definition exit name.

If this name is nonblank, the exit is called when an inbound request for an undefined channel is received.

The format and maximum length of the name depends on the environment:

- On OS/2 Warp, Windows, and Windows NT, 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 OS/400, it is of the form  
    *progrname libname*  
where *progrname* 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 AIX, HP-UX, and Sun Solaris, it is of the form *libraryname(functionname)*. The maximum length of the string is 128 characters.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

#### 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 MQSeries objects" on page 5.

#### DEFXMITQ(*string*)

Local name of the default transmission queue on which messages destined for a remote queue manager are put, if there is no other suitable transmission queue defined.

The queue named must be a local transmission queue. See "Rules for naming MQSeries objects" on page 5.

**DESCR**(*string*)

Plain-text comment. It provides descriptive information about the queue manager.

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.

**INHIBTEV**

Whether inhibit (Inhibit Get and Inhibit Put) events are generated:

**ENABLED** Inhibit events are generated.

**DISABLED** Inhibit events are not generated. This is the queue manager's initial default value.

**LOCALEV**

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.

**MAXHANDS**(*integer*)

The maximum number of open handles that any one task can have at the same time.

Do not specify a value less than zero or greater than 999 999 999.

**MAXMSGL**(*integer*)

The maximum length of messages allowed on queues for this queue manager.

This is in the range 32 KB through 100 MB. The default is 4 MB.

If you reduce the maximum message length for the queue manager, you should also reduce the maximum message length of the SYSTEM.DEFAULT.LOCAL.QUEUE definition, and all other queues connected to the queue manager. This ensures that the queue manager's limit is not less than that of any of the queues associated with it. If you do not do this, and applications inquire only the value of the queue's MAXMSGL, they might not work correctly.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT.

**MAXUMSGS**(*integer*)

The maximum number of uncommitted messages within a syncpoint.

This is a limit on

- the number of messages that can be retrieved, plus
- the number of messages that can be put

within any one syncpoint. It does not apply to messages that are put or retrieved outside syncpoint. The number includes any trigger messages and report messages generated within the same unit of recovery.

Specify a value in the range 1 through 999 999 999.

This attribute is not supported on MVS/ESA. See the DEFINE MAXSMSGS command instead.

**PERFMEV**

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.

## ALTER QMGR

### REMOTEEV

Whether remote error events are generated:

**ENABLED** Remote error events are generated.

**DISABLED** Remote error events are not generated. This is the queue manager's initial default value.

### STRSTPEV

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.

### TRIGINT(*integer*)

A time interval expressed in milliseconds.

The TRIGINT attribute is relevant only if the trigger type (TRIGTYPE) is set to FIRST (see DEFINE QLOCAL on page 92 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 (see *MQSeries Application Programming Guide*), 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.

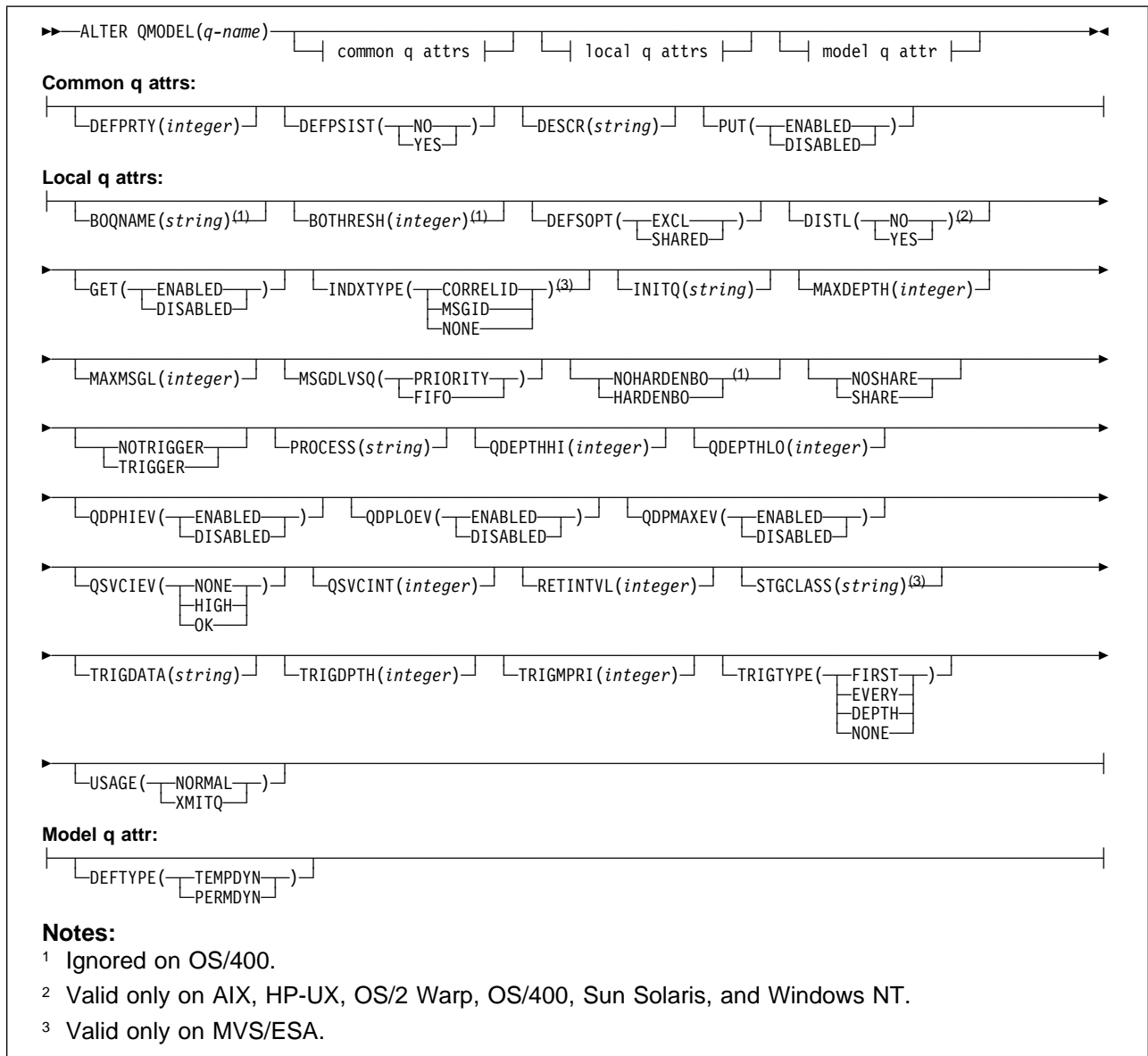
Do not specify a value less than zero or greater than 999 999 999.

## ALTER QMODEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use ALTER QMODEL to alter the attributes of a model queue.

**Synonym:** ALT QM



### Keyword and parameter descriptions

You must specify which model queue you want to alter.

The attributes that you specify override the current values. Attributes that you do not specify are unchanged.

*(q-name)*

The local name of the model queue to be altered (see “Rules for naming MQSeries objects” on page 5). The name must be defined to the local queue manager.

### Common queue attributes

**DEFPRTY***(integer)*

The default priority of messages put on the queue. The value must be greater than or equal to zero, (the lowest priority) and less than or equal to the MAXPRTY queue manager attribute. MAXPRTY can be displayed using the DISPLAY QMGR command. (MAXPRTY is 9.)

**DEFPSIST**

The default message persistence for this queue:

**NO** Messages on this queue are lost across a restart of the queue manager. This is the default supplied with MQSeries, but your installation might have changed it.

**YES** Messages on this queue survive a restart of the queue manager.

Persistent messages are not allowed on a temporary dynamic queue. In order to avoid an error if a message is put to such a queue with default persistence, do not set the DEFPSIST attribute to YES for model queue definitions that have a DEFTYPE of TEMPDYN.

**DESCR***(string)*

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY QUEUE command.

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.

**PUT** Whether messages can be put on the queue.

**ENABLED** Messages can be added to the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Messages cannot be added to the queue.

This attribute can also be changed using the **MQSET** API call.

### Local queue attributes

**BOQNAME***(string)*

The excessive backout requeue name. Apart from maintaining a value for this attribute, the queue manager takes no action based on its value. On OS/400, the keyword is ignored.

**BOTHRESH***(integer)*

The backout threshold. Apart from maintaining a value for this attribute, the queue manager takes no action based on its value. On OS/400, the keyword is ignored.

Specify a value greater than or equal to zero, and less than or equal to 999 999 999.



**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

**DISTL** 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 should not normally change this attribute, because it is set by the MCA. However you can set this attribute when defining a transmission queue if the distribution list capability of the destination queue manager is known.

This keyword is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

**GET** Whether applications are to be permitted to get messages from this queue:

**ENABLED** Messages can be retrieved from the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Applications cannot retrieve messages from the queue.

This attribute can also be changed using the **MQSET** API call.

**INDXTYPE**

The type of index maintained by the queue manager to expedite **MQGET** operations on the queue:

**NONE** No index is maintained. Use this when messages are usually 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 this when messages are usually 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 this when messages are usually retrieved using the correlation identifier as a selection criterion on the **MQGET** call with the message identifier set to NULL.

The **INDXTYPE** attribute can be changed at any time, and the change takes effect immediately if all the following conditions are satisfied:

- No applications have the queue open
- The queue is empty
- There are no uncommitted MQPUT or MQGET operations outstanding against the queue

If these conditions are not satisfied, the attribute is changed immediately, but the index is not rebuilt until the next time the queue manager is restarted. The reply sent by the queue manager indicates if this is the case.

This attribute is supported only on MVS/ESA. On other platforms, retrieval optimization might be provided, but it is not controlled by a queue attribute.

**INITQ(string)**

The local name of a local queue (known as the *initiation queue*) on this queue manager, to which trigger messages relating to this queue are written. See "Rules for naming MQSeries objects" on page 5.

## ALTER QMODEL

### MAXDEPTH(*integer*)

The maximum number of messages allowed on the queue. Specify a value greater than or equal to zero, and less than or equal to:

- 999 999 999 if the queue is on MVS/ESA
- 640 000 if the queue is on any other MQSeries platform

Other factors might still cause the queue to be treated as full, for example, if there is no further DASD space available.

If this value is reduced, any messages that are already on the queue that cause the new maximum to be exceeded remain intact.

### MAXMSGL(*integer*)

The maximum length of messages on this queue. On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the MAXMSGL parameter of the ALTER QMGR command for more information. On other platforms, specify a value greater than or equal to zero, and less than or equal to 4 194 304.

For a transmission queue, this value includes the space required for headers. It is recommended that the value should be at least 4000 bytes larger than the maximum expected length of user data in any message that could be put on a transmission queue.

If this value is reduced, any messages that are already on the queue, whose length exceeds the new maximum, are not affected.

Applications can use this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore, the value should only be reduced if it is known that this will not cause an application to operate incorrectly.

### MSGDLVSEQ

Message delivery sequence:

**PRIORITY** Messages are delivered (in response to **MQGET** API calls) in first-in-first-out (FIFO) order within priority. This is the default supplied with MQSeries, but your installation might have changed it.

**FIFO** Messages are delivered (in response to **MQGET** API calls) in FIFO order. Priority is ignored for messages on this queue.

If the message delivery sequence is changed from PRIORITY to FIFO while there are messages on the queue, the order of the messages already enqueued is not changed. Messages added to the queue subsequently 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 enqueued while the queue was set to FIFO take the default priority.

### NOHARDENBO and HARDENBO

Whether hardening should be used to ensure that the count of the number of times that a message has been backed out is accurate.

**Note:** The count is always hardened on OS/400; the HARDENBO and NOHARDENBO keywords are ignored if specified.

**NOHARDENBO** The count is not hardened. This is the default supplied with MQSeries, but your installation might have changed it.

**HARDENBO** The count is hardened.

**NOSHARE and SHARE**

Whether multiple applications can get messages from this queue:

**NOSHARE** A single application instance only can get messages from the queue

**SHARE** More than one application instance can get messages from the queue

**NOTRIGGER and TRIGGER**

Whether trigger messages are written to the initiation queue (named by the INITQ attribute) to trigger the application (named by the PROCESS attribute):

**NOTRIGGER** Triggering is not active, and trigger messages are not written to the initiation queue. This is the default supplied with MQSeries, but your installation might have changed it.

**TRIGGER** Triggering is active, and trigger messages are written to the initiation queue.

This attribute can also be changed using the **MQSET** API call.

**PROCESS(string)**

The local name of the MQSeries process. This 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 MQSeries objects" on page 5.

The process does not have to be defined 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 gives the name of the channel to be started. This parameter is optional for transmission queues on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT; if you do not specify it, the channel name is taken from the value specified for the TRIGDATA parameter.

**QDEPTHHI(integer)**

The threshold against which the queue depth is compared to generate a Queue Depth High event.

This event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the QDPHIEV attribute.

The value is expressed as a percentage of the maximum queue depth (MAXDEPTH attribute), and must be greater than or equal to zero, and less than or equal to 100.

**QDEPTHLO(integer)**

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

This event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the QDPLOEV attribute.

The value is expressed as a percentage of the maximum queue depth (MAXDEPTH attribute), and must be greater than or equal to zero, and less than or equal to 100.

**QDPHIEV**

Controls whether Queue Depth High events are generated.

A Queue Depth High event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold (see the QDEPTHHI attribute).

**Note:** The value of this attribute can change implicitly. See the description of the Queue Depth High event in the *MQSeries Programmable System Management* manual for more details.

## ALTER QMODEL

**ENABLED** Queue Depth High events are generated

**DISABLED** Queue Depth High events are not generated

### QDPLOEV

Controls whether Queue Depth Low events are generated.

A Queue Depth Low event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold (see the QDEPTHLO attribute).

**Note:** The value of this attribute can change implicitly. See the description of the Queue Depth Low event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Depth Low events are generated

**DISABLED** Queue Depth Low events are not generated

### QDPMAXEV

Controls whether Queue Full events are generated.

A Queue Full event indicates that a put to a queue has been rejected because the queue is full, that is, the queue depth has already reached its maximum value.

**Note:** The value of this attribute can change implicitly. See the description of the Queue Full event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Full events are generated

**DISABLED** Queue Full events are not generated

### QSVCI EV

Controls 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 at least the time indicated by the QSVCI NT attribute.

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 QSVCI NT attribute.

**Note:** The value of this attribute can change implicitly. See the description of the Service Interval High and Service Interval OK events in the *MQSeries Programmable System Management* manual for more details.

**HIGH** Service Interval High events are generated

**OK** Service Interval OK events are generated

**NONE** No service interval events are generated

### QSVCI NT(*integer*)

The service interval used for comparison to generate Service Interval High and Service Interval OK events. See the QSVCI EV attribute.

The value is in units of milliseconds, and must be greater than or equal to zero, and less than or equal to 999 999 999.

### RETINTVL(*integer*)

The number of hours (greater than or equal to zero, and less than or equal to 999 999 999) from the queue creation date and time (the date and time at which the queue was defined), after which the queue is no longer needed. The CRDATE and CRTIME can be displayed using DISPLAY QUEUE on page 141.

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 has not expired. It is the user's responsibility to take any required action.

**STGCLASS**(*string*)

The name of the storage class. This is an installation-defined name.

This attribute is valid only on MVS/ESA. See the *MQSeries for MVS/ESA System Management Guide* for more details. The first character of the name must be uppercase A–Z, and subsequent characters either uppercase A–Z or numeric 0–9.

**Note:** This attribute can be changed only if the queue is empty and closed.

**TRIGDATA**(*string*)

The data that is inserted in the trigger message. The maximum length of the string is 64 bytes.

For a transmission queue on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, or Windows NT, you can use this parameter to specify the name of the channel to be started.

This attribute 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 greater than zero, and less than or equal to 999 999 999.

This attribute can also be changed using the **MQSET** API call.

**TRIGMPRI**(*integer*)

The message priority number that will trigger this queue. The value must be greater than or equal to zero, and less than or equal to the MAXPRTY queue manager attribute (see DISPLAY QMGR on page 138 for details).

This attribute can also be changed using the **MQSET** API call.

**TRIGTYPE**

Whether and under what conditions a trigger message is written to the initiation queue (named by the INITQ attribute):

**FIRST** Whenever the first message of priority equal to or greater than that specified by the TRIGMPRI attribute of the queue arrives on the queue.

**EVERY** Every time a message arrives on the queue with priority equal to or greater than that specified by the TRIGMPRI attribute of the queue.

**DEPTH** When the number of messages with priority equal to or greater than that specified by TRIGMPRI is equal to the number indicated by the TRIGDPTH attribute.

**NONE** No trigger messages are written.

This attribute can also be changed using the **MQSET** API call.

**USAGE**

Queue usage:

**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 until it has been successfully transmitted and stored at the remote queue manager.

## ALTER QMODEL

### Model queue attribute

#### DEFTYPE

Queue definition type:

**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).

Do not specify this value for a model queue definition with a DEFPSIST attribute of YES.

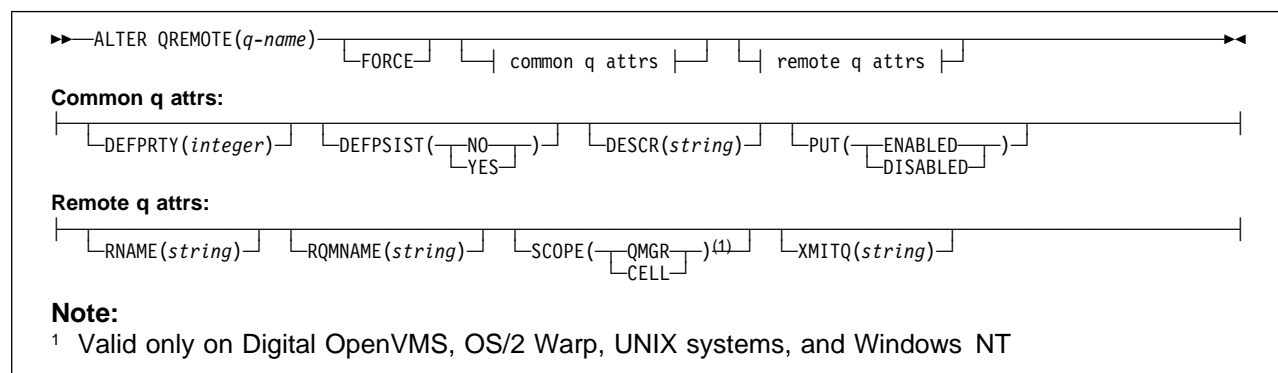
**PERMDYN** A permanent dynamic queue is created when an application issues an **MQOPEN** API call with the name of this model queue specified in the object descriptor (MQOD).

## ALTER QREMOTE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use ALTER QREMOTE to alter the attributes of a local definition of a remote queue, a queue-manager alias, or a reply-to queue alias.

**Synonym:** ALT QR



## Keyword and parameter descriptions

You must specify which remote queue you want to alter.

The attributes you specify override the current values. Attributes that you do not specify are unchanged.

*(q-name)*

The name of the local definition of the remote queue to be altered (see “Rules for naming MQSeries objects” on page 5). The name must be defined to the local queue manager.

**FORCE** Specify this to force completion of the command if both of the following are true:

- The XMITQ attribute is changed
- One or more applications has this queue open as a remote queue

If FORCE is not specified in these circumstances, the command is unsuccessful.

FORCE is also needed if both of the following are true:

- Any of the RNAME, RQMNAME, or XMITQ keywords is changed
- One or more applications has a queue open which resolved through this definition as a queue-manager alias

Again, if FORCE is not specified in these circumstances, the command is unsuccessful.

**Note:** FORCE is not required if this definition is in use as a reply-to queue alias only.

## Common queue attributes

**DEFPRTY**(*integer*)

The default priority of messages put on the queue. The value must be greater than or equal to zero, (the lowest priority) and less than or equal to the MAXPRTY queue manager attribute. MAXPRTY can be displayed using the DISPLAY QMGR command. (MAXPRTY is 9.)

## ALTER QREMOTE

### DEFPSIST

The default message persistence for this queue:

- NO** Messages on this queue are lost across a restart of the queue manager. This is the default supplied with MQSeries, but your installation might have changed it.
- YES** Messages on this queue survive a restart of the queue manager.

### DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY QUEUE command.

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.

**PUT** Whether messages can be put on the queue.

**ENABLED** Messages can be added to the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Messages cannot be added to the queue.

This attribute can also be changed using the **MQSET** API call.

## Remote queue attributes

### RNAME(*string*)

Name of remote queue. This is the local name of the queue as defined on the queue manager specified by RQMNAME. The name is *not* checked to ensure that it contains only those characters normally allowed for queue names (see "Rules for naming MQSeries objects" on page 5).

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.

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.

### RQMNAME(*string*)

The name of the remote queue manager on which the queue RNAME is defined.

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, then 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 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 name is *not* checked to ensure that it contains only those characters normally allowed for queue names (see "Rules for naming MQSeries objects" on page 5).



**SCOPE**

Specifies the scope of the queue definition.

**QMGR** The queue definition has queue-manager scope. This means that the definition of the queue does not extend beyond the queue manager which owns it. To open the queue for output from some other queue manager, either the name of the owning queue manager must be specified, or the other queue manager must have a local definition of the queue.

**CELL** The queue definition has cell scope. This means that the queue is known to all of the queue managers in the cell, and 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.

This value is valid only if a name service supporting a cell directory (for example, the supplied DCE name service) has been configured.

This attribute is only supported on Digital OpenVMS, OS/2 Warp, UNIX systems, and Windows NT.

**XMITQ**(*string*)

The name of the transmission queue to be used for forwarding messages to the remote queue, for either a remote queue or for a queue-manager alias definition.

If XMITQ is blank, a queue with the same name as RQMNAME is used instead as the transmission queue.

This attribute is ignored if the definition is being used as a queue-manager alias and RQMNAME is the name of the local queue manager.

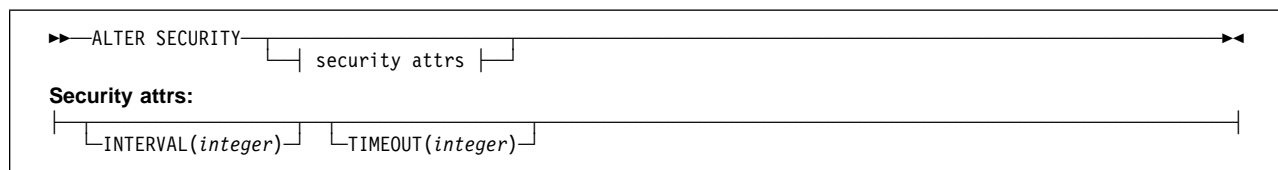
It is also ignored if the definition is used as a reply-to queue alias definition.

## ALTER SECURITY

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use ALTER SECURITY to define system-wide security options.

**Synonym:** ALT SEC



### Keyword and parameter descriptions

The attributes you specify override the current attribute values. Attributes that you do not specify are unchanged.

**Note:** If you do not specify any attributes, the command completes successfully, but no security options are changed.

#### **INTERVAL**(*integer*)

The interval between checks for user IDs for which the TIMEOUT has expired. The value is in minutes, in the range 0–10080 (one week). If INTERVAL is specified as 0, no user time-outs occur.

#### **TIMEOUT**(*integer*)

How long an unused, user ID can remain in the MQSeries subsystem. The value specifies a number of minutes in the range 0–10080 (one week). If TIMEOUT is specified as 0, and INTERVAL is nonzero, then all users are signed off within the queue manager every INTERVAL number of minutes.

The length of time that an unused user ID can remain depends on the value of INTERVAL. The user ID times out at a time between TIMEOUT and TIMEOUT plus INTERVAL.

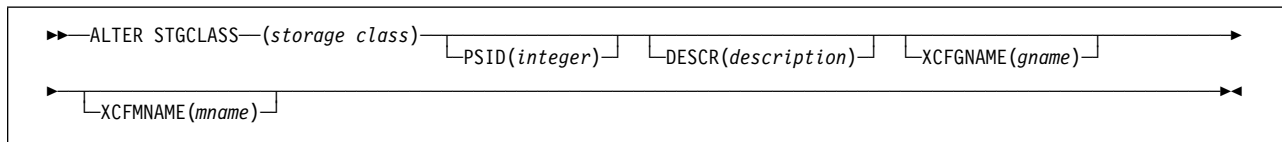
When the TIMEOUT and INTERVAL attributes are changed, the previous timer request is canceled and a new timer request is scheduled immediately, using the new TIMEOUT value. When the timer request is actioned, a new value for INTERVAL is set.

## ALTER STGCLASS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use ALTER STGCLASS to alter the characteristics of a storage class.

**Synonym:** ALT STC



### Keyword and parameter descriptions

You can issue the ALTER STGCLASS command for a given storage class as long as that storage class already exists and is not active. This command will succeed only if all the MQSeries queues that reference the storage class are empty and closed. All parameters can be changed as part of the command.

#### *(storage-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.

#### PSID(*integer*)

The page set identifier that this storage class is to be associated with. No check is made that the page set has been defined; an error will be raised only when you try to put a message to a queue that specifies this storage class (MQRC\_PAGESET\_ERROR).

This is a number in the range 00 through 99.

#### DESCR(*description*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY STGCLASS command.

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). 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.

#### XCFGNAME(*group name*)

If you are using the IMS bridge, this is the name of the XCF group to which the IMS system belongs. (This is the group name specified in the IMS parameter list.)

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. If you want to alter this value to blank, you must also alter the value of XCFMNAME to blank, and enclose the blank characters in single quotation marks, as shown below:

```
ALT STGCLASS(X) XCFMNAME(' ') XCFGNAME(' ')
```

## ALTER STGCLASS

### **XCFMNAME**(*member name*)

If you are using the IMS bridge, this is the XCF member name of the IMS system within the XCF group specified in XCFGNAME. (This is the member name specified in the IMS parameter list.)

This is 1 through 16 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 through 9.

If you want to alter this value to blank, you must also alter the value of XCFGNAME to blank, and enclose the blank characters in single quotation marks, as shown above.

### **Usage notes**

- The resultant values of XCFGNAME and XCFMNAME must either both be blank or both be nonblank.

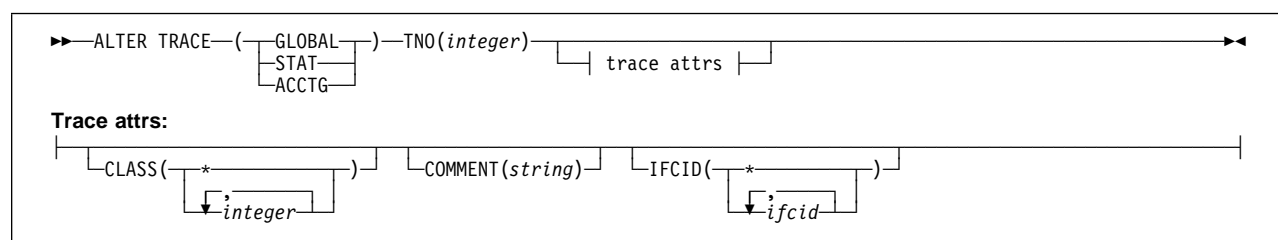
## ALTER TRACE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use ALTER TRACE to change the trace events (IFCIDs) being traced for a particular active trace. ALTER TRACE stops the specified trace, and restarts it with the altered attributes.

**Note:** ALTER TRACE does not affect any RMID(231) settings (although a subsequent DISPLAY TRACE command will show them altered).

**Synonym:** ALT TRACE



## Keyword and parameter descriptions

The trace type you specify determines which IFCIDs are activated. For further descriptions of each trace type, see START TRACE on page 167.

Specify one of the following:

**GLOBAL** Service data from the entire MQSeries subsystem (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. This limits the list to a particular trace, identified by its trace number (1 through 32). You can specify only one trace number.

### Trace attributes

#### CLASS(integer)

The trace class to be altered. This limits the list to IFCIDs activated for particular classes. See START TRACE on page 167 for a list of allowed classes. A range of classes can be specified as m:n (for example, CLASS(01:03)). CLASS(\*) activates all default IFCID 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)

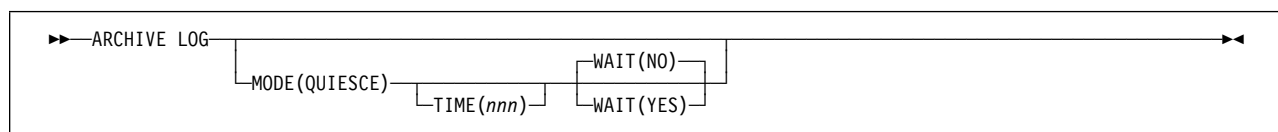
The events to be traced. This specifies the optional IFCIDs to activate. All IFCIDs and classes specified are activated for the trace type specified.

## ARCHIVE LOG

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use ARCHIVE LOG as part of your backup procedure. It takes a copy of the current *active* log following the latest syncpoint.

**Synonym:** ARC LOG



## Keyword and parameter descriptions

The ARCHIVE LOG command takes a copy of the active log, or both logs if you are using dual logging. All the parameters are optional.

### MODE(QUIESCE)

Stops any new update activity on the MQSeries subsystem for a specified period of time, and brings all existing users to a point of consistency after a commit. When the specified period of time expires, archiving of the current active log takes place.

The period of time specified is the maximum time that MQSeries has to attempt a full subsystem quiesce.

If MODE(QUIESCE) is issued without the TIME parameter, the value in the QUIESCE parameter of the CSQ6ARVP macro is used as the quiesce time period.

### TIME(nnn)

Overrides the quiesce time period specified by the QUIESCE parameter of the CSQ6ARVP 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 period of time 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
- MQSeries lock contention might develop
- A time-out might interrupt the quiesce

**WAIT** Specifies whether MQSeries is to wait until the quiesce process has finished before returning to the issuer of the ARCHIVE LOG command, or not.

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. This makes the quiesce process asynchronous to the issuer; you can issue further MQSeries 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. This makes the quiesce process synchronous to the issuer; further MQSeries commands are not processed until the ARCHIVE LOG command finishes.

**Notes:**

1. You cannot issue an ARCHIVE LOG command while a previous ARCHIVE LOG command is in progress.
2. 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 MQSeries would halt all processing until an off-load had been completed.
3. You can issue an ARCHIVE LOG 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.
4. You can issue a DISPLAY THREAD command to discover whether an ARCHIVE LOG command is active. The DISPLAY command returns message CSQV400I if an ARCHIVE LOG command is active.

## CLEAR QLOCAL

---

### CLEAR QLOCAL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√		√	√	√	√	√

Use CLEAR QLOCAL to clear the messages from a local queue.

**Synonym:** CLEAR QL

►—CLEAR QLOCAL(*q-name*)—◄

### Keyword and parameter descriptions

You must specify which local queue you want to clear.

The command fails if either:

- The queue has uncommitted messages
- The queue is currently open by an application (with any open options)

*(q-name)*

The name of the local queue to be cleared. The name must be defined to the local queue manager.

If an application has this queue open (with any open options), 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.



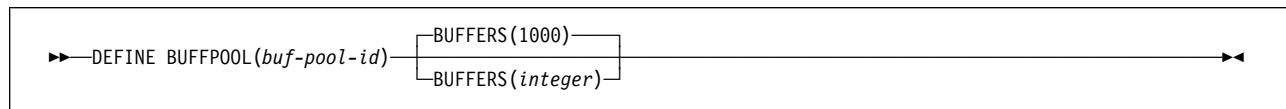
## DEFINE BUFFPOOL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DEFINE BUFFPOOL to define a buffer pool that is used for holding messages in main storage.

DEFINE BUFFPOOL can be issued only from the CSQINP1 initialization data set.

**Synonym:** DEF BP



### Keyword and parameter descriptions

If this command is not issued, the default number of buffers is assumed. If more than one DEFINE BUFFPOOL command is issued for the same buffer pool, only the *last* one is actioned.

*(buf-pool-id)*

Buffer pool identifier. This is required.

This is an integer in the range 0 through 3.

**BUFFERS***(integer)*

The number of 4096-byte buffers to be used in this buffer pool. This is optional. The default number of buffers is 1000, and the minimum is 100. The maximum number of buffers for all the buffer pools is determined by the amount of storage available in the MQSeries address space.

## DEFINE CHANNEL

---

### DEFINE CHANNEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

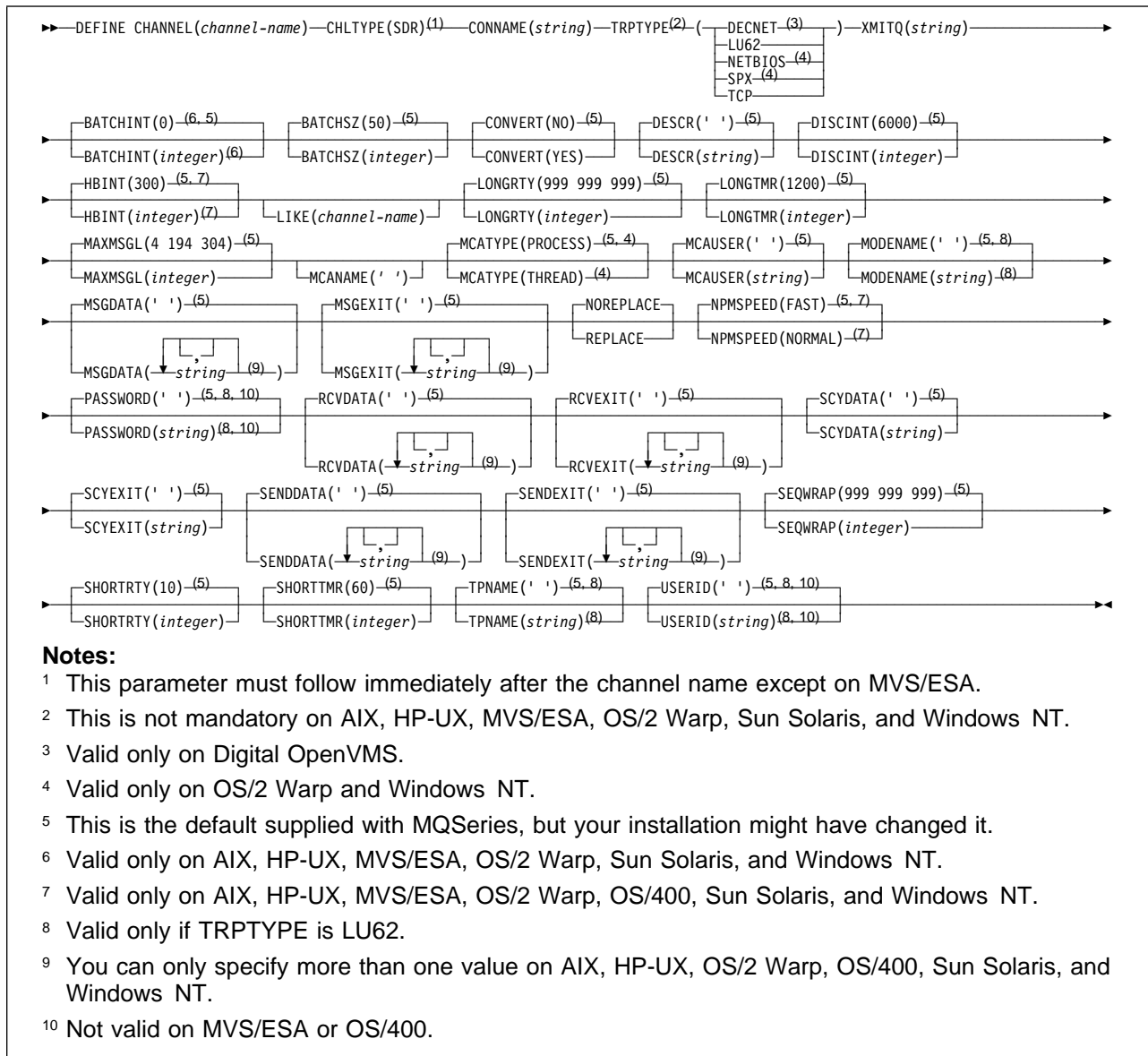
Use DEFINE CHANNEL to define a new channel, and set its attributes.

**Notes:**

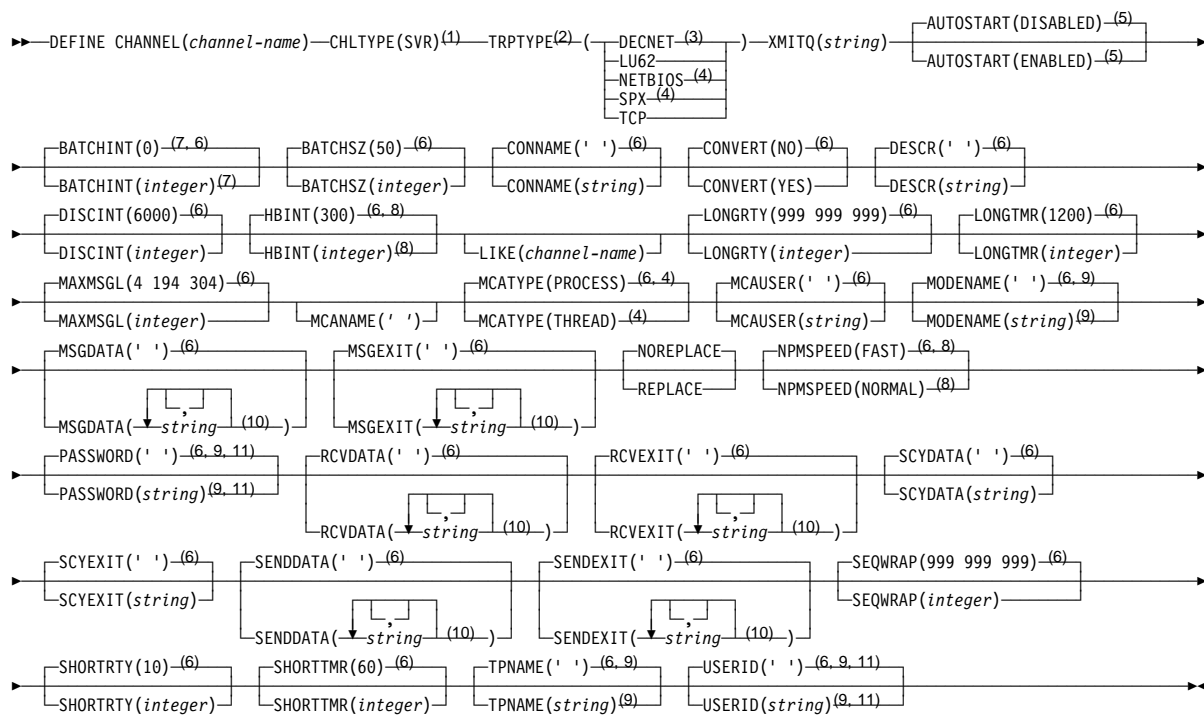
1. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.
2. There is a separate syntax diagram for each of the six types of channel.

**Synonym:** DEF CHL

## Sender channel



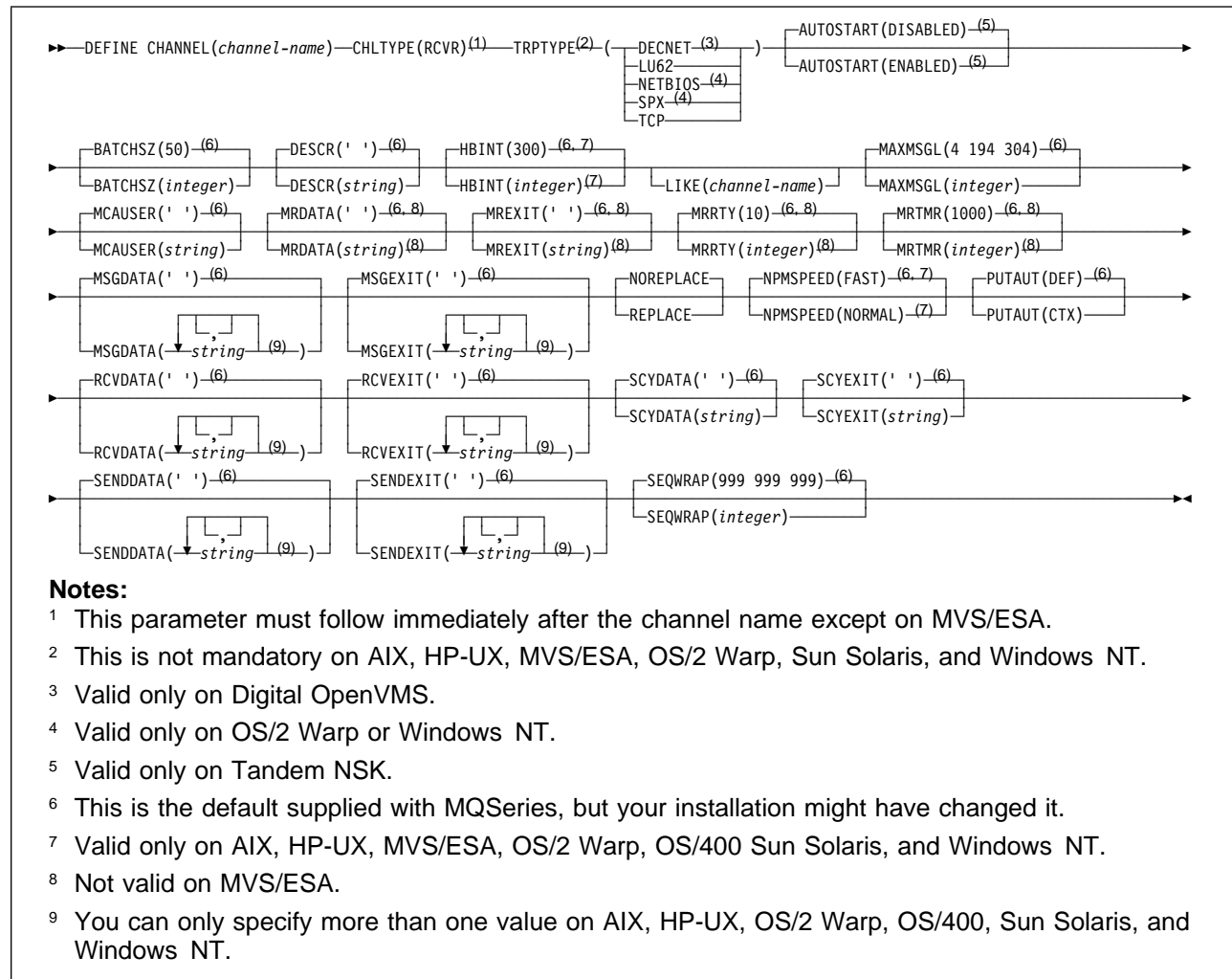
Server channel



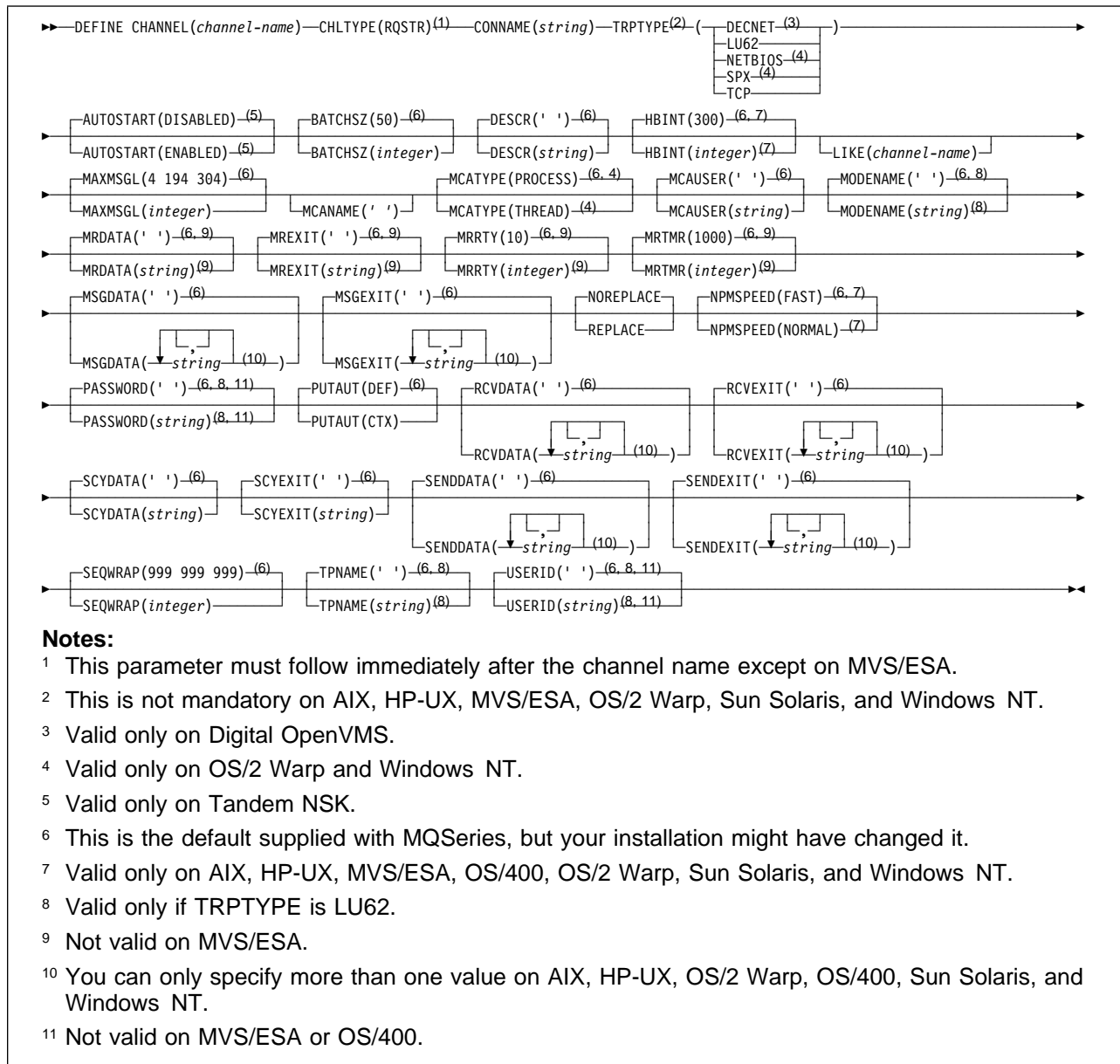
Notes:

- 1 This parameter must follow immediately after the channel name except on MVS/ESA.
- 2 This is not mandatory on AIX, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT.
- 3 Valid only on Digital OpenVMS.
- 4 Valid only on OS/2 Warp and Windows NT.
- 5 Valid only on Tandem NSK.
- 6 This is the default supplied with MQSeries, but your installation might have changed it.
- 7 Valid only on AIX, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT.
- 8 Valid only on AIX, HP-UX, MVS/ESA, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
- 9 Valid only if TRPTYPE is LU62.
- 10 You can only specify more than one value on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
- 11 Not valid on MVS/ESA or OS/400.

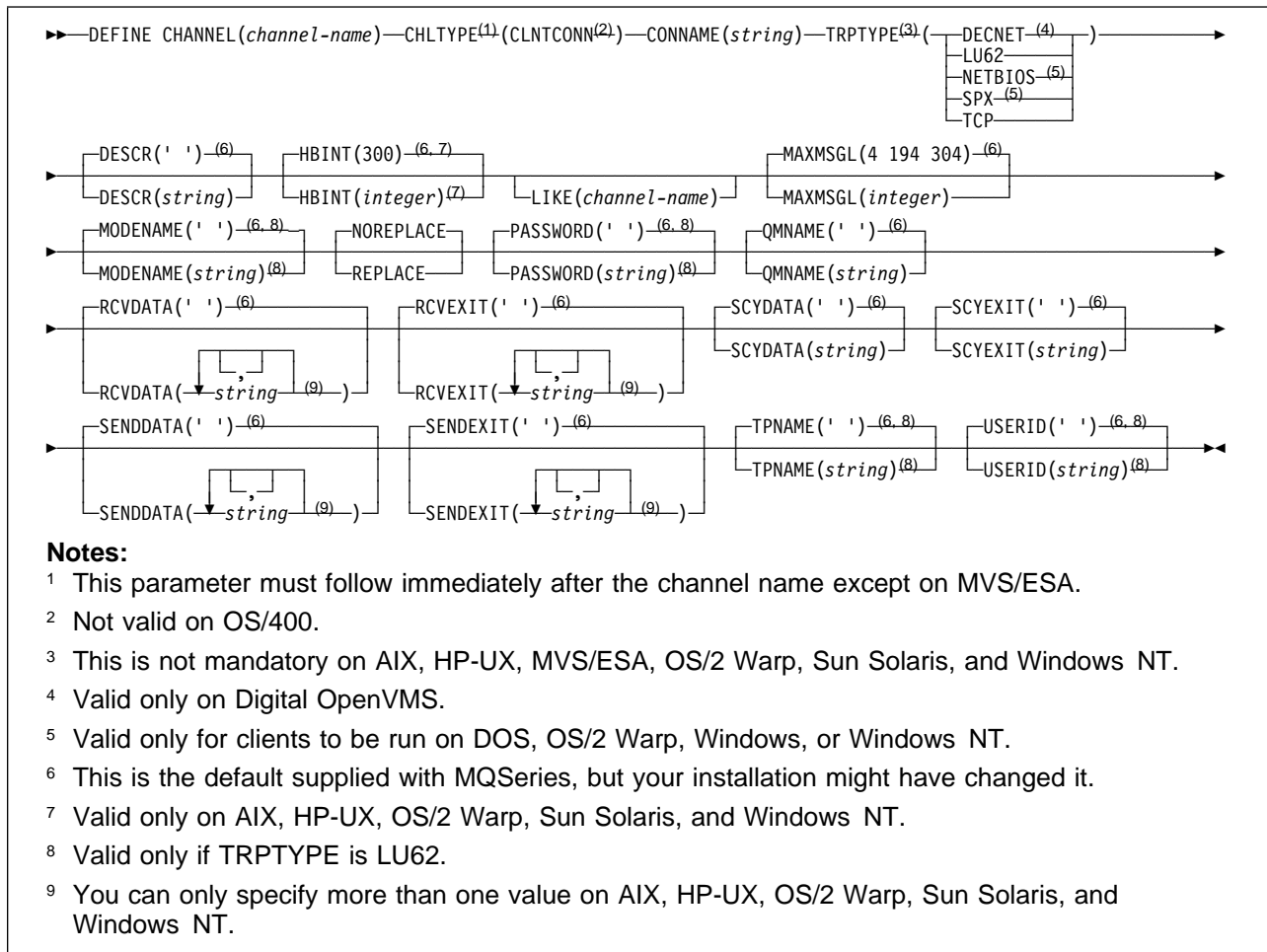
## Receiver channel



Requester channel

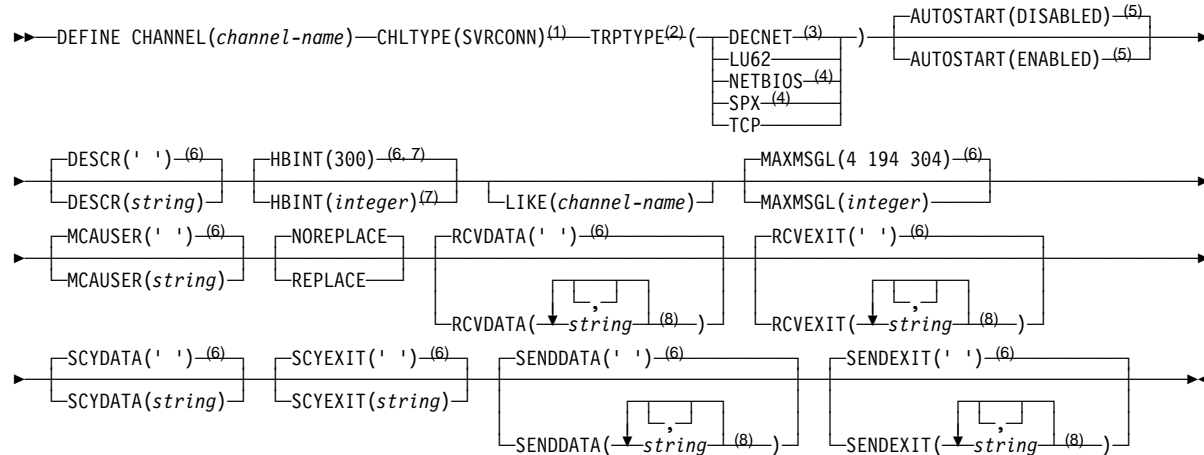


## Client-connection channel



## DEFINE CHANNEL

### Server-connection channel



#### Notes:

- 1 This parameter must follow immediately after the channel name except on MVS/ESA.
- 2 This is not mandatory on AIX, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT.
- 3 Valid only on Digital OpenVMS.
- 4 Valid only for clients to be run on DOS, OS/2 Warp, Windows, or Windows NT.
- 5 Valid only on Tandem NSK.
- 6 This is the default supplied with MQSeries, but your installation might have changed it.
- 7 Valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
- 8 You can only specify more than one value on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

### Keyword and parameter descriptions

Parameters are optional unless the description states that they are required.

*(channel-name)*

The name of the new channel definition. This is required.

The name must not be the same as any existing channel defined on this queue manager (unless REPLACE is specified). On MVS/ESA, 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 MQSeries objects" on page 5.

#### AUTOSTART

Specifies whether an LU 6.2 responder process for the channel will be started at queue manager startup.

**ENABLED** The responder is started

**DISABLED** The responder is not started (this is the default)

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR, RQSTR, SVR, and SVRCONN. It is supported only on Tandem NSK.



**BATCHINT**(*integer*)

The minimum amount of time, in milliseconds, that a channel will keep a batch open.

The batch is terminated by whichever of the following occurs first:

- BATCHSZ messages have been sent, or
- The transmission queue is empty and BATCHINT is exceeded

The default value is zero, 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 greater than or equal to zero, and less than or equal to 999 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR. It is valid only on AIX, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT.

**BATCHSZ**(*integer*)

The maximum number of messages that can be sent through a channel before taking a checkpoint.

The maximum batch size actually used is the lowest of the following:

- The BATCHSZ of the sending channel
- The BATCHSZ of the receiving channel
- The maximum number of uncommitted messages allowed at the sending queue manager
- The maximum number of uncommitted messages allowed at the receiving queue manager

The maximum number of uncommitted messages is specified by the MAXUMSGS parameter of the ALTER QMGR command, or the DEFINE MAXSMMSG command on MVS/ESA.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, RCVR, or RQSTR.

The value must be greater than zero, and less than or equal to 9999.

**CHLTYPE**

Channel type. This is required. It must follow immediately after the (*channel-name*) parameter on all platforms except MVS/ESA.

<b>SDR</b>	Sender channel
<b>SVR</b>	Server channel
<b>RCVR</b>	Receiver channel
<b>RQSTR</b>	Requester channel
<b>CLNTCONN</b>	Client-connection channel (not valid on OS/400)
<b>SVRCONN</b>	Server-connection channel

**Note:** If you are using the REPLACE option, you cannot change the channel type.

**CONNNAME**(*string*)

Connection name.

The connection name of the partner. (The maximum length is 48 characters on MVS, and 264 characters on other platforms.) The type of name depends on the transport type (TRPTYPE) to be used:

**DECnet Phase IV**

The DECnet node name and the DECnet object name, in the form:

CONNNAME('node\_name(object\_name)')

## DEFINE CHANNEL

### LU 6.2

- On Digital OpenVMS this is the gateway node, access name, and the tpname that is used by SNA to invoke the remote program. The format of this information is as follows:  
`CONNAME('gateway_node.access_name(tpname)')`
- On MVS/ESA this is the symbolic destination name for the partner LU, as defined in the side information data set.
- On OS/2 Warp it is the fully-qualified name of the partner Logical Unit.
- On OS/400, Windows NT, and UNIX systems (except SunOS), this is the name of the CPI-C communications side object.
- On SunOS, this is the name of the gateway on which the queue manager is running.
- On Tandem NSK, the value of this depends on whether SNAX or ICE is used as the communications protocol:

- If SNAX is used:

- For sender, requester, and fully qualified server channels, this is the process name of the SNAX/APC process, the name of the local LU, and the name of the partner LU on the remote machine, for example:

```
CONNAME('çPPPP.LOCALLU.REMOTELU')
```

- For receiver and non fully qualified server channels, this is the process name of the SNAX/APC process and the name of the local LU, for example:

```
CONNAME('çPPPP.LOCALLU')
```

The name of the local LU can be an asterisk (\*), indicating any name.

- If ICE is used:

- For sender, requester, and fully qualified server channels, this is the process name of the ICE process, the ICE open name, the name of the local LU, and the name of the partner LU on the remote machine, for example:

```
CONNAME('çPPPP.#OPEN.LOCALLU.REMOTELU')
```

For receiver and non fully qualified server channels, this is the process name of the SNAX/APC process, the ICE open name, and the name of the local LU, for example:

```
CONNAME('çPPPP.#OPEN.LOCALLU')
```

The name of the local LU can be an asterisk (\*), indicating any name.

### 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)')
```

If the socket number is omitted, the MQSeries default value (X'5e86') is assumed.

### TCP/IP

Either the host name, or the network address of the remote machine. This can be followed by an optional port number, enclosed in parentheses.

This parameter is required for channels with a channel type (CHLTYPE) of SDR, RQSTR, or CLNTCONN. It is optional for SVR channels, and is not valid for RCVR or SVRCONN channels.

**Note:** If you are using any of the special characters in your connection name (for example, parentheses) you must enclose the string in single quotes.

### CONVERT

Specifies whether the sending message channel agent should attempt conversion of the application message data, if the receiving message channel agent is unable to perform this conversion.

**NO** No conversion by sender

**YES** Conversion by sender

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### DESCR(*string*)

Plain-text comment. It provides descriptive information about the channel when an operator issues the DISPLAY CHANNEL command.

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.

### 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 greater than or equal to zero, and less than or equal to 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### HBINT(*integer*)

This parameter has a different interpretation depending upon the channel type, as follows:

- For a channel type of SDR, SVR, RCVR, or RQSTR, this is 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 type of heartbeat is valid only on AIX, HP-UX, MVS/ESA, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

**Note:** You should set this value to be significantly less than the value of DISCINT. MQSeries checks only that it is within the permitted range however.

- For a channel type of SVRCONN or CLNTCONN, this is the time, in seconds, between heartbeat flows passed from the server MCA when that MCA has issued an MQGET with WAIT on behalf of a client application. This allows the server to handle situations where the client connection fails during an MQGET with WAIT. This type of heartbeat is valid only for AIX, HP-UX, OS/2 Warp, OS/400 (SVRCONN only), Sun Solaris, and Windows NT.

The value must be in the range zero through 999 999. A value of zero means that no heartbeat exchange takes place. The value that is used is the larger of the values specified at the sending side and the receiving side.

## DEFINE CHANNEL

### LIKE(*channel-name*)

The name of a channel, whose attributes will be used to model this definition.

If this field is not filled in, and you do not complete the attribute fields related to the command, the values are taken from one of the following, 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 (not valid on OS/400)
SYSTEM.DEF.CLNTCONN	Client-connection channel (not valid on OS/400)

This is equivalent to defining the following object:

```
LIKE(SYSTEM.DEF.SENDER)
```

for a sender channel, and similarly for other channel types.

These default channel definitions can be altered by the installation to the default values required.

### LONGRTY(*integer*)

When a sender or server channel is attempting to connect to the remote queue manager, and the count specified by SHORTRTY has been exhausted, this 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 subsequently be restarted with a command (it is not started automatically by the channel initiator), and it then makes only one attempt to connect, because 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.

The value must be greater than or equal to zero, and less than or equal to 999 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### LONGTMR(*integer*)

For long retry attempts, this is the maximum number of seconds to wait before re-attempting 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 greater than or equal to zero, and less than or equal to 999 999 999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999 999; values exceeding this will be treated as 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### MAXMSGL(*integer*)

Specifies the maximum message length that can be transmitted on the channel. This is compared with the value for the partner and the actual maximum used is the lower of the two values.

The value zero means the maximum message length for the queue manager.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the MAXMSGL parameter of the ALTER QMGR command for more information. On other platforms, specify a value greater than or equal to zero, and less than or equal to 4 194 304.

**MCANAME**(*string*)

Message channel agent name.

This is reserved, and if specified must only be set to blanks (maximum length 20 characters).

**MCATYPE**

Specifies whether the message-channel-agent program should run 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

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, or RQSTR. It is supported only on OS/2 Warp and Windows NT.

**MCAUSER**(*string*)

Message channel agent user identifier (maximum length 12 characters).

If *string* is nonblank, it is the user identifier which is to be used by the message channel agent for authorization to access MQSeries 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.

This user identifier can be overridden by one supplied by a channel security exit.

This parameter is not valid for channels with a channel type (CHLTYPE) of CLNTCONN.

**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.

On SunOS, the MODENAME should be set to the *unique\_session\_name* parameter as specified in the Sunlink configuration file.

On Tandem NSK, this should be set to the SNA mode name.

The name can also be nonblank for client connection channels to be used with OS/2 Warp.

On other platforms the mode name can only be set to blanks; the actual name is taken instead from the CPI-C Communications Side Object, or APPC side information data set.

This parameter is not valid for channels with a channel type (CHLTYPE) of RCVR or SVRCONN.

**MRDATA**(*string*)

Channel message-retry exit user data (maximum length 32 characters).

This 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 or RQSTR. It is not supported on MVS/ESA.

**MREXIT**(*string*)

Channel message-retry exit name.

The format and maximum length of the name is the same as for MSGEXIT.

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR or RQSTR. It is not supported on MVS/ESA.

## DEFINE CHANNEL

### MRRTY(*integer*)

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. If the exit name is not blank, the value of MRRTY is passed to the exit for the exit's use, but the number of retries performed (if any) is controlled by the exit, and not by this attribute.

The value must be greater than or equal to zero, and less than or equal to 999 999 999. A value of zero means that no retries will be performed.

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR or RQSTR. It is not supported on MVS/ESA.

### MRTMR(*integer*)

The minimum interval of time that must pass before the channel can retry the MQPUT operation. This time interval is in milliseconds.

This attribute 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 for the exit's use, but the retry interval is controlled by the exit, and not by this attribute.

The value must be greater than or equal to zero, and less than or equal to 999 999 999. A value of zero means that the retry will be performed as soon as possible (provided that the value of MRRTY is greater than zero).

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR or RQSTR. It is not supported on MVS/ESA.

### MSGDATA

User data for the channel message exit (maximum length 32 characters).

This data is passed to the channel message exit when it is called.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, you can specify data for more than one exit program by specifying multiple strings separated by commas. On OS/400, 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.

On other platforms you can specify only one string of message exit data for each channel.

### MSGEXIT

Channel message exit name.

On Tandem NSK, there is only one channel user exit program. 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 attributes. The maximum length of the string is 128 characters. This string is passed to the exit program, but it is not used to determine the program name. See the *MQSeries for Tandem NonStop Kernel System Management Guide* for more information about using channel exit programs on Tandem NSK.

On other platforms, 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, OS/2 Warp, Sun Solaris, and Windows NT, 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 OS/400, you can specify the names of up to 10 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 not relevant, because message exits are not invoked for such channels.

The format and maximum length of the name depends on the environment:

- On Digital OpenVMS and UNIX systems, it is of the form

libraryname(functionname)

The maximum length of the string is 128 characters.

- On OS/2 Warp, Windows, and Windows NT, 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 OS/400, it is of the form

progrname libname

where *progrname* 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 MVS/ESA, it is a load module name, maximum length 8 characters (128 characters are allowed for exit names for client-connection channels).

### NOREPLACE and REPLACE

Whether the existing definition is to be replaced with this one. This is optional. The default is NOREPLACE.

**NOREPLACE** The definition should not replace any existing definition of the same name.

**REPLACE** The definition should replace any existing definition of the same name. If a definition does not exist, one is created.

### 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. This is the default. 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 attribute, or one does not support it, NORMAL is used.

This parameter is valid only for channels with a CHLTYPE of SDR, SVR, RCVR, or RQSTR. It is valid only on AIX, HP-UX, MVS/ESA, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

## DEFINE CHANNEL

### **PASSWORD**(*string*)

Password (maximum length 12 characters).

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, or CLNTCONN. It is not supported on OS/400 and is only supported on MVS/ESA for client-connection channels.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

### **PUTAUT**

Specifies whether the user identifier in the context information associated with a message should be used to establish authority to put the message to the destination queue.

**DEF** Default user ID is used

**CTX** Context user ID is used

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR or RQSTR.

### **QMNAME**(*string*)

Queue manager name.

For channels with a channel type (CHLTYPE) of CLNTCONN, this is the name of the queue manager to which an application running in the MQI client environment can request connection.

For channels of other types this parameter is not valid.

### **RCVDATA**

Channel receive exit user data (maximum length 32 characters).

This is passed to the channel receive exit when it is called.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, you can specify data for more than one exit program by specifying multiple strings separated by commas. On OS/400, 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.

On other platforms you can specify only one string of receive exit data for each channel.

### **RCVEXIT**

Channel receive exit name.

On platforms other than Tandem NSK, 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, OS/2 Warp, Sun Solaris, and Windows NT, 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 OS/400, you can specify the names of up to 10 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**

Channel security exit user data (maximum length 32 characters).

This is passed to the channel security exit when it is called.

**SCYEXIT**

Channel security exit name.

On platforms other than Tandem NSK, 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 given the opportunity 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.

**SENDDATA**

Channel send exit user data (maximum length 32 characters).

This is passed to the channel send exit when it is called.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, you can specify data for more than one exit program by specifying multiple strings separated by commas. On OS/400, 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.

On other platforms you can specify only one string of send exit data for each channel.

**SENDEXIT**

Channel send exit name.

On platforms other than Tandem NSK, 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, OS/2 Warp, Sun Solaris, and Windows NT, 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 OS/400, you can specify the names of up to 10 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.

## DEFINE CHANNEL

### SEQWRAP(*integer*)

When this value is reached, sequence numbers wrap to start again at 1.

This value is non-negotiable and must match in both the local and remote channel definitions.

The value must be greater than or equal to 100, and less than or equal to 999 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR, SVR, RCVR, or RQSTR.

### SHORTRTY(*integer*)

The maximum number of attempts that are made by a sender or server 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 retry is unlikely to be successful, retries are not attempted.

The value must be greater than or equal to zero, and less than or equal to 999 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### SHORTTMR(*integer*)

For short retry attempts, this is the maximum number of seconds to wait before re-attempting 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 greater than or equal to zero, and less than or equal to 999 999 999.

**Note:** For implementation reasons, the maximum retry interval that can be used is 999 999; values exceeding this will be treated as 999 999.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR.

### 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.

On SunOS, the transaction program name should be set to the name to which the listener program (runmqsr) is listening at the remote end.

On Tandem NSK, this should be set to the local TP name. This can be followed by the name of the TP on the remote machine, for example:

```
TPNAME('localtp[.remotetp]')
```

Both names can be up to 16 characters in length.

The name can also be nonblank for client connection channels to be used with OS/2 Warp.

On other platforms, the transaction program name can only be set to blanks; the actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set. On Windows NT SNA Server, and in the side object on MVS/ESA, the TPNAME is wrapped to upper case.

This parameter is not valid for channels with a channel type (CHLTYPE) of RCVR.

**TRPTYPE**

Transport type to be used.

This is not required on AIX, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT. If you do not specify this parameter, 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. On MVS/ESA, if the SYSTEM.DEF.*channel-type* definition does not exist, the default is LU62.

This is required on all other platforms.

**DECNET**    DECnet Phase IV (supported only on Digital OpenVMS)

**LU62**        SNA LU 6.2

**NETBIOS**    NetBIOS (supported only on OS/2 Warp, Windows, Windows NT, and DOS; it also applies to MVS/ESA for client-connection channels)

**SPX**         Sequenced packet exchange (supported only on OS/2 Warp, Windows, Windows NT, and DOS)

**TCP**         Transmission Control Protocol/Internet Protocol (TCP/IP)

**USERID**(*string*)

Task user identifier (maximum length 12 characters).

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, or CLNTCONN. It is not supported on OS/400 and is only supported on MVS/ESA for client-connection channels.

Although the maximum length of the attribute is 12 characters, only the first 10 characters are used.

**XMITQ**(*string*)

Transmission queue name.

The name of the queue from which messages are retrieved. See “Rules for naming MQSeries objects” on page 5.

This parameter is valid only for channels with a channel type (CHLTYPE) of SDR or SVR. For these channel types this parameter is required.

## DEFINE MAXSMSGS

---

### DEFINE MAXSMSGS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DEFINE MAXSMSGS to define the maximum number of messages that a task can get or put within a single unit of recovery.

You can issue the DEFINE MAXSMSGS command at any time to change the number of messages allowed.

**Note:** This command is valid only on MVS/ESA. For other platforms use the MAXUMSGS parameter of the ALTER QMGR command instead.

**Synonym:** DEF MAXSM

▶—DEFINE MAXSMSGS(*integer*)—▶

### Keyword and parameter descriptions

(*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 999 999 999. The default value is 10 000.

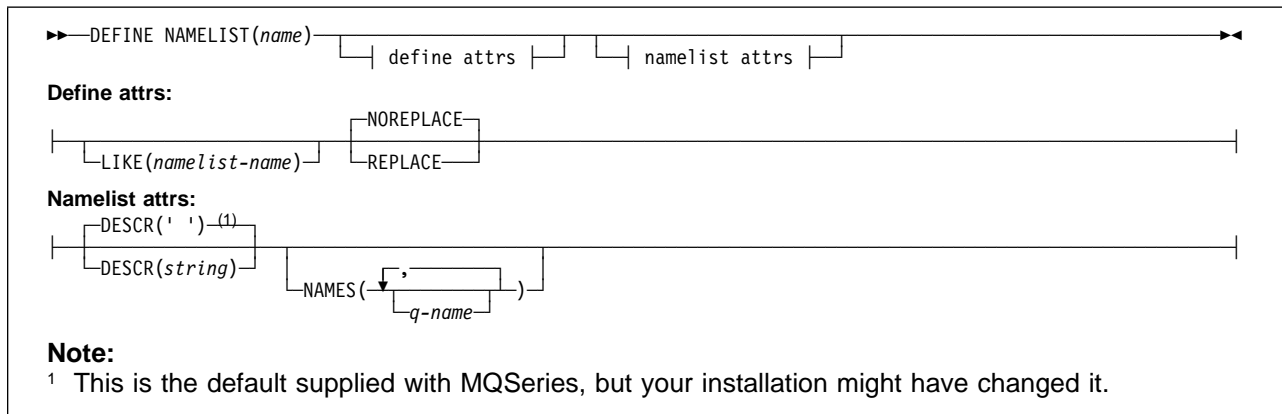
The number includes any trigger messages and report messages generated within the same unit of recovery.

## DEFINE NAMELIST

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DEFINE NAMELIST to define a list of queue names.

**Synonym:** DEF NL



## Keyword and parameter descriptions

(*name*) Name of the list. This is required.

The name must not be the same as any other namelist name currently defined on this queue manager (unless REPLACE is specified). See “Rules for naming MQSeries objects” on page 5.

### Define attributes

#### LIKE(*namelist-name*)

The name of a namelist, whose attributes will be used to model this definition.

If this field is not filled in, and you do not complete the attribute fields related to the command, the values are taken from the default definition for namelists on this queue manager.

This 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 MQSeries objects” on page 5.

#### NOREPLACE and REPLACE

Whether the existing definition is to be replaced with this one. This is optional. The default is NOREPLACE.

**NOREPLACE** The definition should not replace any existing definition of the same name.

**REPLACE** The definition should replace any existing definition of the same name. If a definition does not exist, one is created.

## DEFINE NAMELIST

### Namelist attributes

#### 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 136).

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.

#### NAMES(*q-name, ...*)

List of names of queues.

The queues can be of any type, and the list can contain queues of more than one type. In fact, because the objects defined in the namelist do not have to exist at the time the namelist is defined, there is no check that they are even the names of queues, although the characters used for each name are restricted to those that are valid for queue names (see “Rules for naming MQSeries objects” on page 5).

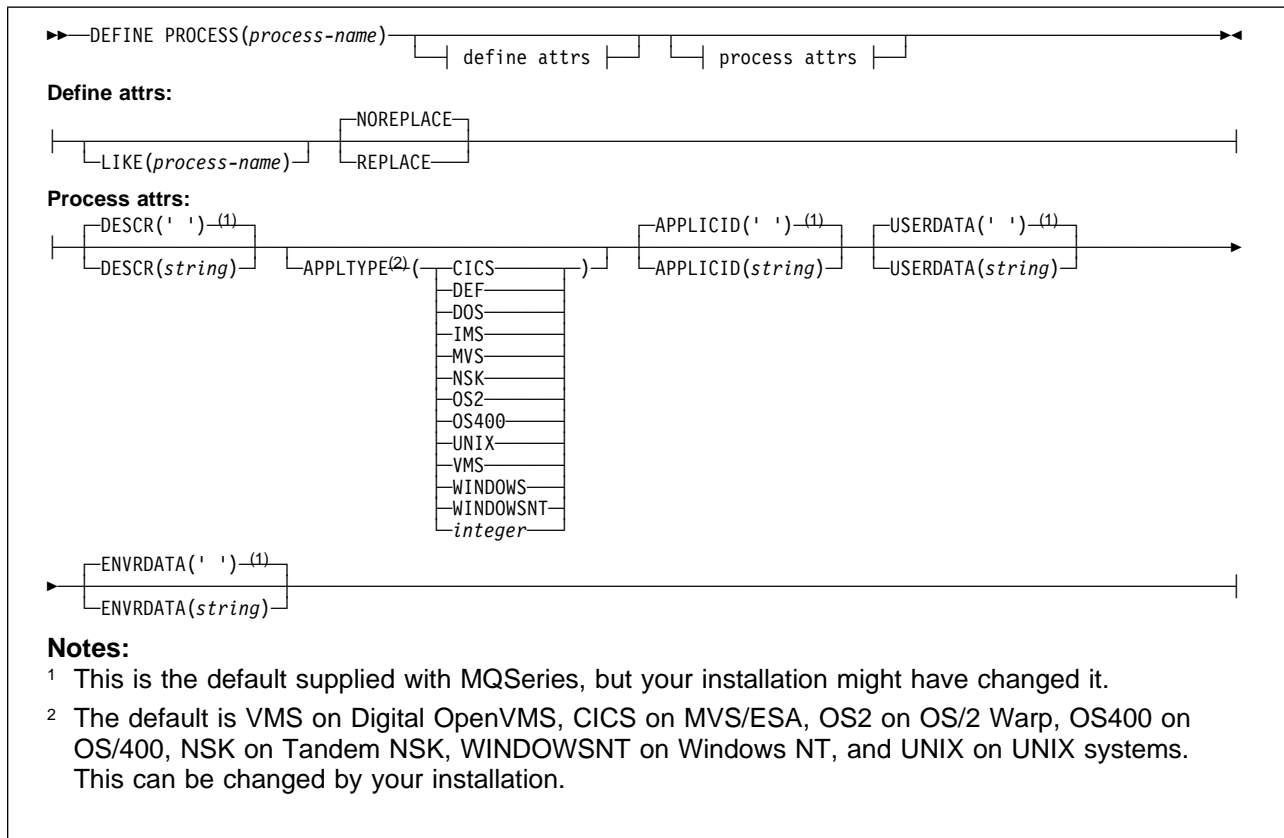
An empty list is valid: specify NAMES(). The maximum number of names in the list is 256.

## DEFINE PROCESS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DEFINE PROCESS to define a new MQSeries process definition, and set its attributes.

**Synonym:** DEF PRO



## Keyword and parameter descriptions

(*process-name*)

Name of the MQSeries process definition (see “Rules for naming MQSeries objects” on page 5). This is required.

The name must not be the same as any other process definition currently defined on this queue manager (unless REPLACE is specified).

## DEFINE PROCESS

### Define attributes

#### LIKE(*process-name*)

The name of an object of the same type, whose attributes will be used to model this definition.

If this field is not filled in, and you do not complete the attribute fields related to the command, the values are taken from the default definition for this object.

This is equivalent to specifying:

```
LIKE(SYSTEM.DEFAULT.PROCESS)
```

A default definition for each object type is provided, but these can be altered by the installation to the default values required. See “Rules for naming MQSeries objects” on page 5.

#### NOREPLACE and REPLACE

Whether the existing definition is to be replaced with this one. This is optional. The default is NOREPLACE.

**NOREPLACE** The definition should not replace any existing definition of the same name.

**REPLACE** The definition should replace any existing definition of the same name. If a definition does not exist, one is created.

### Process attributes

#### APPLICID(*string*)

The name of the application to be started. This might typically be a fully-qualified file name of an executable object. The maximum length is 256 characters.

For a CICS application this is a CICS transaction ID, and for an IMS application it is an IMS transaction ID.

On MVS/ESA, for distributed queuing using CICS it must be “CKSG”, and for distributed queuing without CICS, it must be “CSQX START”.

#### APPLTYPE(*string*)

The type of application to be started. Valid application types are:

<b>CICS</b>	Represents a CICS transaction.
<b>DOS</b>	Represents a DOS application.
<b>IMS</b>	Represents an IMS transaction.
<b>MVS</b>	Represents an MVS application (batch or TSO).
<b>NSK</b>	Represents a Tandem NSK application.
<b>OS2</b>	Represents an OS/2 Warp application.
<b>OS400</b>	Represents an OS/400 application.
<b>UNIX</b>	Represents a UNIX application.
<b>VMS</b>	Represents a Digital OpenVMS application.
<b>WINDOWS</b>	Represents a Windows application.
<b>WINDOWSNT</b>	Represents a Windows NT application.
<b>integer</b>	User-defined application type in the range 65 536 through 999 999 999.
<b>DEF</b>	This 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, this is interpreted as the default application type of the server.

Only application types (other than user-defined types) that are supported on the platform at which the command is executed should be used:

- On Digital OpenVMS, VMS is supported
- On MVS/ESA, CICS (default), IMS, MVS, and DEF are supported



- On OS/400, OS400 (default), CICS, and DEF are supported
- On OS/2 Warp, OS2 (default), DOS, WINDOWS, UNIX, CICS, and DEF are supported
- On Tandem NSK, NSK is supported.
- On UNIX systems, UNIX (default), OS2, DOS, WINDOWS, CICS, and DEF are supported
- On Windows NT, WINDOWSNT (default), DOS, WINDOWS, OS2, UNIX, CICS, and DEF are supported

### **DESCR**(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY PROCESS command.

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.

### **ENVRDATA**(*string*)

A character string that contains environment information pertaining to the application to be started. The maximum length is 128 characters.

### **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.

For MQSeries message channel agents, the format of this field is a channel name of up to 20 characters. See the *MQSeries Intercommunication* manual for information about what these need as APPLICID.

## DEFINE PSID

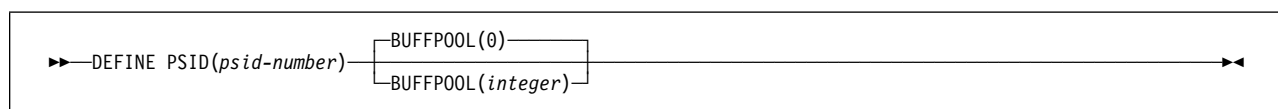
### DEFINE PSID

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DEFINE PSID to define a page set and associated buffer pool.

You can issue DEFINE PSID only from the CSQINP1 initialization data set. If more than one DEFINE PSID command is issued for the same page set, only the last one is actioned.

**Synonym:** DEF PSID



### Keyword and parameter descriptions

*(psid-number)*

Identifier of the page set. This is required.

In MQSeries for MVS/ESA 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 ESDS 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 0 through 3). This is optional. The default is 0.

See DEFINE BUFFPOOL on page 63.

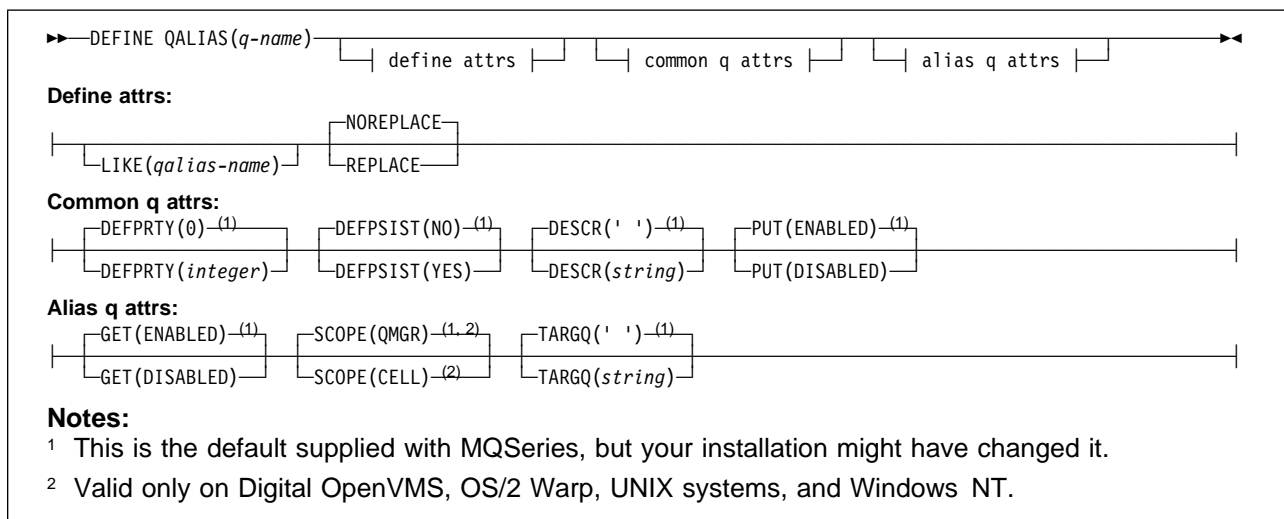
## DEFINE QALIAS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DEFINE QALIAS to define a new alias queue, and set its attributes.

An alias queue provides a level of indirection to another queue. The queue to which the alias refers must be another local or remote queue, defined at this queue manager. It cannot be another alias queue.

**Synonym:** DEF QA



## Keyword and parameter descriptions

(*q-name*)

Local name of the queue. This is required.

The name must not be the same as any other queue name (of whatever queue type) currently defined on this queue manager (unless REPLACE is specified). See “Rules for naming MQSeries objects” on page 5.

### Define attributes

LIKE(*qalias-name*)

The name of an object of the same type, whose attributes will be used to model this definition.

If this field is not filled in, and you do not complete the attribute fields related to the command, the values are taken from the default definition for this object.

This is equivalent to defining the following object:

LIKE(SYSTEM.DEFAULT.ALIAS.QUEUE)

A default definition for each object type is provided, but these can be altered by the installation to the default values required. See “Rules for naming MQSeries objects” on page 5.

## DEFINE QALIAS

### NOREPLACE and REPLACE

Whether the existing definition is to be replaced with this one. This is optional. The default is NOREPLACE.

**NOREPLACE** The definition should not replace any existing definition of the same name.

**REPLACE** If the object does exist, the effect is similar to issuing the ALTER command without the FORCE option and with *all* the other attributes 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 attributes, but DEFINE with REPLACE sets *all* the attributes. When you use REPLACE, the attributes are taken either from the object named on the LIKE attribute, or 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 the FORCE attribute if you were using the ALTER command
- The object is open

The ALTER command with the FORCE attribute succeeds in this situation.

If SCOPE(CELL) is specified on Digital OpenVMS, UNIX systems, OS/2, or Windows NT, and there is already a queue with the same name in the cell directory, the command fails, whether or not REPLACE is specified.

### Common queue attributes

#### DEFPRTY(*integer*)

The default priority of messages put on the queue. The value must be greater than or equal to zero, (the lowest priority) and less than or equal to the MAXPRTY queue manager attribute. (MAXPRTY is 9.)

#### DEFPSIST

The default message persistence for this queue:

**NO** Messages on this queue are lost across a restart of the queue manager. This is the default supplied with MQSeries, but your installation might have changed it.

**YES** Messages on this queue survive a restart of the queue manager.

#### DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY QUEUE command.

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.

**PUT** Whether messages can be put on the queue.

**ENABLED** Messages can be added to the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Messages cannot be added to the queue.

This attribute can also be changed using the **MQSET** API call.

## Alias queue attributes

**GET** Whether applications are permitted to get messages from this queue.

**ENABLED** Messages can be retrieved from the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Applications cannot retrieve messages from the queue.

This attribute can also be changed using the **MQSET** API call.

## SCOPE

Specifies the scope of the queue definition.

**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. To open the queue for output from some other queue manager, either the name of the owning queue manager must be specified, or the other queue manager must have a local definition of the queue.

**CELL** The queue definition has cell scope. This means that the queue is known to all of the queue managers in the cell, and 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.

This value is valid only if a name service supporting a cell directory (for example, the supplied DCE name service) has been configured.

This attribute is only supported on Digital OpenVMS, OS/2, Windows NT, and UNIX systems.

## TARGQ(*string*)

The local name of the base queue being aliased. (See “Rules for naming MQSeries objects” on page 5.) The maximum length is 48 characters.

This must be one of the following (although this is not checked until the alias queue is opened by an application):

- A local queue (not a dynamic queue)
- A local definition of a remote queue

This queue need not be defined until an application process attempts to open the alias queue.

## DEFINE QLOCAL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DEFINE QLOCAL to define a new local queue, and set its attributes.

**Synonym:** DEF QL

```

▶▶ DEFINE QLOCAL(q-name)
└─ define attrs ─┬─┬─ common q attrs ─┬─┬─ local q attrs ─┬─┬─▶▶

```

**Define attrs:**

```

└─ LIKE(qlocal-name) ─┬─ NOREPLACE ─┬─
└─ REPLACE ─┬─

```

**Common q attrs:**

```

└─ DEFPRTY(0) (1) ─┬─ DEFPSIST(NO) (1) ─┬─ DESCR(' ') (1) ─┬─ PUT(ENABLED) (1) ─┬─
└─ DEFPRTY(integer) ─┬─ DEFPSIST(YES) ─┬─ DESCR(string) ─┬─ PUT(DISABLED) ─┬─

```

**Local q attrs:**

```

└─ BOQNAME(' ') (1,2) ─┬─ BOTHRESH(0) (1,2) ─┬─ DEFSOPT(SHARED) (3) ─┬─ DISTL(NO) (1,4) ─┬─ GET(ENABLED) (1) ─┬─
└─ BOQNAME(string) (2) ─┬─ BOTHRESH(integer) (2) ─┬─ DEFSOPT(EXCL) ─┬─ DISTL(YES) (4) ─┬─ GET(DISABLED) ─┬─
▶─ INDXTYPE(NONE) (1,5) ─┬─ INITQ(' ') (1) ─┬─ MAXDEPTH(5000) (6) ─┬─ MAXMSGL(4 194 304) (1) ─┬─
▶─ INDXTYPE( ─MSGID ─CORRELID ) (5) ─┬─ INITQ(string) ─┬─ MAXDEPTH(integer) ─┬─ MAXMSGL(integer) ─┬─
▶─ MSGDLVQS(PRIORITY) (1) ─┬─ NOHARDENBO (1,2) ─┬─ SHARE (7) ─┬─ NOTRIGGER (1) ─┬─ PROCESS(' ') (1) ─┬─
▶─ MSGDLVQS(FIFO) ─┬─ HARDENBO (2) ─┬─ NOSHARE ─┬─ TRIGGER ─┬─ PROCESS(string) ─┬─
▶─ QDEPTHHI(80) (1) ─┬─ QDEPTHLO(40) (1) ─┬─ QDPHIEV(DISABLED) (1) ─┬─ QDPLOEV(DISABLED) (1) ─┬─
▶─ QDEPTHHI(integer) ─┬─ QDEPTHLO(integer) ─┬─ QDPHIEV(ENABLED) ─┬─ QDPLOEV(ENABLED) ─┬─
▶─ QDPMAEV(ENABLED) (1) ─┬─ QSVCIEV(NONE) (1) ─┬─ QSVCINT(999 999 999) (1) ─┬─ RETINTVL(999 999 999) (1) ─┬─
▶─ QDPMAEV(DISABLED) ─┬─ QSVCIEV( ─HIGH ─OK ) ─┬─ QSVCINT(integer) ─┬─ RETINTVL(integer) ─┬─
▶─ SCOPE(QMGR) (1,8) ─┬─ STGCLASS('DEFAULT') (1,5) ─┬─ TRIGDATA(' ') (1) ─┬─ TRIGDPH(1) (1) ─┬─ TRIGMPRI(0) (1) ─┬─
▶─ SCOPE(CELL) (8) ─┬─ STGCLASS(string) (5) ─┬─ TRIGDATA(string) ─┬─ TRIGDPH(integer) ─┬─ TRIGMPRI(integer) ─┬─
▶─ TRIGTYPE(FIRST) (1) ─┬─ USAGE(NORMAL) (1) ─┬─
▶─ TRIGTYPE( ─EVERY ─DEPTH ─NONE ) ─┬─ USAGE(XMITQ) ─┬─

```

**Notes:**

- 1 This is the default supplied with MQSeries, but your installation might have changed it.
- 2 Ignored on OS/400.
- 3 This is the default supplied with MQSeries (except on MVS/ESA, where it is EXCL), but your installation might have changed it.
- 4 Valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
- 5 Used only on MVS/ESA.
- 6 This is the default supplied with MQSeries (except on MVS/ESA, where it is 999 999 999), but your installation might have changed it.
- 7 This is the default supplied with MQSeries (except on MVS/ESA, where it is NOSHARE), but your installation might have changed it.
- 8 Valid only on Digital OpenVMS, OS/2 Warp, UNIX systems, and Windows NT.

## Keyword and parameter descriptions

A local queue is one that is owned by the queue manager to which it is being defined.

*(q-name)*

Local name of the queue. This is required.

The name must not be the same as any other queue name (of whatever queue type) currently defined on this queue manager (unless REPLACE is specified). See “Rules for naming MQSeries objects” on page 5.

### Define attributes

**LIKE***(qlocal-name)*

The name of an object of the same type, whose attributes will be used to model this definition.

If this field is not filled in, and you do not complete the attribute fields related to the command, the values are taken from the default definition for this object.

This is equivalent to defining the following object:

```
LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE)
```

A default definition for each object type is provided, but these might be altered by the installation to the default values required. See “Rules for naming MQSeries objects” on page 5.

### **NOREPLACE** and **REPLACE**

Whether the existing definition is to be replaced with this one. This is optional. The default is NOREPLACE.

**NOREPLACE** The definition should not replace any existing definition of the same name.

**REPLACE** If the object does exist, the effect is similar to issuing the ALTER command without the FORCE option and with *all* the other attributes specified. In particular, note that any messages that are on the existing queue are retained.

(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 attributes, but DEFINE with REPLACE sets *all* the attributes. When you use REPLACE, unspecified attributes are taken either from the object named on the LIKE attribute, or 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 the FORCE attribute if you were using the ALTER command
- The object is open

The ALTER command with the FORCE attribute succeeds in this situation.

If SCOPE(CELL) is specified on Digital OpenVMS, UNIX systems, OS/2 Warp, or Windows NT, and there is already a queue with the same name in the cell directory, the command fails, whether or not REPLACE is specified.

### Common queue attributes

**DEFPRTY***(integer)*

The default priority of messages put on the queue. The value must be greater than or equal to zero, (the lowest priority) and less than or equal to the MAXPRTY queue manager attribute. MAXPRTY can be displayed using the DISPLAY QMGR command. (MAXPRTY is 9.)

## DEFINE QLOCAL

### DEFPSIST

The default message persistence for this queue:

**NO** Messages on this queue are lost across a restart of the queue manager. This is the default supplied with MQSeries, but your installation might have changed it.

**YES** Messages on this queue survive a restart of the queue manager.

### DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY QUEUE command.

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.

**PUT** Whether messages can be put on the queue.

**ENABLED** Messages can be added to the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Messages cannot be added to the queue.

This attribute can also be changed using the **MQSET** API call.

## Local queue attributes

### BOQNAME(*string*)

The excessive backout requeue name. Apart from maintaining a value for this attribute, the queue manager takes no action based on its value. On OS/400, the keyword is ignored.

### BOTHRESH(*integer*)

The backout threshold. Apart from maintaining a value for this attribute, the queue manager takes no action based on its value. On OS/400, the keyword is ignored.

Specify a value greater than or equal to zero, and less than or equal to 999 999 999.

### 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

**DISTL** 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 should not normally change this attribute, because it is set by the MCA. However you can set this attribute when defining a transmission queue if the distribution list capability of the destination queue manager is known.

This keyword is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

**GET** Whether applications are to be permitted to get messages from this queue:

**ENABLED** Suitably authorized applications can retrieve messages from the queue. This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Applications cannot retrieve messages from the queue.

This attribute can also be changed using the **MQSET** API call.



**INDXTYPE**

The type of index maintained by the queue manager to expedite **MQGET** operations on the queue:

- NONE** No index is maintained. Use this when messages are usually 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 this when messages are usually 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 this when messages are usually retrieved using the correlation identifier as a selection criterion on the **MQGET** call with the message identifier set to NULL.

The **INDXTYPE** attribute can be changed at any time, and the change takes effect immediately if all the following conditions are satisfied:

- No applications have the queue open
- The queue is empty
- There are no uncommitted MQPUT or MQGET operations outstanding against the queue

If these conditions are not satisfied, the attribute is changed immediately, but the index is not rebuilt until the next time the queue manager is restarted. The reply sent by the queue manager indicates if this is the case.

This attribute is supported only on MVS/ESA. On other platforms, retrieval optimization might be provided, but it is not controlled by a queue attribute.

**INITQ(string)**

The local name of a local queue (known as the *initiation queue*) on this queue manager, to which trigger messages relating to this queue are written. See “Rules for naming MQSeries objects” on page 5.

**MAXDEPTH(integer)**

The maximum number of messages allowed on the queue. Specify a value greater than or equal to zero, and less than or equal to:

- 999 999 999 if the queue is on MVS/ESA
- 640 000 if the queue is on any other MQSeries platform

Other factors can still cause the queue to be treated as full, for example, if there is no further DASD space available.

If this value is reduced, any messages that are already on the queue that cause the new maximum to be exceeded remain intact.

**MAXMSGL(integer)**

The maximum length of messages on this queue. On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the **MAXMSGL** parameter of the **ALTER QMGR** command for more information. On other platforms, specify a value greater than or equal to zero, and less than or equal to 4 194 304.

For a transmission queue, this value includes the space required for headers. It is recommended that the value should be at least 4000 bytes larger than the maximum expected length of user data in any message that could be put on a transmission queue.

If this value is reduced, any messages that are already on the queue, whose length exceeds the new maximum, are not affected.

## DEFINE QLOCAL

Applications can use this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore, the value should only be reduced if it is known that this will not cause an application to operate incorrectly.

### MSGDLVSQ

Message delivery sequence:

**PRIORITY** Messages are delivered (in response to **MQGET** API calls) in first-in-first-out (FIFO) order within priority. This is the default supplied with MQSeries, but your installation might have changed it.

**FIFO** Messages are delivered (in response to **MQGET** API calls) in FIFO order. Priority is ignored for messages on this queue.

If the message delivery sequence is changed from PRIORITY to FIFO while there are messages on the queue, the order of the messages already enqueued is not changed. Messages added to the queue subsequently 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 enqueued while the queue was set to FIFO take the default priority.

### NOHARDENBO and HARDENBO

Whether hardening should be used to ensure that the count of the number of times that a message has been backed out is accurate.

**Note:** The count is always hardened on OS/400; the HARDENBO and NOHARDENBO keywords are ignored if specified.

**NOHARDENBO** The count is not hardened. This is the default supplied with MQSeries, but your installation might have changed it.

**HARDENBO** The count is hardened.

### NOSHARE and SHARE

Whether multiple applications can get messages from this queue:

**NOSHARE** A single application instance only can get messages from the queue

**SHARE** More than one application instance can get messages from the queue

### NOTRIGGER and TRIGGER

Whether trigger messages are written to the initiation queue (named by the INITQ attribute) to trigger the application (named by the PROCESS attribute):

**NOTRIGGER** Triggering is not active, and trigger messages are not written to the initiation queue. This is the default supplied with MQSeries, but your installation might have changed it.

**TRIGGER** Triggering is active, and trigger messages are written to the initiation queue.

This attribute can also be changed using the **MQSET** API call.

### PROCESS(*string*)

The local name of the MQSeries process. This 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 MQSeries objects" on page 5.

The process does not have to be defined 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 gives the name of the channel to be started. This parameter is optional for transmission queues on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT; if you do not specify it, the channel name is taken from the value specified for the TRIGDATA parameter.

**QDEPTHHI**(*integer*)

The threshold against which the queue depth is compared to generate a Queue Depth High event.

This event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the QDPHIEV attribute.

The value is expressed as a percentage of the maximum queue depth (MAXDEPTH attribute), and must be greater than or equal to zero, and less than or equal to 100.

**QDEPTHLO**(*integer*)

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

This event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the QDPLOEV attribute.

The value is expressed as a percentage of the maximum queue depth (MAXDEPTH attribute), and must be greater than or equal to zero, and less than or equal to 100.

**QDPHIEV**

Controls whether Queue Depth High events are generated.

A Queue Depth High event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold (see the QDEPTHHI attribute).

**Note:** The value of this attribute can change implicitly. See the description of the Queue Depth High event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Depth High events are generated

**DISABLED** Queue Depth High events are not generated

**QDPLOEV**

Controls whether Queue Depth Low events are generated.

A Queue Depth Low event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold (see the QDEPTHLO attribute).

**Note:** The value of this attribute can change implicitly. See the description of the Queue Depth Low event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Depth Low events are generated

**DISABLED** Queue Depth Low events are not generated

**QDPMAXEV**

Controls whether Queue Full events are generated.

A Queue Full event indicates that a put to a queue has been rejected because the queue is full, that is, the queue depth has already reached its maximum value.

**Note:** The value of this attribute can change implicitly. See the description of the Queue Full event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Full events are generated

**DISABLED** Queue Full events are not generated

## DEFINE QLOCAL

### QSVCIEV

Controls 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 at least the time indicated by the QSVCINT attribute.

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 QSVCINT attribute.

**Note:** The value of this attribute can change implicitly. See the description of the Service Interval High and Service Interval OK events in the *MQSeries Programmable System Management* manual for more details.

**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. See the QSVCIEV attribute.

The value is in units of milliseconds, and must be greater than or equal to zero, and less than or equal to 999 999 999.

### RETINTVL(*integer*)

The number of hours (greater than or equal to zero, and less than or equal to 999 999 999) from the queue creation date and time (the date and time at which the queue was defined), after which the queue is no longer needed. The CRDATE and CRTIME can be displayed using DISPLAY QUEUE on page 141.

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 has not expired. It is the user's responsibility to take any required action.

### SCOPE

Specifies the scope of the queue definition.

**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. To open the queue for output from some other queue manager, either the name of the owning queue manager must be specified, or the other queue manager must have a local definition of the queue.

**CELL** The queue definition has cell scope. This means that the queue is known to all of the queue managers in the cell, and 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 keyword has no effect on this.

This value is valid only if a name service supporting a cell directory (for example, the supplied DCE name service) has been configured.

This attribute is valid only on Digital OpenVMS, OS/2 Warp, UNIX systems, and Windows NT.

**STGCLASS**(*string*)

The name of the storage class. This is an installation-defined name.

This attribute is used only on MVS/ESA. See the *MQSeries for MVS/ESA System Management Guide* for more details. The first character of the name must be uppercase A–Z, and subsequent characters either uppercase A–Z or numeric 0–9.

**Note:** This attribute can be changed only if the queue is empty and closed.

On platforms other than MVS/ESA, this attribute is ignored.

**TRIGDATA**(*string*)

The data that is inserted in the trigger message. The maximum length of the string is 64 bytes.

For a transmission queue on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, or Windows NT, you can use this parameter to specify the name of the channel to be started.

This attribute 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 greater than zero, and less than or equal to 999 999 999.

This attribute can also be changed using the **MQSET** API call.

**TRIGMPRI**(*integer*)

The message priority number that will trigger this queue. The value must be greater than or equal to zero, and less than or equal to the MAXPRTY queue manager attribute (see DISPLAY QMGR on page 138 for details).

This attribute can also be changed using the **MQSET** API call.

**TRIGTYPE**

Whether and under what conditions a trigger message is written to the initiation queue (named by the INITQ attribute):

**FIRST** Whenever the first message of priority equal to or greater than that specified by the TRIGMPRI attribute of the queue arrives on the queue.

**EVERY** Every time a message arrives on the queue with priority equal to or greater than that specified by the TRIGMPRI attribute of the queue.

**DEPTH** When the number of messages with priority equal to or greater than that specified by TRIGMPRI is equal to the number indicated by the TRIGDPTH attribute.

**NONE** No trigger messages are written.

This attribute can also be changed using the **MQSET** API call.

**USAGE**

Queue usage:

**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 until it has been successfully transmitted and stored at the remote queue manager.

## DEFINE QMODEL

### DEFINE QMODEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
✓	✓	✓	✓	✓	✓	✓

Use DEFINE QMODEL to define a new model queue, and set its attributes.

**Synonym:** DEF QM

```

▶▶ DEFINE QMODEL(q-name)
└─ define attrs ─┬─ common q attrs ─┬─ local q attrs ─┬─ model q attr ─▶▶

```

**Define attrs:**

```

└─ LIKE(qmodel-name) ─┬─ NOREPLACE ─┬─ REPLACE ─┬─

```

**Common q attrs:**

```

└─ DEFPRTY(0)-(1) ─┬─ DEFPSIST(NO)-(1) ─┬─ DESCR(' ')-(1) ─┬─ PUT(ENABLED)-(1) ─┬─
└─ DEFPRTY(integer) ─┬─ DEFPSIST(YES) ─┬─ DESCR(string) ─┬─ PUT(DISABLED) ─┬─

```

**Local q attrs:**

```

└─ BOQNAME(' ')-(1,2) ─┬─ BOTHRESH(0)-(1,2) ─┬─ DEFISOPT(EXCL)-(1) ─┬─ DISTL(NO)-(1,3) ─┬─ GET(ENABLED)-(1) ─┬─
└─ BOQNAME(string)(2) ─┬─ BOTHRESH(integer)(2) ─┬─ DEFISOPT(SHARED) ─┬─ DISTL(YES)-(3) ─┬─ GET(DISABLED) ─┬─
└─ INDXTYPE(NONE)-(1,4) ─┬─ INITQ(' ')-(1) ─┬─ MAXDEPTH(5000)-(5) ─┬─ MAXMSGL(4 194 304)-(1) ─┬─
└─ INDXTYPE(└─ MSGID ─┬─ CORRELID ─┬─) ─┬─ INITQ(string) ─┬─ MAXDEPTH(integer) ─┬─ MAXMSGL(integer) ─┬─

```

```

└─ MSGDLVSQ(PRIORITY)-(1) ─┬─ NOHARDENBO(1,2) ─┬─ NOSHARE-(1) ─┬─ NOTRIGGER(1) ─┬─ PROCESS(' ')-(1) ─┬─
└─ MSGDLVSQ(FIFO) ─┬─ HARDENBO(2) ─┬─ SHARE ─┬─ TRIGGER ─┬─ PROCESS(string) ─┬─

```

```

└─ QDEPTHHI(80)-(1) ─┬─ QDEPTHLO(40)-(1) ─┬─ QDPHIEV(DISABLED)-(1) ─┬─ QDPLOEV(DISABLED)-(1) ─┬─
└─ QDEPTHHI(integer) ─┬─ QDEPTHLO(integer) ─┬─ QDPHIEV(ENABLED) ─┬─ QDPLOEV(ENABLED) ─┬─

```

```

└─ QDPMAEV(ENABLED)-(1) ─┬─ QSVCIEV(NONE)-(1) ─┬─ QSVCINT(999 999 999)-(1) ─┬─ RETINTVL(999 999 999)-(1) ─┬─
└─ QDPMAEV(DISABLED) ─┬─ QSVCIEV(└─ HIGH ─┬─ OK ─┬─) ─┬─ QSVCINT(integer) ─┬─ RETINTVL(integer) ─┬─

```

```

└─ STGCLASS('DEFAULT')-(1,4) ─┬─ TRIGDATA(' ')-(1) ─┬─ TRIGDPH(1)-(1) ─┬─ TRIGMPRI(0)-(1) ─┬─
└─ STGCLASS(string)(4) ─┬─ TRIGDATA(string) ─┬─ TRIGDPH(integer) ─┬─ TRIGMPRI(integer) ─┬─

```

```

└─ TRIGTYPE(FIRST)-(1) ─┬─ USAGE(NORMAL)-(1) ─┬─
└─ TRIGTYPE(└─ EVERY ─┬─ DEPTH ─┬─ NONE ─┬─) ─┬─ USAGE(XMITQ) ─┬─

```

**Model q attr:**

```

└─ DEFTYPE(TEMPDYN)-(1) ─┬─
└─ DEFTYPE(PERMDYN) ─┬─

```

**Notes:**

- 1 This is the default supplied with MQSeries, but your installation might have changed it.
- 2 Ignored on OS/400.
- 3 Valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
- 4 Used only on MVS/ESA.
- 5 This is the default supplied with MQSeries (except on MVS/ESA, where it is 999 999 999), but your installation might have changed it.

## Keyword and parameter descriptions

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.

*(q-name)*

Local name of the queue. This is required.

The name must not be the same as any other queue name (of whatever queue type) currently defined on this queue manager (unless REPLACE is specified). See “Rules for naming MQSeries objects” on page 5.

### Define attributes

**LIKE***(qmodel-name)*

The name of an object of the same type, whose attributes will be used to model this definition.

If this field is not filled in, and you do not complete the attribute fields related to the command, the values are taken from the default definition for this object.

This is equivalent to specifying:

LIKE(SYSTEM.DEFAULT.MODEL.QUEUE)

A default definition for each object type is provided, but these can be altered by the installation to the default values required. See “Rules for naming MQSeries objects” on page 5.

### **NOREPLACE** and **REPLACE**

Whether the existing definition is to be replaced with this one. This is optional. The default is **NOREPLACE**.

**NOREPLACE** The definition should not replace any existing definition of the same name.

**REPLACE** The definition should replace any existing definition of the same name. If a definition does not exist, one is created.

### Common queue attributes

**DEFPRTY***(integer)*

The default priority of messages put on the queue. The value must be greater than or equal to zero, (the lowest priority) and less than or equal to the **MAXPRTY** queue manager attribute. **MAXPRTY** can be displayed using the **DISPLAY QMGR** command. (**MAXPRTY** is 9.)

**DEFPSIST**

The default message persistence for this queue:

**NO** Messages on this queue are lost across a restart of the queue manager. This is the default supplied with MQSeries, but your installation might have changed it.

**YES** Messages on this queue survive a restart of the queue manager.

Persistent messages are not allowed on a temporary dynamic queue. In order to avoid an error if a message is put to such a queue with default persistence, do not set the **DEFPSIST** attribute to **YES** for model queue definitions that have a **DEFTYPE** of **TEMPDYN**.

## DEFINE QMODEL

### DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY QUEUE command.

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.

**PUT** Whether messages can be put on the queue.

**ENABLED** Messages can be added to the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Messages cannot be added to the queue.

This attribute can also be changed using the **MQSET** API call.

## Local queue attributes

### BOQNAME(*string*)

The excessive backout requeue name. Apart from maintaining a value for this attribute, the queue manager takes no action based on its value. On OS/400, the keyword is ignored.

### BOTHRESH(*integer*)

The backout threshold. Apart from maintaining a value for this attribute, the queue manager takes no action based on its value. On OS/400, the keyword is ignored.

Specify a value greater than or equal to zero, and less than or equal to 999 999 999.

### 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

**DISTL** 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 should not normally change this attribute, because it is set by the MCA. However you can set this attribute when defining a transmission queue if the distribution list capability of the destination queue manager is known.

This keyword is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.

**GET** Whether applications are to be permitted to get messages from this queue:

**ENABLED** Suitably authorized applications can retrieve messages from the queue. This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Applications cannot retrieve messages from the queue.

This attribute can also be changed using the **MQSET** API call.

### INDXTYPE

The type of index maintained by the queue manager to expedite **MQGET** operations on the queue:

**NONE** No index is maintained. Use this when messages are usually 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 this when messages are usually 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 this when messages are usually retrieved using the correlation identifier as a selection criterion on the **MQGET** call with the message identifier set to NULL.

The INDXTYPE attribute can be changed at any time, and the change takes effect immediately if all the following conditions are satisfied:

- No applications have the queue open
- The queue is empty
- There are no uncommitted MQPUT or MQGET operations outstanding against the queue

If these conditions are not satisfied, the attribute is changed immediately, but the index is not rebuilt until the next time the queue manager is restarted. The reply sent by the queue manager indicates if this is the case.

This attribute is supported only on MVS/ESA. On other platforms, retrieval optimization might be provided, but it is not controlled by a queue attribute.

#### **INITQ**(string)

The local name of a local queue (known as the *initiation queue*) on this queue manager, to which trigger messages relating to this queue are written. See “Rules for naming MQSeries objects” on page 5.

#### **MAXDEPTH**(integer)

The maximum number of messages allowed on the queue. Specify a value greater than or equal to zero, and less than or equal to:

- 999 999 999 if the queue is on MVS/ESA
- 640 000 if the queue is on any other MQSeries platform

Other factors might still cause the queue to be treated as full, for example, if there is no further DASD space available.

If this value is reduced, any messages that are already on the queue that cause the new maximum to be exceeded remain intact.

#### **MAXMSGL**(integer)

The maximum length of messages on this queue. On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the MAXMSGL parameter of the ALTER QMGR command for more information. On other platforms, specify a value greater than or equal to zero, and less than or equal to 4 194 304.

For a transmission queue, this value includes the space required for headers. It is recommended that the value should be at least 4000 bytes larger than the maximum expected length of user data in any message that could be put on a transmission queue.

If this value is reduced, any messages that are already on the queue, whose length exceeds the new maximum, are not affected.

Applications can use this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore, the value should only be reduced if it is known that this will not cause an application to operate incorrectly.

## DEFINE QMODEL

### MSGDLVSQ

Message delivery sequence:

**PRIORITY** Messages are delivered (in response to **MQGET** API calls) in first-in-first-out (FIFO) order within priority. This is the default supplied with MQSeries, but your installation might have changed it.

**FIFO** Messages are delivered (in response to **MQGET** API calls) in FIFO order. Priority is ignored for messages on this queue.

If the message delivery sequence is changed from PRIORITY to FIFO while there are messages on the queue, the order of the messages already enqueued is not changed. Messages added to the queue subsequently 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 enqueued while the queue was set to FIFO take the default priority.

### NOHARDENBO and HARDENBO

Whether hardening should be used to ensure that the count of the number of times that a message has been backed out is accurate.

**Note:** The count is always hardened on OS/400; the HARDENBO and NOHARDENBO keywords are ignored if specified.

**NOHARDENBO** The count is not hardened. This is the default supplied with MQSeries, but your installation might have changed it.

**HARDENBO** The count is hardened.

### NOSHARE and SHARE

Whether multiple applications can get messages from this queue:

**NOSHARE** A single application instance only can get messages from the queue

**SHARE** More than one application instance can get messages from the queue

### NOTRIGGER and TRIGGER

Whether trigger messages are written to the initiation queue (named by the INITQ attribute) to trigger the application (named by the PROCESS attribute):

**NOTRIGGER** Triggering is not active, and trigger messages are not written to the initiation queue. This is the default supplied with MQSeries, but your installation might have changed it.

**TRIGGER** Triggering is active, and trigger messages are written to the initiation queue.

This attribute can also be changed using the **MQSET** API call.

### PROCESS(*string*)

The local name of the MQSeries process. This 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 MQSeries objects" on page 5.

The process does not have to be defined 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 gives the name of the channel to be started. This parameter is optional for transmission queues on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT; if you do not specify it, the channel name is taken from the value specified for the TRIGDATA parameter.

**QDEPTHHI**(*integer*)

The threshold against which the queue depth is compared to generate a Queue Depth High event.

This event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the QDPHIEV attribute.

The value is expressed as a percentage of the maximum queue depth (MAXDEPTH attribute), and must be greater than or equal to zero, and less than or equal to 100.

**QDEPTHLO**(*integer*)

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

This event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the QDPLOEV attribute.

The value is expressed as a percentage of the maximum queue depth (MAXDEPTH attribute), and must be greater than or equal to zero, and less than or equal to 100.

**QDPHIEV**

Controls whether Queue Depth High events are generated.

A Queue Depth High event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold (see the QDEPTHHI attribute).

**Note:** The value of this attribute can change implicitly. See the description of the Queue Depth High event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Depth High events are generated

**DISABLED** Queue Depth High events are not generated

**QDPLOEV**

Controls whether Queue Depth Low events are generated.

A Queue Depth Low event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold (see the QDEPTHLO attribute).

**Note:** The value of this attribute can change implicitly. See the description of the Queue Depth Low event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Depth Low events are generated

**DISABLED** Queue Depth Low events are not generated

**QDPMAXEV**

Controls whether Queue Full events are generated.

A Queue Full event indicates that a put to a queue has been rejected because the queue is full, that is, the queue depth has already reached its maximum value.

**Note:** The value of this attribute can change implicitly. See the description of the Queue Full event in the *MQSeries Programmable System Management* manual for more details.

**ENABLED** Queue Full events are generated

**DISABLED** Queue Full events are not generated

## DEFINE QMODEL

### QSVCIEV

Controls 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 at least the time indicated by the QSVCINT attribute.

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 QSVCINT attribute.

**Note:** The value of this attribute can change implicitly. See the description of the Service Interval High and Service Interval OK events in the *MQSeries Programmable System Management* manual for more details.

**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. See the QSVCIEV attribute.

The value is in units of milliseconds, and must be greater than or equal to zero, and less than or equal to 999 999 999.

### RETINTVL(*integer*)

The number of hours (greater than or equal to zero, and less than or equal to 999 999 999) from the queue creation date and time (the date and time at which the queue was defined), after which the queue is no longer needed. The CRDATE and CRTIME can be displayed using DISPLAY QUEUE on page 141.

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 has not expired. It is the user's responsibility to take any required action.

### STGCLASS(*string*)

The name of the storage class. This is an installation-defined name.

This attribute is used only on MVS/ESA. See the *MQSeries for MVS/ESA System Management Guide* for more details. The first character of the name must be uppercase A–Z, and subsequent characters either uppercase A–Z or numeric 0–9.

**Note:** This attribute can be changed only if the queue is empty and closed.

On platforms other than MVS/ESA, this attribute is ignored.

### TRIGDATA(*string*)

The data that is inserted in the trigger message. The maximum length of the string is 64 bytes.

For a transmission queue on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, or Windows NT, you can use this parameter to specify the name of the channel to be started.

This attribute can also be changed using the **MQSET** API call.

### TRIGDPH(*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 greater than zero, and less than or equal to 999 999 999.

This attribute can also be changed using the **MQSET** API call.

**TRIGMPRI**(*integer*)

The message priority number that will trigger this queue. The value must be greater than or equal to zero, and less than or equal to the MAXPRTY queue manager attribute (see DISPLAY QMGR on page 138 for details).

This attribute can also be changed using the **MQSET** API call.

**TRIGTYPE**

Whether and under what conditions a trigger message is written to the initiation queue (named by the INITQ attribute):

**FIRST** Whenever the first message of priority equal to or greater than that specified by the TRIGMPRI attribute of the queue arrives on the queue.

**EVERY** Every time a message arrives on the queue with priority equal to or greater than that specified by the TRIGMPRI attribute of the queue.

**DEPTH** When the number of messages with priority equal to or greater than that specified by TRIGMPRI is equal to the number indicated by the TRIGDPTH attribute.

**NONE** No trigger messages are written.

This attribute can also be changed using the **MQSET** API call.

**USAGE**

Queue usage:

**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 until it has been successfully transmitted and stored at the remote queue manager.

**Model queue attributes**

**DEFTYPE**

Queue definition type:

**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).

Do not specify this value for a model queue definition with a DEFPSIST attribute of YES.

**PERMDYN** 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).

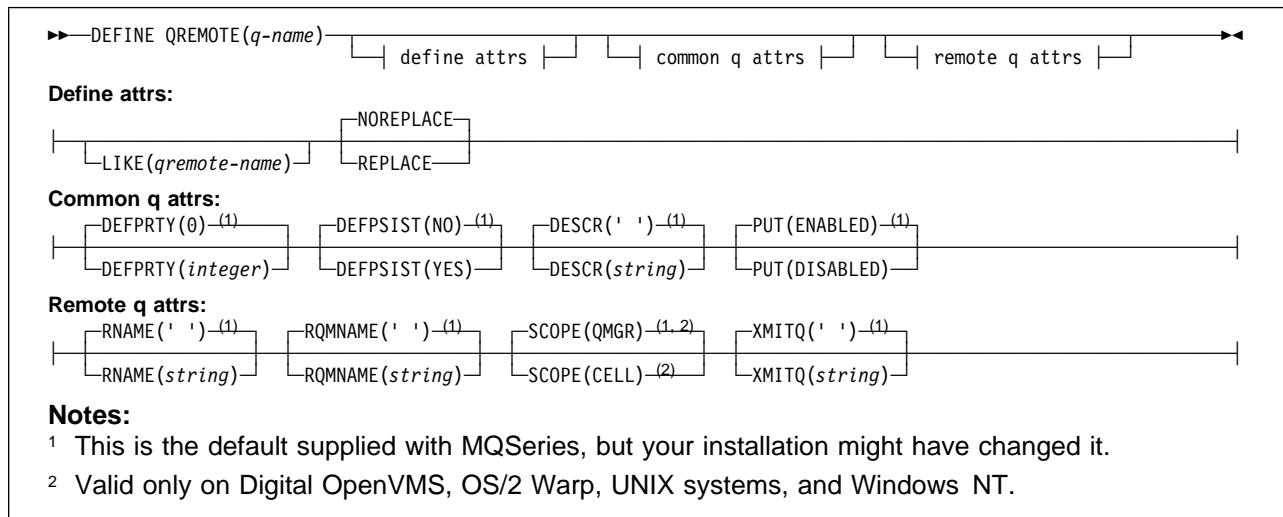
## DEFINE QREMOTE

### DEFINE QREMOTE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

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 attributes.

**Synonym:** DEF QR



## Keyword and parameter descriptions

A remote queue is one that is owned by another queue manager that application processes connected to this queue manager need to access.

Remote queues do not have to be defined locally in this way. The advantage of doing so is that applications can refer to the queue by a simple, locally defined name, rather than by one that is qualified by the ID of the queue manager on which the queue resides. This means that applications do not need to be aware of the real location of the queue.

A remote queue definition can also be used as a mechanism for holding a queue-manager alias definition, or a reply-to queue alias definition. The name of the definition in these cases is:

- The queue-manager name being used as the alias for another queue-manager name (queue-manager alias), or
- The queue name being used as the alias for the reply-to queue (reply-to queue alias).

(*q-name*)

Name of the local definition of the remote queue. This is required.

The name must not be the same as any other queue name (of whatever queue type) currently defined on this queue manager (unless REPLACE is specified). See “Rules for naming MQSeries objects” on page 5.

## Define attributes

### LIKE(*qremote-name*)

The name of an object of the same type, whose attributes will be used to model this definition.

If this field is not filled in, and you do not complete the attribute fields related to the command, the values are taken from the default definition for this object.

This is equivalent to defining the following object:

```
LIKE(SYSTEM.DEFAULT.REMOTE.QUEUE)
```

A default definition for each object type is provided, but these might be altered by the installation to the default values required. See “Rules for naming MQSeries objects” on page 5.

### NOREPLACE and REPLACE

Whether the existing definition is to be replaced with this one. This is optional. The default is NOREPLACE.

**NOREPLACE** The definition should not replace any existing definition of the same name.

**REPLACE** If the object does not exist already, one is created.

If the object does exist, the effect is similar to issuing the ALTER command without the FORCE option and with *all* the other attributes 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 attributes, but DEFINE with REPLACE sets *all* the attributes. When you use REPLACE, the attributes are taken either from the object named on the LIKE attribute, or 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 the FORCE attribute if you were using the ALTER command
- The object is open

The ALTER command with the FORCE attribute succeeds in this situation.

If SCOPE(CELL) is specified on Digital OpenVMS, OS/2 Warp, UNIX systems, or Windows NT, and there is already a queue with the same name in the cell directory, the command fails, whether or not REPLACE is specified.

## Common queue attributes

### DEFPRTY(*integer*)

The default priority of messages put on the queue. The value must be greater than or equal to zero, (the lowest priority) and less than or equal to the MAXPRTY queue manager attribute. MAXPRTY can be displayed using the DISPLAY QMGR command. (MAXPRTY is 9.)

### DEFPSIST

The default message persistence for this queue:

**NO** Messages on this queue are lost across a restart of the queue manager. This is the default supplied with MQSeries, but your installation might have changed it.

**YES** Messages on this queue survive a restart of the queue manager.

## DEFINE QREMOTE

### DESCR(*string*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY QUEUE command.

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.

**PUT** Whether messages can be put on the queue.

**ENABLED** Messages can be added to the queue (by suitably authorized applications). This is the default supplied with MQSeries, but your installation might have changed it.

**DISABLED** Messages cannot be added to the queue.

This attribute can also be changed using the **MQSET** API call.

## Remote queue attributes

### RNAME(*string*)

Name of remote queue. This is the local name of the queue as defined on the queue manager specified by RQMNAME. The name is *not* checked to ensure that it contains only those characters normally allowed for queue names (see “Rules for naming MQSeries objects” on page 5).

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.

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.

### RQMNAME(*string*)

The name of the remote queue manager on which the queue RNAME is defined.

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, then 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 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 name is *not* checked to ensure that it contains only those characters normally allowed for queue names (see “Rules for naming MQSeries objects” on page 5).



**SCOPE**

Specifies the scope of the queue definition.

**QMGR** The queue definition has queue-manager scope. This means that the definition of the queue does not extend beyond the queue manager which owns it. To open the queue for output from some other queue manager, either the name of the owning queue manager must be specified, or the other queue manager must have a local definition of the queue.

**CELL** The queue definition has cell scope. This means that the queue is known to all of the queue managers in the cell, and 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.

This value is valid only if a name service supporting a cell directory (for example, the supplied DCE name service) has been configured.

This attribute is only supported on Digital OpenVMS, OS/2 Warp, UNIX systems, and Windows NT.

**XMITQ**(*string*)

The name of the transmission queue to be used for forwarding messages to the remote queue, for either a remote queue or for a queue-manager alias definition.

If XMITQ is blank, a queue with the same name as RQMNAME is used instead as the transmission queue.

This attribute 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.

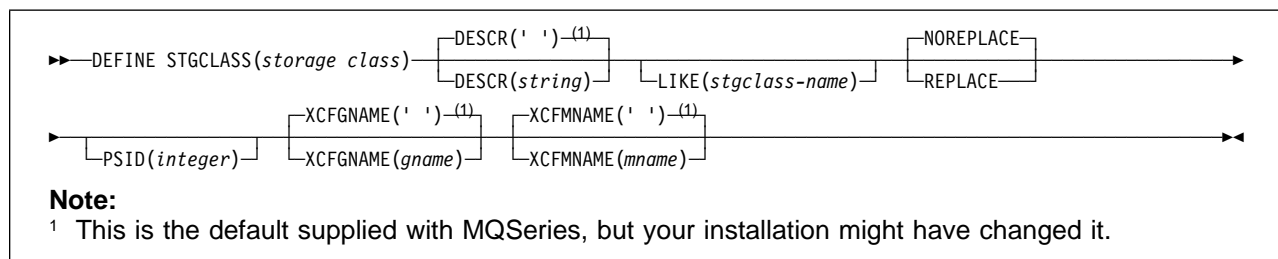
## DEFINE STGCLASS

### DEFINE STGCLASS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DEFINE STGCLASS to define a storage class to page set mapping.

**Synonym:** DEF STC



## Keyword and parameter descriptions

### *(storage-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.

**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.

### **DESCR**(*description*)

Plain-text comment. It provides descriptive information about the object when an operator issues the DISPLAY STGCLASS command.

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

### **LIKE**(*stgclass-name*)

The name of an object of the same type, whose attributes will be used to model this definition.

If this field is not filled in, and you do not complete the attribute fields related to the command, the values are taken from the default definition for this object.

This is equivalent to specifying:

```
LIKE(SYSTEMST)
```

This default storage class definition can be altered by your installation to the default values required.

**NOREPLACE** and **REPLACE**

Whether the existing definition is to be replaced with this one. This is optional, the default is NOREPLACE.

**NOREPLACE** The definition should not replace any existing definition of the same name.

**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 storage class must be empty.

**PSID**(*integer*)

The page set identifier that this storage class is to be associated with. If you do not specify this, the value is taken from the default storage class SYSTEMST.

**Note:** No check is made that the page set has been defined; an error will be 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 88.

**XCFGNAME**(*group name*)

If you are using the IMS bridge, this is the name of the XCF group to which the IMS system belongs. (This is the group name specified in the IMS parameter list.)

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.

**XCFMNAME**(*member name*)

If you are using the IMS bridge, this is the XCF member name of the IMS system within the XCF group specified in XCFGNAME. (This is the member name specified in the IMS parameter list.)

This is 1 through 16 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 through 9.

**Usage notes**

- The resultant values of XCFGNAME and XCFMNAME must either both be blank or both be nonblank.

## DELETE CHANNEL

### DELETE CHANNEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DELETE CHANNEL to delete a channel definition.

#### Notes for MVS/ESA users:

1. This is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.
2. The command fails if the channel initiator has not been started, or the channel status is RUNNING, except for client-connection channels which can be deleted without the channel initiator running.
3. You cannot delete the default channel definitions.

**Synonym:** DELETE CHL

►► DELETE CHANNEL (*channel-name*) CHLTABLE(QMGR TBL)-(1)  
CHLTABLE(CLNT TBL)-(1) ◄◄

**Note:**  
<sup>1</sup> Not valid on OS/400.

### Keyword and 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.

**QMGR TBL** 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.

**CLNT TBL** This is the channel table for CLNTCONN channels. On Digital OpenVMS, OS/2 Warp, Tandem NSK, Windows NT, and UNIX systems this is a system-wide, queue-manager independent, channel table with a system-defined name. On MVS/ESA, this is associated with the target queue manager, but separate from the main channel table.

This parameter is not supported on OS/400.

---

## DELETE NAMELIST

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DELETE NAMELIST to delete a namelist definition.

**Synonym:** DELETE NL

▶—DELETE NAMELIST (*name*)—◀

### Keyword and parameter descriptions

You must specify which namelist definition you want to delete. You cannot delete the default namelist SYSTEM.DEFAULT.NAMELIST.

(*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.

## DELETE PROCESS

---

### DELETE PROCESS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DELETE PROCESS to delete a process definition.

**Synonym:** DELETE PRO

▶—DELETE PROCESS(*process-name*)—————▶

### Keyword and parameter descriptions

You must specify which process definition you want to delete. On MVS/ESA, you cannot delete the default process SYSTEM.DEFAULT.PROCESS.

(*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.

---

## DELETE QALIAS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DELETE QALIAS to delete an alias queue definition.

**Synonym:** DELETE QA

▶—DELETE QALIAS(*q-name*)—▶

### Keyword and parameter descriptions

You must specify which alias queue you want to delete. On MVS/ESA you cannot delete the default alias queue SYSTEM.DEFAULT.ALIAS.QUEUE.

*(q-name)*

The local name of the alias queue to be deleted. The name must be defined to the local queue manager.

If an application has this queue open, or has a queue open that eventually resolves to this queue, the command fails.

If this queue has a SCOPE attribute of CELL, the entry for the queue is also deleted from the cell directory.

## DELETE QLOCAL

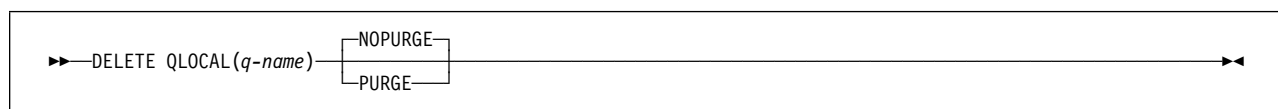
---

### DELETE QLOCAL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

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



### Keyword and parameter descriptions

You must specify which local queue you want to delete. On MVS/ESA you cannot delete the default local queue SYSTEM.DEFAULT.LOCAL.QUEUE, or other system-defined queues; for example you cannot delete the system-command input queue SYSTEM.COMMAND.INPUT.

*(q-name)*

The name of the local queue to be deleted. The name must be defined to the local queue manager.

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.

#### **NOPURGE** and **PURGE**

Specifies whether or not 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.

**NOPURGE** The deletion is not to go ahead if there are any committed messages on the named queue.

**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.

**Note:** A queue cannot be deleted if it contains uncommitted messages.



---

## DELETE QMODEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DELETE QMODEL to delete a model queue definition.

**Synonym:** DELETE QM

▶—DELETE QMODEL(*q-name*)—◀

### Keyword and parameter descriptions

You must specify which model queue you want to delete. On MVS/ESA you cannot delete the default model queue SYSTEM.DEFAULT.MODEL.QUEUE. or other system-defined model queues like the system command reply-to model queue SYSTEM.COMMAND.REPLY.MODEL.

*(q-name)*

The local name of the model queue to be deleted. The name must be defined to the local queue manager.

## DELETE QREMOTE

---

### DELETE QREMOTE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

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(*q-name*)—▶

### Keyword and parameter descriptions

You must specify which remote queue you want to delete. On MVS/ESA you cannot delete the default remote queue SYSTEM.DEFAULT.REMOTE.QUEUE.

*(q-name)*

The local name of the remote queue to be deleted. The name must be defined to the local queue manager.

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 an application has a queue open which resolved 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.

If this queue has a SCOPE attribute of CELL, the entry for the queue is also deleted from the cell directory.

---

## DELETE STGCLASS

Digital OpenVMS	MVS/ESA	OS/400	OS/2	Tandem NSK	UNIX systems	Windows NT
	√					

Use DELETE STGCLASS to delete a storage class definition

**Synonym:** DELETE STC

▶—DELETE STGCLASS(*name*)—◀

### Keyword and parameter descriptions

You must specify which storage class definition you want to delete. You cannot delete the default storage class SYSTEMST.

All queues that use the storage class must be empty and closed.

(*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.

## DISPLAY CHANNEL

---

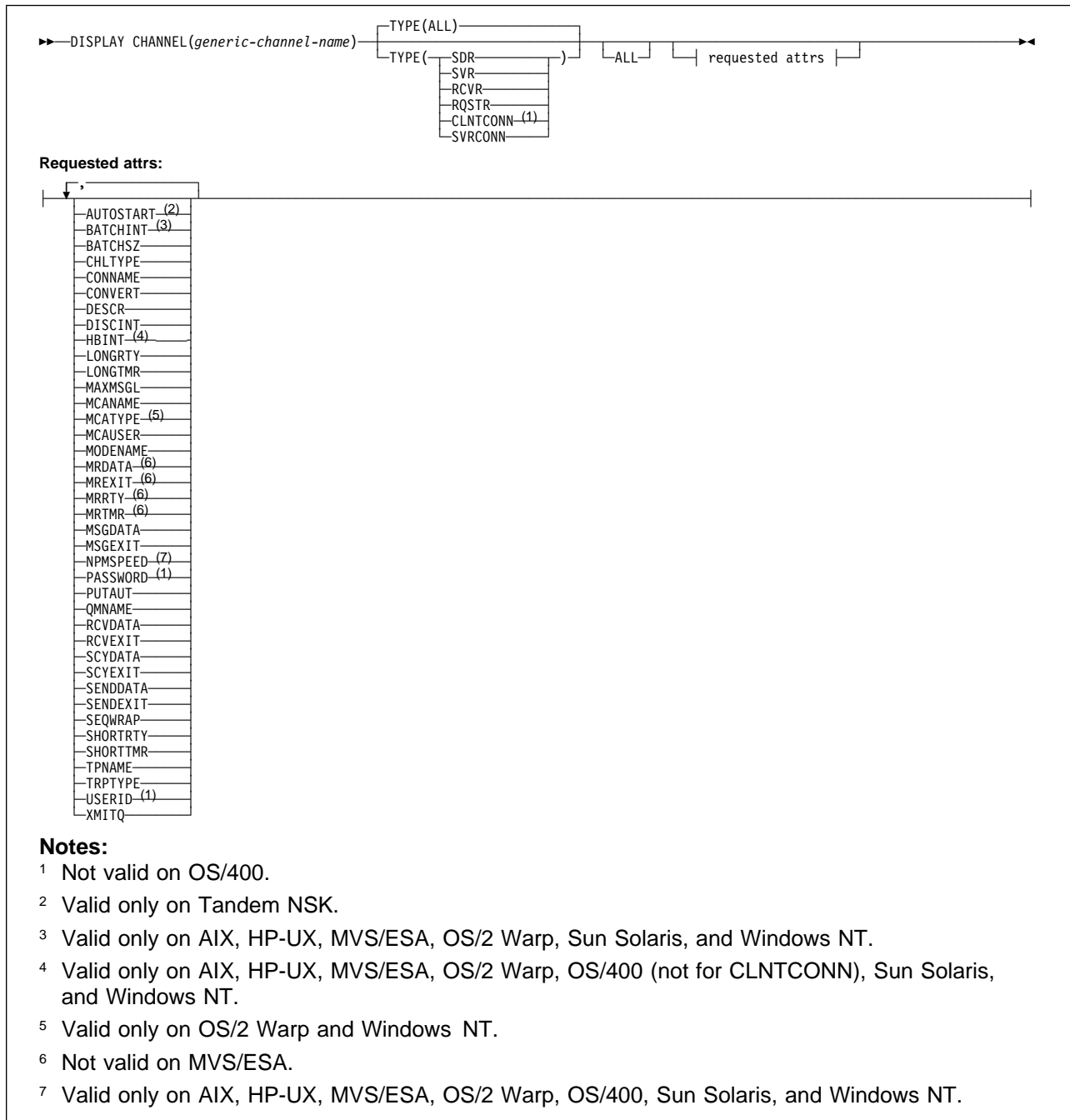
### DISPLAY CHANNEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DISPLAY CHANNEL to display a channel definition.

**Note:** On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.

**Synonym:** DIS CHL



## Keyword and parameter descriptions

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 MQSeries objects” on page 5). 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. The names must all be defined to the local queue manager.

## DISPLAY CHANNEL

**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 (excluding client-connection channels) are displayed (this is the default). On MVS/ESA, client connection channels are also displayed.

**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 (this is not valid on OS/400).

**SVRCONN** Server-connection channels only are displayed.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, *CHLTYPE(type)* can be used as a synonym for this parameter.

**ALL** Specify this to cause all attributes to be displayed. If this keyword is specified, any attributes that are also requested specifically have no effect; all attributes are still displayed.

On AIX, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT, this is the default if you do not specify a generic name, and do not request any specific attributes.

If no attributes are specified (and the ALL keyword is not specified or defaulted), the default is that the channel names only are displayed. On MVS/ESA, the *CHLTYPE* is also displayed.

**Requested attributes:** Specify one or more attributes that define the data to be displayed. You can specify the attributes in any order, but do not specify the same attribute more than once.

Some attributes 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.

| **AUTOSTART** Whether an LU 6.2 responder process should be started for the channel

**BATCHINT** Minimum batch duration

**BATCHSZ** Batch size

**CHLTYPE** Channel type.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT the channel type is always displayed if you specify a generic channel name and do not request any other attributes. On MVS/ESA, the channel type is always displayed.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, *TYPE(type)* can be used as a synonym for this parameter.

**CONNAME** Connection name

**CONVERT** Whether sender should convert application message data

**DESCR** Description

**DISCINT** Disconnection interval

| **HBINT** Heartbeat interval

**LONGRTY** Long retry count

**LONGTMR** Long retry timer

**MAXMSGL** Maximum message length for channel

<b>MCANAME</b>	Message channel agent name
<b>MCATYPE</b>	Whether message channel agent runs as a separate process or a separate thread
<b>MCAUSER</b>	Message channel agent user identifier
<b>MODENAME</b>	LU 6.2 mode name
<b>MRDATA</b>	Channel message-retry exit user data
<b>MREXIT</b>	Channel message-retry exit name
<b>MRRTY</b>	Channel message-retry exit retry count
<b>MRTMR</b>	Channel message-retry exit retry time
<b>MSGDATA</b>	Channel message exit user data
<b>MSGEXIT</b>	Channel message exit names
<b>NPMSPEED</b>	Nonpersistent message speed
<b>PASSWORD</b>	Password for initiating LU 6.2 session (if nonblank, this is displayed as asterisks)
<b>PUTAUT</b>	Put authority
<b>QMNAME</b>	Queue manager name
<b>RCVDATA</b>	Channel receive exit user data
<b>RCVEXIT</b>	Channel receive exit names
<b>SCYDATA</b>	Channel security exit user data
<b>SCYEXIT</b>	Channel security exit name
<b>SENDDATA</b>	Channel send exit user data
<b>SENDEXIT</b>	Channel send exit names
<b>SEQWRAP</b>	Sequence number wrap value
<b>SHORTRTY</b>	Short retry count
<b>SHORTTMR</b>	Short retry timer
<b>TPNAME</b>	LU 6.2 transaction program name
<b>TRPTYPE</b>	Transport type
<b>USERID</b>	User identifier for initiating LU 6.2 session
<b>XMITQ</b>	Transmission queue name

## DISPLAY CHSTATUS

### DISPLAY CHSTATUS

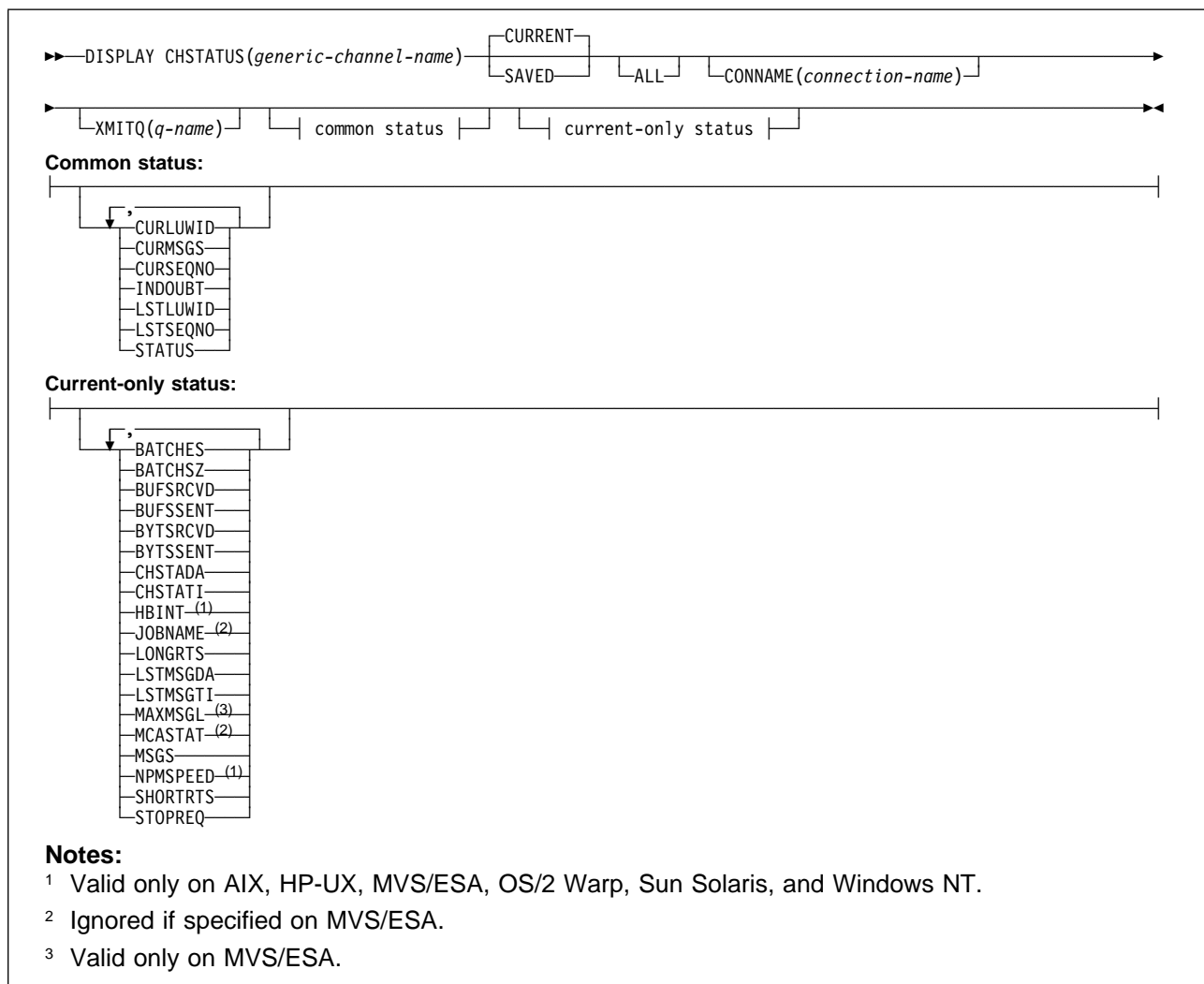
Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	UNIX systems	Tandem NSK	Windows NT
√	√	√	√	√	√	√

Use DISPLAY CHSTATUS to display the status of one or more channels.

#### Notes:

1. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS.
2. This command cannot be used for CLNTCONN channels.
3. On MVS/ESA, the command fails if the channel initiator has not been started.

**Synonym:** DIS CHS





## Keyword and parameter descriptions

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:

- 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 current status data (of current channels only), or
- The saved status data of all channels.

Before explaining the syntax and options for this command, it is necessary to describe the format of the status data that is available for channels and the states that channels can have.

There are two classes of data available for channel status. These are **saved** and **current**. 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. This data is reset at the following times:
  - For all channels:
    - When the channel enters or leaves STOPPED or RETRY state
    - On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, when the queue manager is ended
  - 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 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 will 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.

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.
- An current channel always has current status and might have saved status.

## DISPLAY CHSTATUS

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 might also have **saved** status.

The term **Active** is used to describe the set of current channels which 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 a receiver, requester, 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 given 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 in connection 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 which are displayed for a given channel can be limited by using the XMITQ, CONNAME, and CURRENT keywords 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. The channels must all be defined to the local queue manager.

### XMITQ(*q-name*)

The name of the transmission queue for which status information is to be displayed, for the specified channel or channels.

This keyword 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.

**CONNNAME**(*connection-name*)

The connection name for which status information is to be displayed, for the specified channel or channels.

This keyword can be used to limit the number of sets of status information that is displayed. If it is not specified, the display is not limited in this way.

The value returned for CONNNAME can not be the same as in the channel definition, and might differ between the current channel status and the saved channel status. (Using CONNNAME for limiting the number of sets of status is therefore not recommended.) For example, if CONNNAME is blank in the channel definition or (when using TCP/IP) is in "host name" format, the channel status value will have the resolved network address; if the CONNNAME includes the port number (again when using TCP/IP), the current channel status value will include the port number, but the saved channel status value will not.

This value could also be the queue manager name of the remote system.

**CURRENT**

This is the default, and indicates that current status information for current channels only is to be displayed.

Both common and current-only status information can be requested for current channels.

**SAVED** Specify this to cause saved status information for both current and inactive channels to be displayed.

Only common status information can be displayed. Current-only status information is not displayed for current channels if this keyword is specified.

**ALL** Specify this to display all of 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 keyword is specified, any keywords requesting specific status information that are also specified have no effect; all of the information is displayed.

The following information is always returned, for each set of status information:

- The channel name
- The channel type
- The transmission queue name (for sender and server channels)
- The connection name
- The type of status information returned (CURRENT or SAVED)
- On MVS/ESA, STATUS

If no keywords requesting specific status information are specified (and the ALL keyword is not specified), no further information is returned.

If status information is requested which is not relevant for the particular channel type, this is not an error.

**Common status:** The following information applies to all sets of channel status, whether or not the set is current. The information applies to all channel types except server-connection.

**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.

## DISPLAY CHSTATUS

For a saved channel instance, this attribute has meaningful information only if the channel instance is in doubt. However, the attribute 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.

**CURMSGGS** 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 attribute has meaningful information only if the channel instance is in doubt. However, the attribute 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.

**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 attribute has meaningful information only if the channel instance is in doubt. However, the attribute 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.

**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, which 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.

**LSTLUWID** The logical unit of work identifier associated with the last committed batch of messages transferred.

**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.

**STATUS** Current status of the channel. This is one of the following:

### **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.

### **BINDING**

Channel is performing channel negotiation and is not yet ready to transfer messages.

### **INITIALIZING**

The channel initiator is attempting to start a channel. This is valid only on AIX, Digital OpenVMS, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT. On MVS/ESA, this 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**

Channel is stopping or a close request has been received.

**RETRYING**

A previous attempt to establish a connection has failed. The MCA will re-attempt connection after the specified time interval.

**PAUSED**

The channel is waiting for the message-retry interval to complete before retrying an MQPUT operation. This is not valid on MVS/ESA.

**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.

**REQUESTING**

A local requester channel is requesting services from a remote MCA.

**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.

**BATCHES** Number of completed batches during this session (since the channel was started).

**BATCHSZ** The batch size being used for this session (valid only on MVS/ESA, AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT).

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.

**BUFSENT** 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.

**BYTSENT** 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).

**JOBNAME** Name of job currently serving the channel.

- On Digital OpenVMS and UNIX systems, this is the process identifier, displayed in hex.
- On OS/2 Warp and Windows NT, this is the concatenation of the process identifier and the thread identifier of the MCA program, displayed in hex.

## DISPLAY CHSTATUS

- On Tandem NSK, this is the CPU ID and PID, displayed in decimal.  
This information is not available on MVS/ESA. The keyword is ignored if specified.
- HBINT** The heartbeat interval being used for this session.
- 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.
- MAXMSGGL** The maximum message length being used for this session (valid only on MVS/ESA).
- 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 MVS/ESA. The keyword is ignored if specified.
- 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).
- NPMSPEED** The nonpersistent message handling technique being used for this session.
- SHORTRTS** Number of short retry wait start attempts left. This applies only to sender or server channels.
- STOPREQ** Whether a user stop request is outstanding. This is either YES or NO.

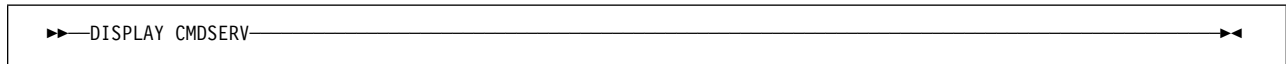
---

## DISPLAY CMDSERV

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY CMDSERV to display the status of the command server.

**Synonym:** DIS CS



### Usage notes

The command server takes messages from the system command input queue and processes them. DISPLAY CMDSERV displays the status of the command server.

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 messages  
**DISABLED** Not available to process messages  
**STARTING** START CMDSERV in progress  
**STOPPING** STOP CMDSERV in progress  
**STOPPED** STOP CMDSERV completed  
**RUNNING** Processing a message  
**WAITING** Waiting for a message

---

### DISPLAY DQM

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY DQM to display information about the channel initiator.

**Note:** This is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.

**Synonym:** DIS DQM

```
▶▶—DISPLAY DQM—◀◀
```

### Usage notes

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
- Whether the TCP/IP listener is started or not, and what port it is using
- Whether the LU 6.2 listener is started or not, and what LU name it is using
- How many dispatchers are started, and how many were requested
- How many adapter subtasks are started, and how many were requested
- The TCP/IP address space name
- How many channel connections are current, and whether they are active, stopped, or retrying
- The maximum number of current connections



## DISPLAY MAXSMSGS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY MAXSMSGS to see the maximum number of messages that a task can get or put within a single unit of recovery.

**Note:** This command is valid only on MVS/ESA. For other platforms, use the MAXUMSGS keyword of the DISPLAY QMGR command instead.

You can issue the DISPLAY MAXSMSGS command at any time to see the number of messages allowed.

**Synonym:** DIS MAXSM

```
▶—DISPLAY MAXSMSGS—▶
```

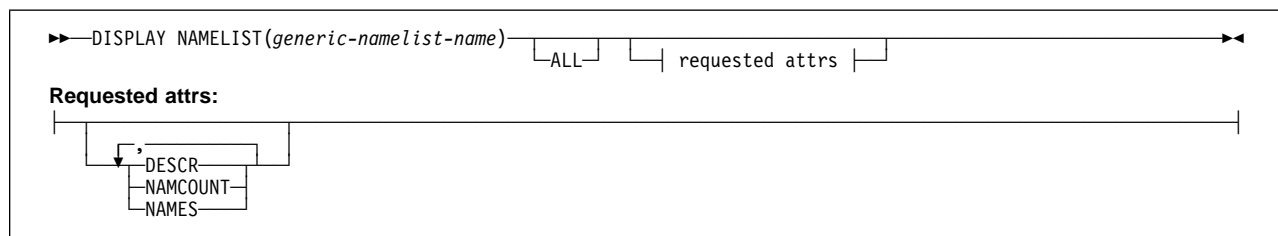
## DISPLAY NAMELIST

### DISPLAY NAMELIST

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY NAMELIST to display the names of queues in a namelist.

**Synonym:** DIS NL



### Keyword and parameter descriptions

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 MQSeries objects” on page 5). A trailing asterisk (\*) matches all namelists with the specified stem followed by zero or more characters. An asterisk (\*) on its own specifies all namelists. The namelists must all be defined to the local queue manager.

**ALL** Specify this to display all the attributes. If this keyword is specified, any attributes that are requested specifically have no effect; all the attributes are displayed.

This is the default if you do not specify a generic name, and do not request any specific attributes.

**Requested attributes:** You can request the following information for each namelist definition:

<b>DESCR</b>	Description
<b>NAMCOUNT</b>	Number of queue names in the list
<b>NAMES</b>	List of queue names

See DEFINE NAMELIST on page 83 for more information about the DESCR and NAMES attributes.



## DISPLAY QMGR

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DISPLAY QMGR to display the queue manager attributes for this queue manager.

**Synonym:** DIS QMGR

```

▶▶ DISPLAY QMGR [ALL] [requested attrs]
Requested attrs:
AUTHOREV
CCSID
CHAD (1)
CHADEV (1)
CHADEXIT (1)
CMDLEVEL
COMMANDQ
CPILEVEL (2)
DEADQ
DEFXMITQ
DESCR
DISTL (1)
INHIBTEV
LOCALEV
MAXHANDS
MAXMSGL
MAXPRTY
MAXUMSGS (3)
PERFMEV
PLATFORM
QMNAME
REMOTEEV
STRSTPEV
SYNCPT
TRIGINT
    
```

**Notes:**

- 1 Valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
- 2 Valid only on MVS/ESA.
- 3 Not valid on MVS/ESA.

## Keyword and parameter descriptions

**ALL** Specify this to cause all attributes to be displayed. If this keyword is specified, any attributes that are requested specifically have no effect; all attributes are still displayed.

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, this is the default if you do not request any specific attributes.

**Requested attributes:** Specify one or more attributes that define the data to be displayed. The attributes can be specified in any order, but do not specify the same attribute more than once.

**Note:** If no attributes are specified (and the ALL keyword is not specified or defaulted), the queue manager name is returned.

You can request the following information for the queue manager:

<b>AUTHOREV</b>	Whether authorization events are generated.
<b>CCSID</b>	Coded character set identifier. This 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.
<b>CHAD</b>	Whether auto-definition of receiver and server-connection channels is enabled. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
<b>CHADEV</b>	Whether auto-definition events are enabled. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
<b>CHADEXIT</b>	The name of the channel auto-definition exit. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
<b>CMDLEVEL</b>	Command level. This indicates the function level of the queue manager.
<b>COMMANDQ</b>	The name of the system-command input queue. Suitably authorized applications can put commands on this queue.
<b>CPILEVEL</b>	Reserved, this value has no significance.
<b>DEADQ</b>	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 needs to 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 which 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 attribute.

<b>DEFXMITQ</b>	Default transmission queue name. This is the transmission queue on which messages, destined for a remote queue manager, are put if there is no other suitable transmission queue defined.
<b>DESCR</b>	Description.
<b>DISTL</b>	Whether distribution lists are supported by the queue manager. This is valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
<b>INHIBTEV</b>	Whether inhibit events are generated.

## DISPLAY QMGR

<b>LOCALEV</b>	Whether local error events are generated.
<b>MAXHANDS</b>	The maximum number of open handles that any one task can have at any one time.
<b>MAXMSGL</b>	The maximum message length that can be handled by the queue manager. Individual queues might have a smaller or greater maximum than this. (See the description of MAXMSGL under "ALTER CHANNEL" on page 11 for more information.)
<b>MAXPRTY</b>	The maximum priority. This is 9.
<b>MAXUMSGS</b>	Maximum number of uncommitted messages within one syncpoint. This keyword is not supported on MVS/ESA; use DISPLAY MAXSMSGS instead.
<b>PERFMEV</b>	Whether performance-related events are generated.
<b>PLATFORM</b>	The architecture of the platform on which the queue manager is running. This is MVS, OPENVMS, NSK, OS2, OS400, UNIX, or WINDOWSNT.
<b>QMNAME</b>	The name of the local queue manager. See "Rules for naming MQSeries objects" on page 5.
<b>REMOTEEV</b>	Whether remote error events are generated.
<b>STRSTPEV</b>	Whether start and stop events are generated.
<b>SYNCPT</b>	Whether syncpoint support is available with the queue manager. On MVS/ESA, UNIX systems, OS/2 Warp, and Windows NT it is always available.
<b>TRIGINT</b>	The trigger interval.

## DISPLAY QUEUE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DISPLAY QUEUE to display the attributes of one or more queues of any type.

On AIX, HP-UX, MVS/ESA, OS/2 Warp, OS/400, Sun Solaris, and Windows NT, you can use DISPLAY QLOCAL, DISPLAY QALIAS, DISPLAY QMODEL, or DISPLAY QREMOTE (or their synonyms) as an alternative way to display these attributes. These commands produce the same output as the DISPLAY QUEUE TYPE(*q-type*) command. If you enter the commands this way, do not use the TYPE keyword because this causes an error.

**Synonym:** DIS Q

▶—DISPLAY QUEUE(*generic-q-name*) [ALL] [STGCLASS *(generic-name)*(1)] [TYPE(*queue-type*)]

requested attrs

**Requested attrs:**

- BOQNAME
- BOTHRESH
- CRDATE
- CRTIME
- CURDEPTH
- DEFPRTY
- DEFPST
- DEFSOPT
- DEFTYPE
- DESCR
- DISTL (2)
- GET
- HARDENBO
- INDXTYPE (1)
- INITQ
- IPPROCS
- MAXDEPTH
- MAXMSGL
- MSGDLVSQ
- OPPROCS
- PROCESS
- PUT
- QDEPTHHI
- QDEPTHLO
- QDPHIEV
- QDPLOEV
- QDPMAXEV
- QSVCEV
- QSVCIINT
- QTYPE
- RETIINTVL
- RNAME
- RQMNAME
- SCOPE (3)
- SHARE
- TARGQ
- TRIGDATA
- TRIGDPH
- TRIGGER
- TRIGMPRI
- TRIGTYPE
- USAGE
- XMITQ

**Notes:**

- 1 Valid only on MVS/ESA.
- 2 Valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
- 3 Not valid on MVS/ESA or OS/400.

## Keyword and parameter descriptions

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

*(generic-q-name)*

The local name of the queue definition to be displayed (see “Rules for naming MQSeries objects” on page 5). 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 names must all be defined to the local queue manager.

**ALL** Specify this to cause all attributes to be displayed. If this keyword is specified, any attributes that are also requested specifically have no effect; all attributes are still displayed.

On AIX, HP-UX, MVS/ESA, OS/2 Warp, Sun Solaris, and Windows NT, this is the default if you do not specify a generic name, and do not request any specific attributes.

**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 attribute, and storage class information is returned about all the queues displayed.

This keyword is valid only on MVS/ESA.

**TYPE***(queue-type)*

This is optional, and specifies the type of queues you want to be displayed. The default is to display all queue types. If specified, it must be the same as the primary keyword (or synonym) used on the DEFINE command for that queue type (for example, QLOCAL or QL).

On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, QTYPE(*type*) can be used as a synonym for this parameter.

If no attributes are specified (and the ALL keyword is not specified or defaulted), the queue name and queue type are displayed.

**Requested attributes:** Specify one or more attributes that define the data to be displayed. The attributes can be specified in any order, but do not specify the same attribute more than once.

Most attributes are relevant only for queues of a particular type or types. Attributes that are not relevant for a particular type of queue cause no output, nor is an error raised. Table 3 shows the attributes that are relevant for each type of queue. There is a brief description of each attribute after the table, but for more information, see the DEFINE command for each queue type.

<i>Table 3 (Page 1 of 2). Attributes that can be returned by the DISPLAY QUEUE command</i>				
	<b>Local queue</b>	<b>Model queue</b>	<b>Alias queue</b>	<b>Remote queue</b>
BOQNAME	√	√		
BOTHRESH	√	√		
CRDATE	√	√		
CRTIME	√	√		
CURDEPTH	√			
DEFPRTY	√	√	√	√



Table 3 (Page 2 of 2). Attributes that can be returned by the DISPLAY QUEUE command

	Local queue	Model queue	Alias queue	Remote queue
DEFPSIST	√	√	√	√
DEFSOPT	√	√		
DEFTYPE	√	√		
DESCR	√	√	√	√
DISTL <sup>1</sup>	√	√		
GET	√	√	√	
HARDENBO	√	√		
INDXTYPE <sup>2</sup>	√	√		
INITQ	√	√		
IPPROCS	√			
MAXDEPTH	√	√		
MAXMSGL	√	√		
MSGDLVSQ	√	√		
OPPROCS	√			
PROCESS	√	√		
PUT	√	√	√	√
QDEPTHHI	√	√		
QDEPTHLO	√	√		
QDPHIEV	√	√		
QDPLOEV	√	√		
QDPMAXEV	√	√		
QSVCI EV	√	√		
QSVCI NT	√	√		
QTYPE	√	√	√	√
RETINTVL	√	√		
RNAME				√
RQMNAME				√
SCOPE <sup>3</sup>	√		√	√
SHARE	√	√		
STGCLASS <sup>2</sup>	√	√		
TARGQ			√	
TRIGDATA	√	√		
TRIGDPTH	√	√		
TRIGGER	√	√		
TRIGMPRI	√	√		
TRIGTYPE	√	√		
USAGE	√	√		
XMITQ				√

**Notes:**

1. Supported only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT
2. Supported only on MVS/ESA
3. Not supported on MVS/ESA or OS/400

## DISPLAY QUEUE

<b>BOQNAME</b>	Backout requeue name.
<b>BOTHRESH</b>	Backout threshold.
<b>CRDATE</b>	The date on which the queue was defined (in the form yyyy-mm-dd).
<b>CRTIME</b>	The time at which the queue was defined (in the form hh.mm.ss).
<b>CURDEPTH</b>	Current depth of queue.
<b>DEFPRTY</b>	Default priority of the messages put on the queue.
<b>DEFPSIST</b>	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.
<b>DEFSOPT</b>	Default share option on a queue opened for input.
<b>DEFTYPE</b>	Queue definition type. This can be: <ul style="list-style-type: none"><li>• <b>PREDEFINED</b> (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.</li><li>• <b>PERMDYN</b> (Permanent dynamic) Either the queue was created by an application issuing <b>MQOPEN</b> 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.</li><li>• <b>TEMPDYN</b> (Temporary dynamic) Either the queue was created by an application issuing <b>MQOPEN</b> 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.</li></ul>
<b>DESCR</b>	Descriptive comment.
<b>DISTL</b>	Whether distribution lists are supported by the partner queue manager. (Supported only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.)
<b>GET</b>	Whether the queue is enabled for gets.
<b>HARDENBO</b>	Whether to harden the get back out count.
<b>INDXTYPE</b>	Index type (supported only on MVS/ESA).
<b>INITQ</b>	Initiation queue name.
<b>IPPROCS</b>	Number of handles indicating that the queue is open for input.
<b>MAXDEPTH</b>	Maximum depth of queue.
<b>MAXMSGL</b>	Maximum message length.
<b>MSGDLVSQ</b>	Message delivery sequence.
<b>OPPROCS</b>	Number of handles indicating that the queue is open for output.
<b>PROCESS</b>	Process name.
<b>PUT</b>	Whether the queue is enabled for puts.
<b>QDEPTHHI</b>	Queue Depth High event generation threshold.
<b>QDEPTHLO</b>	Queue Depth Low event generation threshold.
<b>QDPHIEV</b>	Whether Queue Depth High events are generated.
<b>QDPLOEV</b>	Whether Queue Depth Low events are generated.

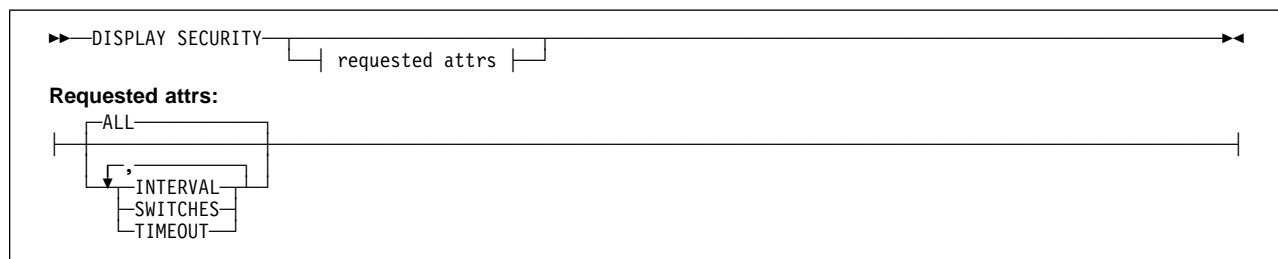
<b>QDPMAXEV</b>	Whether Queue Full events are generated.
<b>QSVCIEV</b>	Whether service interval events are generated.
<b>QSVCINT</b>	Service interval event generation threshold.
<b>QTYPE</b>	Queue type.  On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT the queue type is always displayed if you specify a generic queue name and do not request any other attributes. On MVS/ESA, the queue type is always displayed.  On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, TYPE( <i>type</i> ) can be used as a synonym for this parameter.
<b>RETINTVL</b>	Retention interval.
<b>RNAME</b>	Name of the local queue, as known by the remote queue manager.
<b>RQMNAME</b>	Remote queue manager name.
<b>SCOPE</b>	Scope of queue definition (not supported on MVS/ESA or OS/400).
<b>SHARE</b>	Whether the queue can be shared.
<b>STGCLASS</b>	Storage class.
<b>TARGQ</b>	Local name of aliased queue.
<b>TRIGDATA</b>	Trigger data.
<b>TRIGDPTH</b>	Trigger depth.
<b>TRIGGER</b>	Whether triggers are active.
<b>TRIGMPRI</b>	Threshold message priority for triggers.
<b>TRIGTYPE</b>	Trigger type.
<b>USAGE</b>	Whether or not the queue is a transmission queue.
<b>XMITQ</b>	Transmission queue name.

## DISPLAY SECURITY

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY SECURITY to display the current settings for the security attributes.

**Synonym:** DIS SEC



### Keyword and parameter descriptions

**ALL** Display the TIMEOUT, INTERVAL, and SWITCHES attributes. This is the default if no requested attributes are specified.

#### INTERVAL

Time interval between checks.

#### SWITCHES

Display the current setting of the switch profiles.

If the ssid.NO.SUBSYS.SECURITY switch is off, no other switch profile settings are displayed.

If the ssid.NO.SUBSYS.SECURITY switch is on, the following switch profile settings are displayed:

- ssid.NO.SUBSYS.SECURITY (subsystem security)
- ssid.NO.CONNECT.CHECKS (connection security)
- ssid.NO.CMD.CHECKS (command security)
- ssid.NO.CMD.RESC.CHECKS (command resource security)
- ssid.NO.QUEUE.CHECKS (queue security)
- ssid.NO.PROCESS.CHECKS (process security)
- ssid.NO.NLIST.CHECKS (namelist security)
- ssid.NO.CONTEXT.CHECKS (context security)
- ssid.NO.ALTERNATE.USER.CHECKS (alternate user security)

**TIMEOUT** Timeout value.

See ALTER SECURITY for details of the TIMEOUT and INTERVAL attributes.



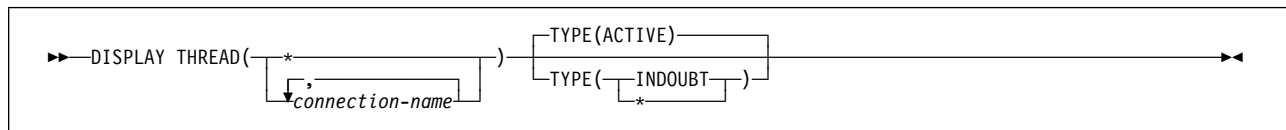
## DISPLAY THREAD

### DISPLAY THREAD

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY THREAD to display information about active and in-doubt threads. Threads shown as in doubt on one invocation of this command will probably be resolved for subsequent invocations.

**Synonym:** DIS THD



### Keyword and parameter descriptions

(*connection-name*)

List of one or more *connection-names* (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

Threads are selected from the address spaces associated with these connections only.

(\*) Displays threads associated with all connections to MQSeries.

A *connection-name* or \* must be used; no default is available.

**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 MQSeries 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 MQSeries on its behalf. External intervention is needed to resolve the status of in-doubt threads. They might have been in doubt at the last restart, or they might have become in doubt since the last restart.

Batch programs do not participate in two-phase commit for MQSeries resources and therefore never have in-doubt threads.

\* Display both active and in-doubt threads.

If, during command processing, an active thread becomes in doubt, it might appear twice: once as active and once as in doubt.

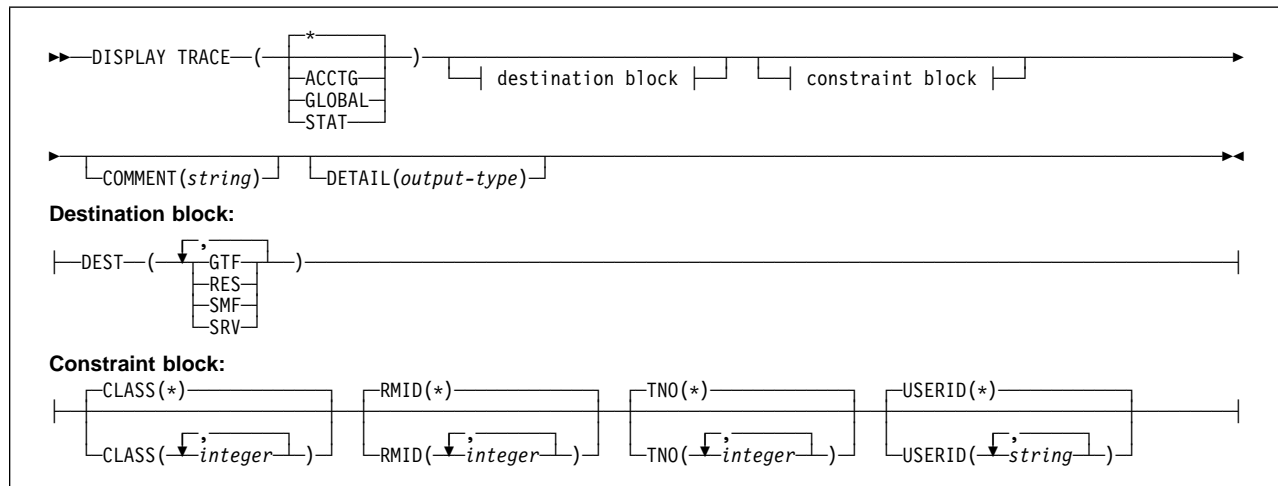
The response to this command is a list of the current unit-of-recovery IDs. For details of the information returned by the DISPLAY THREAD command, see messages CSQV401I through CSQV406I in the *MQSeries for MVS/ESA Messages and Codes* manual.

## DISPLAY TRACE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY TRACE to display a list of active traces.

**Synonym:** DIS TRACE



## Keyword and parameter descriptions

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 of the remaining keywords in this section limits the list to traces of the corresponding type:

**ACCTG** Accounting data (the synonym is A)

**GLOBAL** Service data from the entire MQSeries subsystem (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)**

Limits the information that a trace displays based on the *output-type* specified.

Possible values for *output-type* are:

- 1 Display summary trace information: TNO, TYPE, CLASS, and DEST
- 2 Display qualification trace information: TNO and RMID. Refer to message CSQW1271 (in the *MQSeries for MVS/ESA Messages and Codes* manual) for more information about trace qualification.
- 1,2 Display both summary and qualification information
- \* Display both summary and qualification information

## DISPLAY TRACE

If no parameter follows DETAIL (either DETAIL() or just DETAIL is used), type 1 trace information is displayed.

### 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 wrap-around table residing in the ECSA (extended common service area)
- SMF** The System Management Facility
- SRV** A serviceability routine designed for IBM for problem diagnosis

See START TRACE on page 167 for a list of allowed destinations for each trace type.

### Constraint block

#### **CLASS**(*integer*)

Limits the list to traces started for particular classes. See START TRACE on page 167 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 167 for a list of allowed resource manager identifiers. Do not use this option with STAT.

The default is RMID(\*), which does not limit the list.

**Note:** Information about RMID 231 might be inaccurate if the trace has been altered using the ALTER TRACE command, or if the channel initiator has been stopped.

#### **TNO**(*integer*)

Limits the list to particular traces, identified by their trace number (1 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.

#### **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.

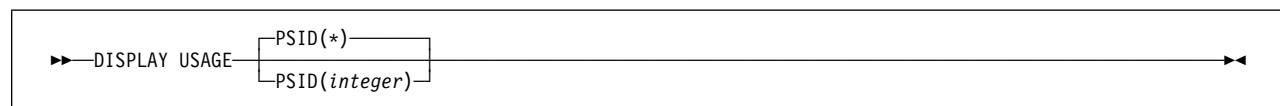


## DISPLAY USAGE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY USAGE to display information about the current state of a page set.

**Synonym:** DIS USAGE



## Keyword and parameter descriptions

### PSID(*integer*)

The page-set identifier that a storage class maps to. This is optional.

This is a number, in the range 00 through 99. An asterisk (\*) on its own specifies all page set identifiers. This is the default. See DEFINE PSID on page 88.

DISPLAY USAGE returns three sets of information in the following messages:

**CSQI018I** The number of pages currently being used on the page set specified:

- *total* is the total number of 4-KB pages in the page set (this relates to the records parameter on your VSAM definition of the page set)
- *unused* is the number of pages that are not used (that is, available page sets)
- *persist* is the number of pages holding persistent data (these pages are being used to store object definitions and persistent message data)
- *nonpersist* is the number of pages holding nonpersistent data (these pages are being used to store nonpersistent message data)

Use these figures as a guide only; they are changing constantly as applications put messages on and get messages from queues.

**CSQI030I** The number of extents at restart, and the number of times the page set has been expanded dynamically since restart.

**CSQI024I** The restart RBA (relative byte address) for the subsystem. This value can be used to determine where to truncate logs, if required.

See the *MQSeries for MVS/ESA Messages and Codes* manual for more information about these messages.

## PING CHANNEL

---

### PING CHANNEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

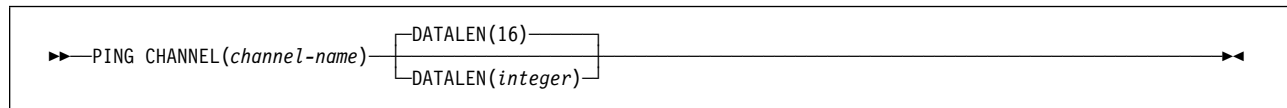
Use PING CHANNEL to test a channel by sending data as a special message to the remote queue manager, and checking that the data is returned. The data is generated by the local queue manager.

#### Notes:

1. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.
2. On MVS/ESA, the command fails if the channel initiator has not been started.

This command can be used only for sender (SDR) and server (SVR) channels. It is not valid if the channel is running; however, it is valid if the channel is stopped or in retry mode.

**Synonym:** PING CHL



### Keyword and parameter descriptions

*(channel-name)*

The name of the channel to be tested. This is required.

**DATALEN**(*integer*)

The length of the data, in the range 16 through 32 768. This is optional.

---

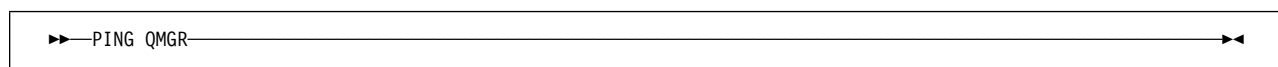
**PING QMGR**

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√		√	√	√	√	√

Use PING QMGR to test whether the queue manager is responsive to commands.

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.

**Synonym:** PING QMGR



## RECOVER BSDS

---

### RECOVER BSDS

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use RECOVER BSDS to reestablish a dual bootstrap data set (BSDS) after one has been disabled by a data set error. 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.

**Synonym:** REC BSDS

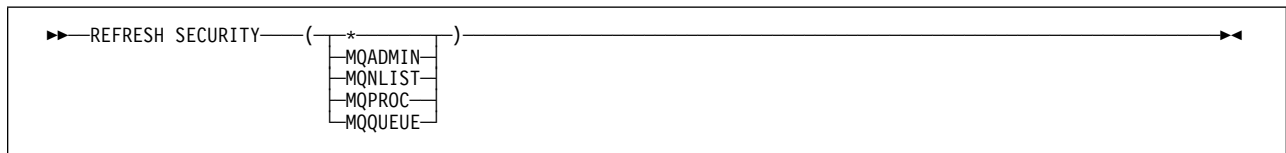
▶—RECOVER BSDS—◀

## REFRESH SECURITY

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use REFRESH SECURITY to cause a security refresh to be carried out.

**Synonym:** REF SEC



### Keyword and parameter descriptions

This command causes MQSeries to refresh in-storage ESM (external security manager, for example RACF) profiles. 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 must specify the resource type for which the security refresh is to be performed. The types are:

\* All resource types

**MQADMIN** RADMIN resources only

**MQNLIST** RNLIST resources only

**MQPROC** RPROC resources only

**MQQUEUE** RQUEUE resources 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 will also take place.

REBUILD SECURITY is another synonym for REFRESH SECURITY.

## RESET CHANNEL

### RESET CHANNEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

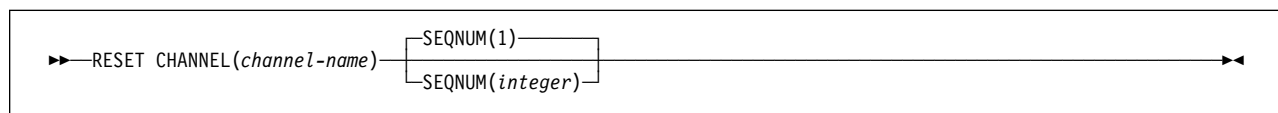
Use RESET CHANNEL to reset the message sequence number for an MQSeries channel with, optionally, a specified sequence number to be used the next time that the channel is started.

**Note:** On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.

This command can be issued to a channel of any type (except SVRCONN and CLNTCONN channels). However, if it is issued to a sender or a 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 will also be reset to the same value the next time this channel is initiated (and resynchronized if necessary).

If the command is issued to a receiver or requester channel, the value at the other end is *not* reset as well; this must be done separately if necessary.

**Synonym:** RESET CHL



### Keyword and parameter descriptions

*(channel-name)*

The name of the channel to be reset. This is required.

**SEQNUM***(integer)*

The new message sequence number, which must be greater than or equal to 1, and less than or equal to 999 999 999. This is optional.

## RESET TPIPE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

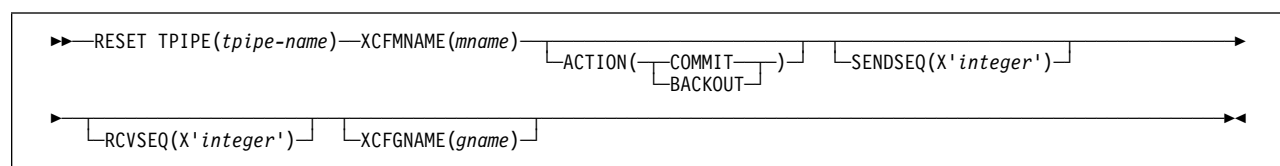
Use RESET TPIPE to reset the recoverable sequence numbers for an IMS Tpipe used by the MQSeries-IMS bridge.

This command is used in response to the resynchronization error reported in message CSQ2020E, and initiates resynchronization of the Tpipe with IMS.

### Notes:

1. The command fails if the queue manager is not connected to the specified XCF member.
2. The command fails if the queue manager is connected to the specified XCF member, but the Tpipe is open.
3. RESET TPIPE cannot be issued from the CSQINP1 and CSQINP2 initialization data sets.

**Synonym:** There is no synonym for this command.



## Keyword and parameter descriptions

### (*tpipe-name*)

The name of the Tpipe to be reset. This is required.

### XCFMNAME(*mname*)

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.

### 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 MQSeries are confirmed as having already transferred to IMS; that is, they are deleted from the MQSeries-IMS bridge queue.

**BACKOUT** The messages from MQSeries are backed out; that is, they are returned to the MQSeries-IMS bridge queue.

### SENDSEQ(*integer*)

The new recoverable sequence number to be set in the Tpipe for messages sent by MQSeries and to be set as the partner's receive sequence number. It must be hexadecimal and can be up to 8 digits long. It is optional; if omitted, the sequence number is not changed but the partner's receive sequence is set to the MQSeries send sequence number.

## RESET TPIPE

### **RCVSEQ**(*integer*)

The new recoverable sequence number to be set in the Tpipe for messages received by MQSeries and to be set as the partner's send sequence number. It must be hexadecimal and can be up to 8 digits long. It is optional; if omitted, the sequence number is not changed but the partner's send sequence is set to the MQSeries receive sequence number.

### **XCFGNAME**(*gname*)

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.



## RESOLVE CHANNEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use RESOLVE CHANNEL to request a channel to commit or back out in-doubt messages.

### Notes:

1. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.
2. On MVS/ESA, the command fails if the channel initiator has not been started.

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 doubt, as to whether or not the messages were received. Any outstanding units of work need to be resolved by being backed out or committed.

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 be used only for sender (SDR) and server (SVR) channels.

**Synonym:** RESOLVE CHL (RES CHL on MVS/ESA)

```

▶—RESOLVE CHANNEL (channel-name)—ACTION (—COMMIT—)
                                     [—BACKOUT—]
▶

```

## Keyword and parameter descriptions

*(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

## RESOLVE INDOUBT

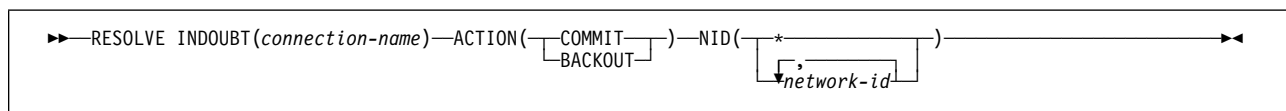
### RESOLVE INDOUBT

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use RESOLVE INDOUBT to resolve threads left in doubt because MQSeries or a transaction manager could not resolve them automatically.

This command does not apply to units of recovery associated with batch or TSO applications.

**Synonym:** RES IND



### Keyword and parameter descriptions

*(connection-name)*

1 through 8 character connection name. For a CICS connection it is the CICS applid. For an IMS adaptor connection, it is the IMS control region job name. For an IMS bridge connection, it is the MQSeries subsystem name.

#### ACTION

Specifies whether to commit or back out the in-doubt threads:

**COMMIT** Commits the threads

**BACKOUT** Backs out the threads

#### NID

Network identifier. Specifies the thread or threads to be resolved.

*(network-id)*

This is of the form

*connection-name.unit-of-recovery-id*

where:

- *Connection-name* is the 1 though 8 character connection name, as above
- *Unit-of-recovery-id* is the 1 through 16 character name of *unit-of-recovery-id* of a specific thread to be resolved

There must be a period (.) between *connection-name* and *unit-of-recovery-id*, so the maximum number of characters that can be entered for *network-id* is 25.

(\*) Resolves all threads associated with the connection.

Examples:

```
RESOLVE INDOUBT(CICSA) ACTION(COMMIT) NID(CICSA.ABCDEF0123456789)
```

```
RESOLVE INDOUBT(CICSA) ACTION(BACKOUT) NID(*)
```

The *unit-of-recovery-id* values can be obtained from the DISPLAY THREAD command, see DISPLAY THREAD on page 148.

---

## RVERIFY SECURITY

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use 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.

**Synonym:** REV SEC

```
▶ RVERIFY SECURITY (userid) ▶
```

### Keyword and parameter descriptions

*userid* You must specify one or more user IDs. Each user ID specified is signed off and signed back on again the next time that a request is issued on behalf of that user that requires security checking.

**Note:** REVERIFY SECURITY is another synonym for RVERIFY SECURITY.

## START CHANNEL

---

### START CHANNEL

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use START CHANNEL to start a channel.

| This command can be issued to a channel of any type (except CLNTCONN channels). If, however, it is  
| issued to a receiver (RCVR) or server-connection (SVRCONN) channel, the only action is to enable the  
| channel, not to start it.

#### Notes:

1. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10
2. On MVS/ESA, the command fails if the channel initiator has not been started.

**Synonym:** STA CHL

▶—START CHANNEL(*channel-name*)————▶

### Keyword and parameter descriptions

(*channel-name*)

The name of the channel definition to be started. This is required. The name must be that of an existing channel defined on this queue manager.

## START CHINIT

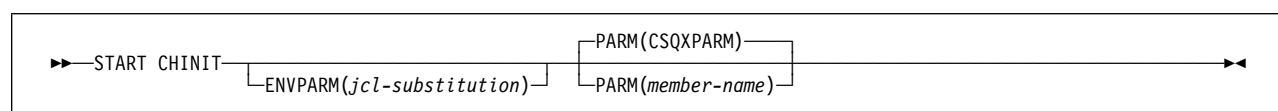
Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√		√	√

Use START CHINIT to start a channel initiator.

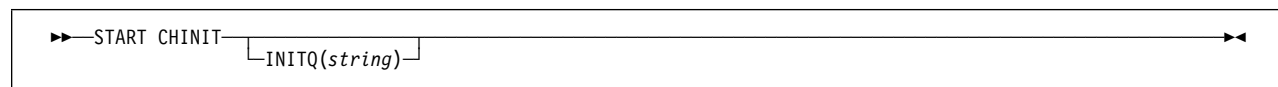
**Note:** On MVS/ESA, this is valid only for channels used for distributed queuing without CICS.

**Synonym:** STA CHI

### MQSeries for MVS/ESA



### MQSeries on other platforms



## Keyword and parameter descriptions

### 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=120')`.

This parameter is valid only on MVS/ESA.

### 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.

This must not be specified on MVS/ESA (the initiation queue on MVS/ESA is always `SYSTEM.CHANNEL.INITQ`). On other platforms, you can specify which initiation queue to use; if you do not specify this, `SYSTEM.CHANNEL.INITQ` is used.

### PARM(*member-name*)

The load module that contains the channel initiator initialization parameters. *member-name* is the name of a load module provided by the installation. The default is `CSQXPARM`, which is provided by MQSeries.

This keyword is valid only on MVS/ESA.

## START CMDSERV

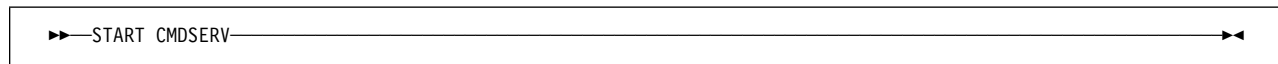
---

### START CMDSERV

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use START CMDSERV to initialize the command server.

**Synonym:** STA CS



### Usage notes

START CMDSERV starts the command server and allows it to process commands in the system-command input queue (SYSTEM.COMMAND.INPUT).

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.

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 immediately.

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.

## START LISTENER

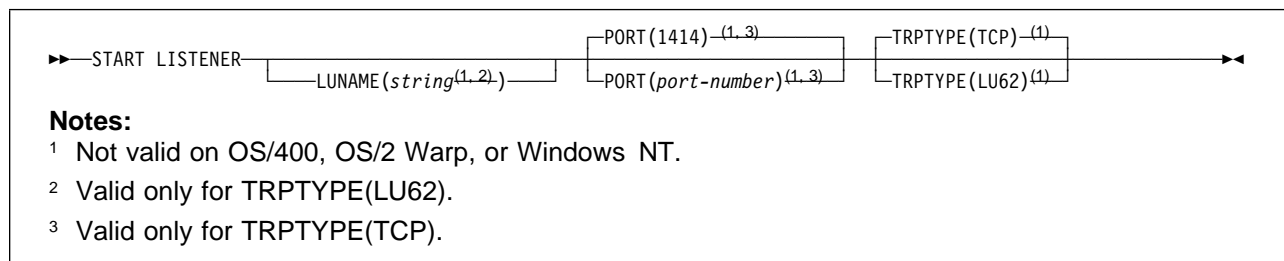
Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√	√	√			√

Use START LISTENER to start a channel listener.

### Notes:

1. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS.
2. On OS/400, OS/2 Warp, and Windows NT this command is valid only for channels for which the transmission protocol (TRPTYPE) is TCP/IP.
3. On MVS/ESA, the command fails if the channel initiator has not been started.

**Synonym:** STA LSTR



## Keyword and parameter descriptions

### LUNAME(*string*)

The symbolic destination name for the logical unit as specified in the APPC side information data set. (This 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 transmission protocol (TRPTYPE) of LU 6.2. A START LISTENER command which specifies TRPTYPE(LU62) must also specify the LUNAME parameter.

This parameter is not supported on OS/400, OS/2 Warp, or Windows NT.

### PORT(*port-number*)

Port number for TCP/IP. This is valid only if the transmission protocol (TRPTYPE) is TCP/IP. If it is not specified in this case the value is taken from the defaults file if available; otherwise 1414 is assumed.

This parameter is not supported on OS/400, OS/2 Warp, or Windows NT.

**Note:** On MVS/ESA, a defaults file is not supported.

### TRPTYPE

Transmission protocol to be used. This is optional.

**TCP** TCP/IP. This is the default if TRPTYPE is not specified.

**LU62** SNA LU 6.2.

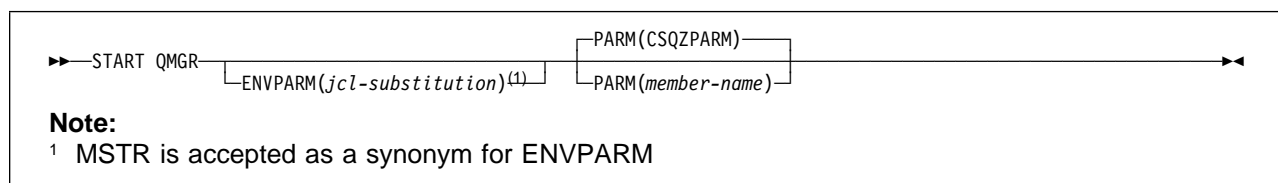
This parameter is not supported on OS/400, OS/2 Warp, or Windows NT.

## START QMGR

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use START QMGR to initialize the queue manager. When the operation has been completed, the queue manager is active and available to CICS, IMS, batch, and TSO applications.

**Synonym:** STA QMGR



## Keyword and parameter descriptions

These are optional.

### ENVPARAM(*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 ENVPARAM('HLQ=CSQ,VER=120').

MSTR is accepted as a synonym for ENVPARAM

### 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 MQSeries.

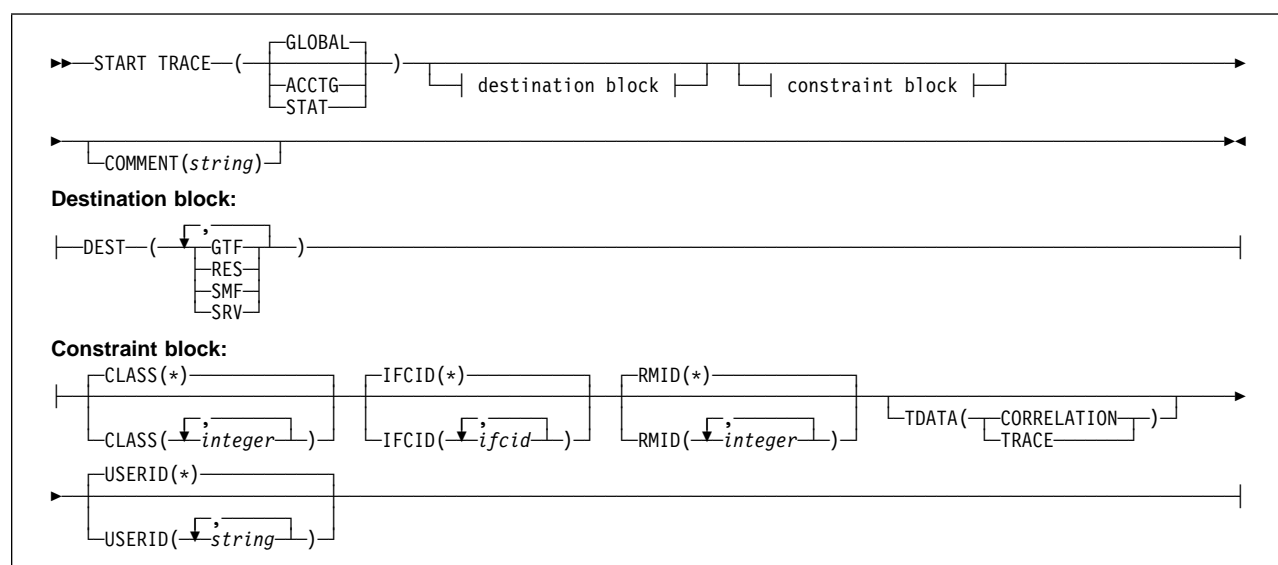


## START TRACE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use START TRACE to start traces. 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.

**Synonym:** STA TRACE



## Keyword and parameter descriptions

If you do not specify a trace type to be started, the default (GLOBAL) trace is started. The types are:

**ACCTG** Collects accounting data that can be used to charge your customers for their use of your queue manager. The synonym is A.

**Note:** Accounting data can be lost if the accounting trace is started or stopped while applications are running. See the *MQSeries for MVS/ESA System Management Guide* for information about the conditions that must be satisfied for successful collection of accounting data.

**GLOBAL** This includes data from the entire queue manager. The synonym is G.

**STAT** Collects statistical data broadcast by various components of MQSeries, at time intervals that can be chosen during installation. The synonym is S.

### 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.

## START TRACE

### 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 MVS 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 RMID 231.

**SMF** The System Management Facility (SMF). If used, the SMF must be functioning before the START TRACE command is issued. The SMF record numbers reserved for use by MQSeries are 115 and 116.

**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 will be produced at the MVS console when you issue the START TRACE command.

Allowed values, and the default value, depend on the type of trace started, as shown in the following table:

Type	GTF	RES	SMF	SRV
GLOBAL	Allowed	Default	No	Allowed
STAT	No	No	Default	Allowed
ACCTG	Allowed	No	Default	Allowed

**Constraint block:** The constraint block places optional constraints on the kinds of data collected by the trace. The allowed constraints depend on the type of trace started, as shown in the following table:

Type	CLASS	IFCID	RMID	USERID
GLOBAL	Allowed	Allowed	Allowed	Allowed
STAT	Allowed	No	No	No
ACCTG	Allowed	No	No	No

**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.

(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.

Class	IFCID	Description
		Global trace
01	0000	Reserved for IBM service
02	0018	User parameter error detected in a control block
03	0016	User parameter error detected on entry to MQI
	0017	User parameter error detected on exit from MQI
	0018	User parameter error detected in a control block
04	Various	Reserved for IBM service
		Statistics trace
01	0001	Subsystem statistics
	0002	Queue manager statistics
		Accounting trace
01	0003	The CPU time spent processing MQI calls and a count of MQPUT and MQGET calls

**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 or ACCTG traces.

(\*) Starts a trace for all resource managers. This is the default.

(integer) The identifying number of any resource manager in Table 7. You can use up to 8 of the allowed resource manager identifiers; do not use the same one twice.

If the list of RMIDs includes 231, the tracing for this resource manager is not started if one of the following is true:

- TRACE(STAT) or TRACE(ACCTG) is specified
- The list of destinations does not include RES
- This list of classes does not include 01 or 04

Also, comments will be truncated to 120 characters.

If tracing for RMID 231 is started, it stops if the channel initiator is stopped.

RMID	Resource manager
1	Initialization procedures
2	Agent services management
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

## START TRACE

RMID	Resource manager
24	Message generator
26	Instrumentation accounting and statistics
148	Connection manager
199	Functional recovery
200	Security management
201	Data management
211	Lock management
212	Message management
213	Command server
215	Buffer management
231	Channel Initiator
242	MQSeries-IMS bridge

**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 or ACCTG 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.

## STOP CHANNEL

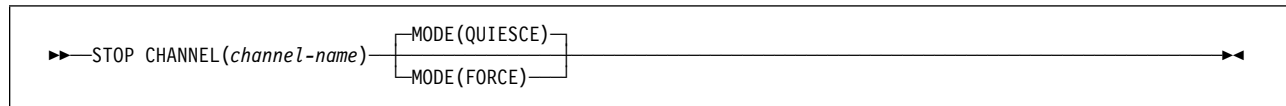
Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use STOP CHANNEL to stop a channel. This command can be issued to a channel of any type (except CLNTCONN channels).

### Notes:

1. On MVS/ESA, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 10.
2. On MVS/ESA, the command fails if the channel initiator has not been started.
3. You need to issue a START CHANNEL command to restart the channel, it will not restart automatically. See the *MQSeries Intercommunication* manual for information about restarting stopped channels.

**Synonym:** STOP CHL



## Keyword and parameter descriptions

*(channel-name)*

The name of the channel to be stopped. This is required.

**MODE** Specifies whether the current batch is allowed to finish in a controlled manner. This parameter is optional.

**QUIESCE** Allows the current batch to finish processing, except on MVS/ESA where 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:

- The next batch to start
- The next heartbeat (if heartbeats are being used)

before it stops.

For server-connection channels, allows the current connection to end.

This is the default.

**FORCE** Terminates transmission of any current batch. This is likely to result in in-doubt situations.

For server-connection channels, breaks the current connection, returning MQRC\_CONNECTION\_BROKEN.

## STOP CHINIT

---

### STOP CHINIT

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use STOP CHINIT to stop a channel initiator.

**Note:** This command is valid only for channels used for distributed queuing without CICS.

**Synonym:** STOP CHI

▶▶—STOP CHINIT—◀◀

### Usage notes

Any channels that are running are allowed to quiesce, with current batches allowed to finish processing.

- | If some of the channels are receiver or requester channels which are running but not active, then a stop request issued to either the receiver's or sender's channel initiator will cause it to stop immediately.
- | However, if messages are flowing, then the channel initiator waits for the current batch of messages to complete before it stops.

If you need to force the channel initiator to stop, use the STOP QMGR MODE(FORCE) command (or the MVS CANCEL command).

---

## STOP CMDSERV

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use STOP CMDSERV to stop the command server.

**Synonym:** STOP CS

►—STOP CMDSERV—◄

### Usage notes

STOP CMDSERV stops the command server from processing commands in the system-command input queue (SYSTEM.COMMAND.INPUT).

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.

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.

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.

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 LISTENER

---

### STOP LISTENER

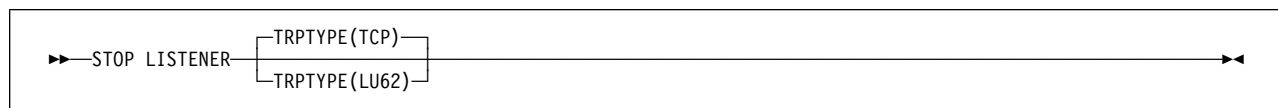
Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use STOP LISTENER to stop a channel listener.

#### Notes:

1. This is valid only for channels used for distributed queuing without CICS.
2. The command fails if the channel initiator has not been started.

**Synonym:** STOP LSTR



### Keyword and parameter descriptions

#### TRPTYPE

Transmission protocol used. This is optional.

**TCP** TCP/IP. This is the default if TRPTYPE is not specified.

**LU62** SNA LU 6.2.

On MVS/ESA, only one listener for each protocol is allowed for a given queue manager, and therefore no further parameters are needed to identify which listener is to be stopped.

The listener stops in quiesce mode (it disregards any further requests).



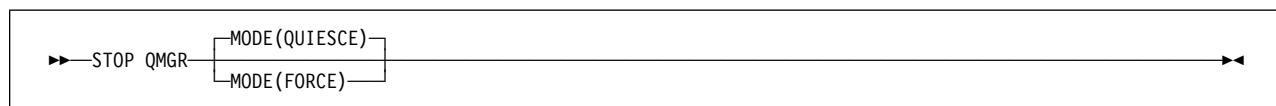
---

## STOP QMGR

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use STOP QMGR to stop the queue manager.

**Synonym:** There is no synonym for this command.



## Keyword and parameter descriptions

The parameters are optional.

**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 THREAD command, and can cancel remaining connections using MVS commands.

**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 it will be necessary to issue the MVS CANCEL command to terminate.

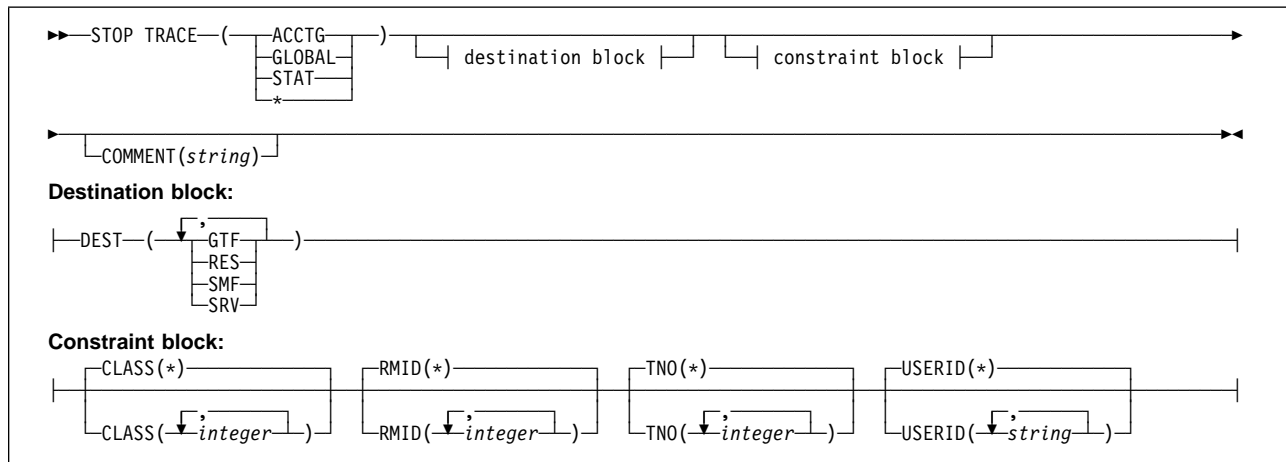
## STOP TRACE

### STOP TRACE

Digital OpenVMS	MVS/ESA	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use STOP TRACE to stop tracing.

**Synonym:** There is no synonym for this command.



## Keyword and parameter descriptions

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. See the *MQSeries for MVS/ESA System Management Guide* for information about the conditions that must be satisfied for successful collection of accounting data.

**GLOBAL** Service data from the entire MQSeries subsystem (the synonym is G)

**STAT** Statistical data (the synonym is S)

**\*** All active traces

For further descriptions of each type, see START TRACE on page 167.

**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 of the STOP TRACE 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

See START TRACE on page 167 for a list of allowed destinations for each trace type.

## Constraint block

### **CLASS**(*integer*)

Limits the action of the STOP TRACE 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 action of the STOP TRACE 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 or ACCTG trace type.

If the list of RMIIDs includes 231, the tracing for this resource manager is left unchanged if one of the following is true:

- TRACE(GLOBAL) or TRACE(\*) is not specified
- The list of destinations does not include RES
- This list of classes does not include 01 or 04

Also, comments will be truncated to 120 characters.

The default is RMID(\*), which does not limit the command.

### **TNO**(*integer*)

Limits the action of the STOP TRACE to particular traces, identified by their trace numbers (1 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 command.

### **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 STAT.

The default is USERID(\*), which does not limit the command.

**STOP TRACE**

## Appendix A. Command summary

The following tables show how the various command formats in MQSeries relate to each other. The command formats available are:

- Programmable command format (PCF) commands
- MQSeries (MQSC) commands
- MQSeries for OS/400 CL commands
- Control commands for MQSeries products on distributed platforms, that is, MQSeries on UNIX systems, MQSeries for Digital OpenVMS, MQSeries for Tandem NonStop Kernel, MQSeries for OS/2 Warp, and MQSeries for Windows NT

### Notes:

1. The PCF commands are not supported on MVS/ESA.
2. Unless otherwise specified, the MQSC commands are supported on all platforms.
3. An empty cell indicates that there is no equivalent command in the specified format.
4. In MQSeries for Tandem NonStop Kernel control commands are entered in lowercase.

Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Change Queue Manager attributes	Change Queue Manager	ALTER QMGR DEFINE MAXSMSGS (See note 1)	CHGMQM	
Display Queue Manager attributes	Inquire Queue Manager	DISPLAY QMGR DISPLAY MAXSMSGS (See note 1)	DSPMQM	
Connect a Queue Manager			CCTMQM	
Create a Queue Manager			CRTMQM	CRTMQM
Delete a Queue Manager			DLTMQM	DLTMQM
Disconnect a Queue Manager			DSCMQM	
Stop a Queue Manager		STOP QMGR (See note 1)	ENDMQM	ENDMQM
Ping a Queue Manager	Ping Queue Manager	PING QMGR (See note 2)		
Start a Queue Manager		START QMGR (See note 1)	STRMQM	STRMQM
Add a Queue Manager to Windows NT Service Control Manager				SCMMQM (See note 3)
Start an MQSeries trial period				SETMQTRY (See note 4)

## Command summary

<i>Table 8 (Page 2 of 2). Commands for queue manager administration</i>				
Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Enroll an MQSeries production licence				SETMQPRD (See note 4)
<b>Notes:</b>				
1. Applies on MVS/ESA only 2. Does not apply on MVS/ESA 3. Applies on Windows NT only 4. Applies on V5.0 of MQSeries for AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT				

<i>Table 9 (Page 1 of 2). Commands for queue administration</i>				
Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Change queue attributes	Change Queue	ALTER QALIAS ALTER QLOCAL ALTER QMODEL ALTER QREMOTE	CHGMQMQ	
Clear a queue	Clear Queue	CLEAR QLOCAL (See note 1)  The following sequence: DELETE QLOCAL(x), DEFINE QLOCAL(x)  or the following sequence: DEFINE QLOCAL(y) LIKE(x), DELETE QLOCAL(x), DEFINE QLOCAL(x) LIKE(y), DELETE QLOCAL(y)	CLRMQMQ	
Copy a queue definition	Copy Queue	DEFINE QALIAS(x) LIKE(y) DEFINE QLOCAL(x) LIKE(y) DEFINE QMODEL(x) LIKE(y) DEFINE QREMOTE(x) LIKE(y)	CPYMQMQ	
Create a queue	Create Queue	DEFINE QALIAS DEFINE QLOCAL DEFINE QMODEL DEFINE QREMOTE	CRTMQMQ	
Delete a queue	Delete Queue	DELETE QALIAS DELETE QLOCAL DELETE QMODEL DELETE QREMOTE	DLTMQMQ	
Display queue attributes	Inquire Queue	DISPLAY QUEUE DISPLAY QALIAS (See note 2) DISPLAY QLOCAL (See note 2) DISPLAY QMODEL (See note 2) DISPLAY QREMOTE (See note 2)	DSPMQMQ	
Display queue names	Inquire Queue Names	DISPLAY QUEUE	WRKMQMQ	
Work with a queue			WRKMQMQ	
Work with messages			WRKMQMMSG	

*Table 9 (Page 2 of 2). Commands for queue administration*

Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Reset queue statistics	Reset Queue Statistics (See note 3)			
<b>Notes:</b>				
1. Does not apply on MVS/ESA				
2. Applies on AIX, HP-UX, MVS/ESA, OS/2, Sun Solaris, and Windows NT only				
3. Does not apply on Tandem NSK				

*Table 10. Commands for process definition administration*

Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Change process attributes	Change Process	ALTER PROCESS	CHGMQMPCRC	
Copy a process	Copy Process	DEFINE PROCESS(x) LIKE(y)	CPYMQMPCRC	
Create a process	Create Process	DEFINE PROCESS	CRTMQMPCRC	
Delete a process	Delete Process	DELETE PROCESS	DLTMQMPCRC	
Display process attributes	Inquire Process	DISPLAY PROCESS	DSPMQMPCRC	
Display process names	Inquire Process Names	DISPLAY PROCESS	WRKMQMPCRC	
Work with a process			WRKMQMPCRC	

*Table 11. Commands for namelist administration (MVS/ESA only)*

Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Alter a namelist		ALTER NAMELIST		
Copy a namelist		DEFINE NAMELIST(x) LIKE(y)		
Define a namelist		DEFINE NAMELIST		
Delete a namelist		DELETE NAMELIST		
Display a namelist		DISPLAY NAMELIST		

*Table 12 (Page 1 of 2). Commands for channel administration*

Operation	PCF	MQSC (See note 1)	OS/400 CL	Commands for distributed platforms
Change channel attributes	Change Channel	ALTER CHANNEL	CHGMQMCHL	
Copy channel attributes	Copy Channel	DEFINE CHANNEL (x) LIKE (y)	CPYMQMCHL	
Create a channel	Create Channel	DEFINE CHANNEL	CRTMQMCHL	
Delete a channel	Delete Channel	DELETE CHANNEL	DLTMQMCHL	
Display a channel	Inquire Channel	DISPLAY CHANNEL	DSPMQMCHL	
Display channel names	Inquire Channel Names	DISPLAY CHANNEL	WRKMQMCHL	

## Command summary

Table 12 (Page 2 of 2). Commands for channel administration

Operation	PCF	MQSC (See note 1)	OS/400 CL	Commands for distributed platforms
Display channel status	Inquire Channel Status	DISPLAY CHSTATUS	WRKMQMCHST	
Display distributed queuing		DISPLAY DQM (See note 2)		
Ping a channel	Ping Channel	PING CHANNEL	PNGMQMCHL	
Reset a channel	Reset Channel	RESET CHANNEL	RSTMQMCHL	
Resolve a channel	Resolve Channel	RESOLVE CHANNEL	RSVMQMCHL	
Start a channel	Start Channel	START CHANNEL	STRMQMCHL	RUNMQCHL
Start a channel initiator	Start Channel Initiator	START CHINIT (See note 2)	STRMQMCHLI	RUNMQCHI
Start a channel listener	Start Channel Listener	START LISTENER (See note 3)	STRMQMLSR	RUNMQLSR (See note 4)
Stop a channel	Stop Channel	STOP CHANNEL	ENDMQMCHL	
Stop a channel initiator		STOP CHINIT (See note 2)		
Stop a channel listener		STOP LISTENER (See note 2)		ENDMQLSR (See note 5)
Work with channels			WRKMQMCHL	
Work with channel status			WRKMQMCHST	
<b>Notes:</b>				
1. Does not apply on MVS/ESA if you are using CICS for distributed queuing				
2. Applies on MVS/ESA only				
3. Does not apply on UNIX systems, Digital OpenVMS, or Tandem NSK				
4. Applies on OS/2, Windows NT, Digital OVMS, and Tandem NSK only				
5. Applies on OS/2 and Windows NT only				
In MQSeries for Tandem NonStop Kernel, use TS/MP or the control command <b>runmqlsr</b> to start TCP/IP channel listeners.				

Table 13. Commands for security administration

Operation	PCF	MQSC (See note 1)	OS/400 CL	Commands for distributed platforms
Display object authority			DSPMQMAUT	DSPMQAUT
Grant object authority			GRTMQMAUT	SETMQAUT
Revoke object authority			RVKMQMAUT	SETMQAUT
Alter security options		ALTER SECURITY		
Display security settings		DISPLAY SECURITY		DSPMQAUT
Refresh security		REFRESH SECURITY		
Set a reverification flag		RVERIFY SECURITY		
<b>Note:</b>				
1. Applies on MVS/ESA only				

Table 14 (Page 1 of 2). Commands for system-dependent function

Operation	PCF	MQSC (see note 1)	OS/400 CL	Commands for distributed platforms
Alter trace parameters		ALTER TRACE		



Table 14 (Page 2 of 2). Commands for system-dependent function

Operation	PCF	MQSC (see note 1)	OS/400 CL	Commands for distributed platforms
Display trace activity		DISPLAY TRACE		
Start a trace		START TRACE	TRCMQM	STRMQTRC (See note 2)
Stop a trace		STOP TRACE	TRCMQM	ENDMQTRC (See note 2)
Archive a log		ARCHIVE LOG		
Define a buffer pool		DEFINE BUFFPOOL		
Define a page set		DEFINE PSID		
Display page set information		DISPLAY USAGE		
Alter a storage class		ALTER STGCLASS		
Define a storage class		DEFINE STGCLASS		
Delete a storage class		DELETE STGCLASS		
Display storage class information		DISPLAY STGCLASS		
Display a thread		DISPLAY THREAD		
Recover a bootstrap data set		RECOVER BSDS		
Resolve in-doubt threads		RESOLVE INDOUBT		
Display the command server		DISPLAY CMDSERV	DSPMQMCSVR	DSPMQCSV
Start the command server		START CMDSERV	STRMQMCSVR	STRMQCSV
Stop the command server		STOP CMDSERV	ENDMQMCSVR	ENDMQCSV
Reset an IMS transaction pipe		RESET TPIPE		
Display an object name			DSPMQMOBJN	
Start a service job			STRMQMSRV	
End a service job			ENDMQMSRV	
Start the administrator			STRMQMADM	
Record an object image			RCDMQMIMG	RCDMQIMG (See note 3)
Recreate an object			RCRMQMOBJ	RCRMQOBJ (See note 3)
Display MQSeries formatted trace output				DSPMQTRC (See note 4)
Dump contents of MQSeries log				DMPMQLOG (See note 5)

**Notes:**

1. Applies on MVS/ESA only
2. Does not apply on AIX
3. Does not apply on Tandem NSK
4. Applies on AT&T, HP-UX, SINIX and DC/OSx, SunOS, Sun Solaris, and Tandem NSK
5. Applies on V5.0 of MQSeries for AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT

In MQSeries for Tandem NonStop Kernel, as an alternative to the control commands dspmqcsv, strmqcsv, and endmqcsv, you may use PATHCOM commands.

## Command summary

<i>Table 15. Other control commands in MQSeries for Tandem NonStop Kernel</i>	
<b>Operation</b>	<b>Commands</b>
Alter queue volume	altmqfls
Perform housekeeping on a queue manager	cleanqm
Convert V1.5.1 queues and channels to V2.2	cnv1520
Convert V1.5.1 messages to V2.2	cnvmsg
Convert client channel definition table	cnvclchl
Install MQSeries for Tandem NonStop Kernel	instmqm
Run dead-letter queue handler	runmqdlq
<b>Note:</b> As an alternative to the control command <b>runmqtrm</b> , you may use PATHCOM commands. There are no MQSC or PCF equivalents of commands in this group.	

---

## Appendix B. How to issue MQSC commands on Digital OpenVMS

This appendix tells you how to issue MQSC commands from a Digital OpenVMS system in which MQSeries is installed. Specifically, it tells you how to issue MQSC commands locally, that is, the commands are processed by a queue manager on the machine on which you issue them. Remote administration, where you issue commands to be processed by a queue manager on a different machine, is described in the *MQSeries for Digital OpenVMS System Management Guide* manual.

---

### Using the runmqsc command

To issue MQSC commands, use the **runmqsc** command from the DCL prompt. This command takes its input from the system input device (SYS\$INPUT) and sends its output to the system output device (SYS\$OUTPUT). For more information about the **runmqsc** command, see the *MQSeries for Digital OpenVMS System Management Guide* manual.

### Issuing MQSC commands interactively

Provided that SYS\$INPUT is the keyboard and the SYS\$OUTPUT is the display, you can type in MQSC commands, one at a time, at the prompt. The results from the commands are then displayed on your screen.

For example, if you have a queue manager called QMAN1, you type:

```
runmqsc QMAN1
```

If you omit the queue manager name, the default queue manager is used. The queue manager responds with output similar to the following:

```
5697-270 (C) Copyright IBM Corp. 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
```

You can then type in your commands, as required, for example:

```
DEFINE QLOCAL(TEST) +
  REPLACE +
  DESCR('This is a test queue') +
  TRIGGER +
  INITQ(SYSTEM.SAMPLE.TRIGGER)
```

The plus sign indicates that the command is continued on the next line. When you press Enter, the queue manager confirms that the command has been carried out:

```
1: DEFINE QLOCAL(TEST) +
:   REPLACE +
:   DESCR('This is a test queue') +
:   TRIGGER +
:   INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006: MQSeries queue created.
```

## Issuing Digital OpenVMS commands

You can then continue to enter additional commands, as required. When you have finished, type the end-of-file character, which in Digital OpenVMS is CTRL+Z. The queue manager then displays a summary report, for example:

```
3 MQSC commands read.  
0 commands have a syntax error.  
0 commands cannot be processed.
```

## Issuing MQSC commands from a command file

You can put a sequence of MQSC commands in an ASCII text file, called a command file, using a standard editor. The following **runmqsc** command redirects both SYS\$INPUT and SYS\$OUTPUT. It takes input from the command file `commnds.in`, processes the commands contained in the file, and sends a report to the file `report.out`. The commands are processed by the default queue manager.

```
runmqsc < commnds.in > report.out
```

The command file `commnds.in` contains the following:

```
* This is a sample MQSC command file  
  
* Step 1 - Create the queue  
DEFINE QLOCAL(TEST) +  
  REPLACE +  
  DESCR('This is a test queue') +  
  TRIGGER +  
  INITQ(SYSTEM.SAMPLE.TRIGGER)  
  
* Step 2 - Display selected queue attributes  
DISPLAY Q(TEST) +  
  DESCR +  
  TRIGGER +  
  INITQ +  
  GET +  
  PUT  
  
* Step 3 - Delete the queue  
DELETE QLOCAL (TEST)
```

Figure 1. Example command input file for Digital OpenVMS

The following report is sent to report.out, the output file:

```

5697-270 (C) Copyright IBM Corp. 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
  : * This is a sample MQSC command file
  :
  : * Step 1 - Create the queue
1  : DEFINE QLOCAL(TEST) +
  :       REPLACE +
  :       DESCR('This is a test queue') +
  :       TRIGGER +
  :       INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006 MQSeries queue created.
  :
  : * Step 2 - Display selected queue attributes
2  : DISPLAY Q(TEST) +
  :       DESCR +
  :       TRIGGER +
  :       INITQ +
  :       GET +
  :       PUT
  :
AMQ8409 Display Queue details.
DESCR(This is a test queue)
INITQ(SYSTEM.SAMPLE.TRIGGER)
QUEUE(TEST)
GET(ENABLED)
PUT(ENABLED)
TRIGGER
  :
  : * Step 3 - Delete the test queue
3  : DELETE QL(TEST)
AMQ8007 MQSeries queue deleted.
  :
  : * End of this sample
3 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.

```

Figure 2. Example report file from Digital OpenVMS

## Error messages

If the command fails, there might be additional information about the problem in the error log.

### Verifying MQSC commands

To verify the commands in an input file, use the **runmqsc** command with the v flag:

```
runmqsc -v < commnds.in > report.out
```

This command verifies the MQSC commands in the input file and writes a report to the report file. The report contains the same information as is obtained when the commands are actually run, except that it does not contain any messages indicating the outcome of each command.

---

## Appendix C. How to issue MQSC commands on MVS/ESA

The MQSeries for MVS/ESA commands in this book can be issued from the following sources (except where specifically indicated otherwise):

- The MVS/ESA console (or equivalent, such as SDSF).  
**Note:** The DEFINE CHANNEL command for sender, server, and requester channels might require too many mandatory parameters to be issued using SDSF. This command has to be issued by one of the other methods shown below.
- The initialization input data set, CSQINP1, to be processed before the restart phase of queue manager initialization.
- The initialization input data set, CSQINP2, to be processed after the restart phase of queue manager initialization.
- The initialization input data set, CSQINPX, to be processed after the restart phase of channel initiator initialization.
- The supplied batch utility (CSQUTIL) processing a list of commands in a sequential data set or member of a partitioned data set.
- The MVS Master Get Console Routine, MGCR or MGCRE (SVC34).
- A suitably authorized application, by sending a command as a message to the system-command input queue. This can be either:
  - A batch region program
  - A CICS application
  - An IMS application
  - A TSO application

When entering commands from the MVS console, use the command prefix string (CPF) defined for your queue manager. When entering commands by any other means, the CPF must not be present.

Much of the functionality of these commands is available in a user-friendly way from the operations and control panels, described in the *MQSeries for MVS/ESA System Management Guide*.

Changes made to the resource definitions of a queue manager using the commands (directly or indirectly) are preserved across restarts of the queue manager.

See the *MQSeries for MVS/ESA System Management Guide* for more information about issuing commands on MQSeries for MVS/ESA.

### Directing the command to the correct queue manager

The method you use to enter a command determines how you indicate the destination queue manager for it.

- If you issue the command through the console, use the CPF of the destination queue manager.
- If you issue the command through the utility program, (CSQUTIL), specify the destination queue manager with the TGTQMGR keyword; you also need to specify the queue manager to which you will connect via the EXEC PARM parameter of your JCL.
- If you issue the command through an administration program, the command is directed to the queue manager that owns the system-command input queue onto which the command message is put.



---

## Appendix D. How to issue MQSC commands on OS/2 Warp

This appendix tells you how to issue MQSC commands from an OS/2 Warp system in which MQSeries is installed. Specifically, it tells you how to issue MQSC commands locally, that is, the commands are processed by a queue manager on the machine on which you issue them. Remote administration, where you issue commands to be processed by a queue manager on a different machine, is described in the *MQSeries System Administration* manual.

---

### Using the runmqsc command

To issue MQSC commands from an OS/2 Warp system, use the **runmqsc** command from an OS/2 window. This command takes its input from standard in and sends its output to standard out. For more information about the **runmqsc** command, see the *MQSeries System Administration* manual.

### Issuing MQSC commands interactively

Provided that standard in is the keyboard and standard out is an OS/2 window, you can type in MQSC commands, one at a time, in an OS/2 shell window. The results from the commands are then displayed in the window.

For example, if you have a queue manager called QMAN1, you type:

```
runmqsc QMAN1
```

The queue manager QMAN1 responds with:

```
5621-390 (C) Copyright IBM Corp. 1995, 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
```

You can then type in your commands, as required, for example:

```
DEFINE QLOCAL(TEST) +
REPLACE +
DESCR('This is a test queue') +
TRIGGER +
INITQ(SYSTEM.SAMPLE.TRIGGER)
```

The plus sign indicates that the command is continued on the next line. When you press Enter, the queue manager confirms that the command has been carried out:

```
1: DEFINE QLOCAL(TEST) +
:   REPLACE +
:   DESCR('This is a test queue') +
:   TRIGGER +
:   INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006: MQSeries queue created.
```

If you accidentally add a + to the end of a line when you want to end a command, use the ; character to end the command.

You can then continue to enter additional commands, as required. When you have finished, type END, then press Enter. (Alternatively, you can type the end-of-file

## Issuing OS/2 commands

character, which in OS/2 Warp is Ctrl+Z, then press Enter.) The queue manager then displays a summary report, for example:

```
3 MQSC commands read.  
0 commands have a syntax error.  
0 commands cannot be processed.
```

### Getting help

When you are issuing commands interactively on OS/2 Warp, you can get help by entering command ? (where command is the name of the command you are interested in). MQSeries displays the syntax of the command.

If you enter an invalid command, MQSeries displays the syntax of the command if it can determine what the command was or, alternatively, displays a list of commands for you to select from.

## Issuing MQSC commands from a command file

You can put a sequence of MQSC commands in an ASCII text file, called a command file, using a standard editor. The following **runmqsc** command redirects both standard in and standard out. It takes input from the command file `commnds.in`, processes the commands contained in the file, and sends a report to the file `report.out`. The commands are processed by the default queue manager.

```
runmqsc < commnds.in > report.out
```

The command file `commnds.in` contains the following:

```
* This is a sample MQSC command file  
  
* Step 1 - Create the queue  
DEFINE QLOCAL(TEST) +  
  REPLACE +  
  DESCR('This is a test queue') +  
  TRIGGER +  
  INITQ(SYSTEM.SAMPLE.TRIGGER)  
  
* Step 2 - Display selected queue attributes  
DISPLAY Q(TEST) +  
  DESCR +  
  TRIGGER +  
  INITQ +  
  GET +  
  PUT  
  
* Step 3 - Delete the queue  
DELETE QLOCAL (TEST)
```

Figure 3. Example command input file for OS/2 Warp

The following report is sent to report.out, the output file:

```

5621-390 (C) Copyright IBM Corp. 1995, 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
  : * This is a sample MQSC command file
  :
  : * Step 1 - Create the queue
1  : DEFINE QLOCAL(TEST) +
  :       REPLACE +
  :       DESCR('This is a test queue') +
  :       TRIGGER +
  :       INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006 MQSeries queue created.
  :
  : * Step 2 - Display selected queue attributes
2  : DISPLAY Q(TEST) +
  :       DESCR +
  :       TRIGGER +
  :       INITQ +
  :       GET +
  :       PUT
  :
AMQ8409 Display Queue details.
  DESCR(This is a test queue)      INITQ(SYSTEM.SAMPLE.TRIGGER)
  QUEUE(TEST)                     GET(ENABLED)
  PUT(ENABLED)                    TRIGGER
  :
  : * Step 3 - Delete the test queue
3  : DELETE QL(TEST)
AMQ8007 MQSeries queue deleted.
  :
  : * End of this sample
3 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.

```

Figure 4. Example report file from OS/2 Warp

## Error messages

If the command fails, there might be additional information about the problem in the error log.

## Verifying MQSC commands

To verify the commands in an input file, use the **runmqsc** command with the v flag:

```
runmqsc /v < commnds.in > report.out
```

This command verifies the MQSC commands in the input file and writes a report to the report file. The report contains the same information as is obtained when the commands are actually run, except that it does not contain any messages indicating the outcome of each command.

## Issuing OS/2 commands

---

## Appendix E. How to issue MQSC commands on OS/400

To issue MQSC commands on OS/400, create a list of MQSC commands in a text file. The source physical file member maximum line length for this file is 80 characters. You then have the option of running or verifying the commands in the file.

To run the commands in the file, use the STRMQMMQSC command in the default mode. This processes the commands in the file, and writes a report to the printer spool file.

---

### Example OS/400 MQSeries command file and report

This example creates a local queue, called TEST, displays its attributes, and then deletes it:

```
* This is a sample MQSeries file to show report format

* Step 1 - Create a queue
DEFINE QLOCAL (TEST) REPLACE DESCR('A test queue')    +
TRIGGER INITQ(system.sample.trigger)

* Step 2 - Display selected queue attributes
DISPLAY Q(TEST) DESCR  +
TRIGGER TRIGTYPE TRIGDPH GET PUT INITQ

* Step 3 - Delete the test queue
DELETE QLOCAL(TEST)

* End of this sample
```

Figure 5. Example command input file for OS/400

The report generated contains 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

## Issuing OS/400 commands

For example, the report generated by running the commands in this example is as follows:

```
MQM/400      STRMQMMQSC: HHLIB/MQSC(SAMPLE)
Starting MQSeries Commands.
: * This is a sample MQSeries file to show report format
:
: * Step 1 - Create a queue
1 : DEF QL(TEST) REPLACE DESCR('A test queue') +
:   TRIGGER INITQ(system.sample.trigger)
AMQ8006 MQM queue created.
:
: * Step 2 - Display selected queue attributes
2 : DIS Q(TEST) DESCR +
:   TRIGGER TRIGTYPE TRIGDPH GET PUT INITQ
AMQ8409 Display Queue details.
DESCR(This is a test queue)
INITQ(SYSTEM.SAMPLE.TRIGGER)
QUEUE(TEST)
GET(ENABLED)
PUT(ENABLED)
TRIGGER
TRIGTYPE(FIRST)
TRIGDPH(1)
:
: * Step 3 - Delete the test queue
3 : DELETE QL(TEST)
AMQ8007 MQM queue deleted.
:
: * End of this sample
3 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.
```

Figure 6. Example report file from OS/400

## Verifying MQSC commands

To verify the commands in an input file, use the STRMQMMQSC command with the OPTION keyword set to (\*VERIFY). This command verifies the MQSC commands in the input file and writes a report to the printer spool file. The report contains the same information that is obtained when the commands are actually run, except that it does not contain any messages indicating the outcome of each command.

## Appendix F. How to issue MQSC commands on Tandem NSK

This appendix tells you how to issue MQSC commands from a Tandem NSK system in which MQSeries is installed. Specifically, it tells you how to issue MQSC commands locally, that is, the commands are processed by a queue manager on the machine on which you issue them. Remote administration, where you issue commands to be processed by a queue manager on a different machine, is described in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

### Using the runmqsc command

To issue MQSC commands, use the **runmqsc** command from the command prompt. This command takes its input from the system input device and sends its output to the system output device. For more information about the **runmqsc** command, see the *MQSeries for Tandem NonStop Kernel System Management Guide*.

### Issuing MQSC commands interactively

Open a TACL session and enter runmqsc. You can enter MQSC commands, one at a time, at the prompt. The results from the commands are then displayed on your screen.

For example, if you have a queue manager called my.queue.manager, you type:

```
runmqsc my.queue.manager
```

If you omit the queue manager name, the default queue manager is used. The queue manager responds with output similar to the following:

```
5697-A17 (C) Copyright IBM Corp. 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
```

You can then type in your commands, as required, for example:

```
DEFINE QLOCAL(TEST) +
  REPLACE +
  DESCR('This is a test queue') +
  TRIGGER +
  INITQ(SYSTEM.SAMPLE.TRIGGER)
```

The plus sign indicates that the command is continued on the next line. When you press Enter, the queue manager confirms that the command has been carried out:

```
1: DEFINE QLOCAL(TEST) +
  :     REPLACE +
  :     DESCR('This is a test queue') +
  :     TRIGGER +
  :     INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006: MQSeries queue created.
```

## Issuing Tandem NSK commands

You can then continue to enter additional commands, as required. When you have finished, type the end-of-file character, which in Tandem NSK is CTRL+Y, or exit. The queue manager then displays a summary report, for example:

```
3 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.
```

## Issuing MQSC commands from a command file

You can put a sequence of MQSC commands in an ASCII text file, called a command file, using a standard editor. When you use the **runmqsc** command, use the TACL IN and OUT redirection operators, or the flags **-i** and **-o** on **runmqsc**. For example, the following command runs a sequence of commands contained in a text file called `commndin` and redirects the output to a report file called `commndou`:

```
runmqsc -i commndin -o commndou
```

The command file `commndin` contains the following:

```
* This is a sample MQSC command file

* Step 1 - Create the queue
DEFINE QLOCAL(TEST) +
  REPLACE +
  DESCR('This is a test queue') +
  TRIGGER +
  INITQ(SYSTEM.SAMPLE.TRIGGER)

* Step 2 - Display selected queue attributes
DISPLAY Q(TEST) +
  DESCR +
  TRIGGER +
  INITQ +
  GET +
  PUT

* Step 3 - Delete the queue
DELETE QLOCAL (TEST)
```

Figure 7. Example command input file for Tandem NSK



The following report is sent to commndou, the output file:

```

5697-A17 (C) Copyright IBM Corp. 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
  : * This is a sample MQSC command file
  :
  : * Step 1 - Create the queue
1  : DEFINE QLOCAL(TEST) +
  :         REPLACE +
  :         DESCR('This is a test queue') +
  :         TRIGGER +
  :         INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006 MQSeries queue created.
  :
  : * Step 2 - Display selected queue attributes
2  : DISPLAY Q(TEST) +
  :         DESCR +
  :         TRIGGER +
  :         INITQ +
  :         GET +
  :         PUT
  :
AMQ8409 Display Queue details.
DESCR(This is a test queue)
INITQ(SYSTEM.SAMPLE.TRIGGER)
QUEUE(TEST)
GET(ENABLED)
PUT(ENABLED)
TRIGGER
  :
  : * Step 3 - Delete the test queue
3  : DELETE QL(TEST)
AMQ8007 MQSeries queue deleted.
  :
  : * End of this sample
3 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.

```

Figure 8. Example report file from Tandem NSK

### Error messages

If the command fails, there might be additional information about the problem in the error log.

### Verifying MQSC commands

To verify the commands in an input file, use the **runmqsc** command with the **-v** flag:

```
runmqsc -i commndin -o commndou -v
```

This command verifies the MQSC commands in the input file and writes a report to the report file. The report contains the same information as is obtained when the commands are actually run, except that it does not contain any messages indicating the outcome of each command.

---

## Appendix G. How to issue MQSC commands on UNIX systems

This appendix tells you how to issue MQSC commands from a UNIX system in which MQSeries is installed. Specifically, it tells you how to issue MQSC commands locally, that is, the commands are processed by a queue manager on the machine on which you issue them. Remote administration, where you issue commands to be processed by a queue manager on a different machine, is described in the *MQSeries System Administration* manual.

---

### Using the runmqsc command

To issue MQSC commands, use the **runmqsc** command from the UNIX system shell. This command takes its input from stdin and sends its output to stdout. For more information about the **runmqsc** command, see the *MQSeries System Administration* manual.

### Issuing MQSC commands interactively

Provided that stdin is the keyboard and stdout is the shell window, you can type in MQSC commands, one at a time, in a shell window. The results from the commands are then displayed in the window.

For example, if you have a queue manager called QMAN1, you type:

```
runmqsc QMAN1
```

If you omit the queue manager name, the default queue manager is used. The queue manager responds with output similar to the following:

```
5765-514 (C) Copyright IBM Corp. 1993,1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
```

You can then type in your commands, as required, for example:

```
DEFINE QLOCAL(TEST) +
  REPLACE +
  DESCR('This is a test queue') +
  TRIGGER +
  INITQ(SYSTEM.SAMPLE.TRIGGER)
```

The plus sign indicates that the command is continued on the next line. When you press Enter, the queue manager confirms that the command has been carried out:

```
1: DEFINE QLOCAL(TEST) +
:   REPLACE +
:   DESCR('This is a test queue') +
:   TRIGGER +
:   INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006: MQSeries queue created.
```

If you accidentally add a + to the end of a line on AIX, HP-UX, or Sun Solaris when you want to end a command, use the ; character to end the command.

## Issuing UNIX system commands

You can then continue to enter additional commands, as required. When you have finished, type END, then press Enter (on AIX, HP-UX, and Sun Solaris), or type the end-of-file character, which in UNIX systems is CTRL+D. The queue manager then displays a summary report, for example:

```
3 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.
```

### Getting help

Man pages are provided for all the MQSC commands.

In addition, when you are issuing commands interactively on AIX, HP-UX, or Sun Solaris, you can get help by entering command ? (where command is the name of the command you are interested in). MQSeries displays the syntax of the command.

If you enter an invalid command, MQSeries displays the syntax of the command if it can determine what the command was or, alternatively displays a list of commands for you to select from.

## Issuing MQSC commands from a command file

You can put a sequence of MQSC commands in an ASCII text file, called a command file, using a standard editor. The following **runmqsc** command redirects both stdin and stdout. It takes input from the command file `commnds.in`, processes the commands contained in the file, and sends a report to the file `report.out`. The commands are processed by the default queue manager.

```
runmqsc < commnds.in > report.out
```

The command file `commnds.in` contains the following:

```
* This is a sample MQSC command file

* Step 1 - Create the queue
DEFINE QLOCAL(TEST) +
  REPLACE +
  DESCR('This is a test queue') +
  TRIGGER +
  INITQ(SYSTEM.SAMPLE.TRIGGER)

* Step 2 - Display selected queue attributes
DISPLAY Q(TEST) +
  DESCR +
  TRIGGER +
  INITQ +
  GET +
  PUT

* Step 3 - Delete the queue
DELETE QLOCAL (TEST)
```

Figure 9. Example command input file for UNIX systems

The following report is sent to report.out, the output file:

```

5765-115 (C) Copyright IBM Corp. 1993,1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
  : * This is a sample MQSC command file
  :
  : * Step 1 - Create the queue
1  : DEFINE QLOCAL(TEST) +
  :         REPLACE +
  :         DESCR('This is a test queue') +
  :         TRIGGER +
  :         INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006 MQSeries queue created.
  :
  : * Step 2 - Display selected queue attributes
2  : DISPLAY Q(TEST) +
  :         DESCR +
  :         TRIGGER +
  :         INITQ +
  :         GET +
  :         PUT
  :
AMQ8409 Display Queue details.
  DESCR(This is a test queue)      INITQ(SYSTEM.SAMPLE.TRIGGER)
  QUEUE(TEST)                     GET(ENABLED)
  PUT(ENABLED)                    TRIGGER
  :
  : * Step 3 - Delete the test queue
3  : DELETE QL(TEST)
AMQ8007 MQSeries queue deleted.
  :
  : * End of this sample
3  MQSC commands read.
0  commands have a syntax error.
0  commands cannot be processed.

```

Figure 10. Example report file from UNIX systems

### Error messages

If the command fails, there might be additional information about the problem in the error log.

### Verifying MQSC commands

To verify the commands in an input file, use the **runmqsc** command with the v flag:

```
runmqsc -v < commnds.in > report.out
```

This command verifies the MQSC commands in the input file and writes a report to the report file. The report contains the same information as is obtained when the commands are actually run, except that it does not contain any messages indicating the outcome of each command.

## Issuing UNIX system commands

---

## Appendix H. How to issue MQSC commands on Windows NT

This appendix tells you how to issue MQSC commands from a Windows NT system in which MQSeries is installed. Specifically, it tells you how to issue MQSC commands locally, that is, the commands are processed by a queue manager on the machine on which you issue them. Remote administration, where you issue commands to be processed by a queue manager on a different machine, is described in the *MQSeries System Administration* manual.

---

### Using the runmqsc command

To issue MQSC commands from a Windows NT system, use the **runmqsc** command from a Windows NT window. This command takes its input from standard in and sends its output to standard out. For more information about the **runmqsc** command, see the *MQSeries System Administration* manual.

### Issuing MQSC commands interactively

Provided that standard in is the keyboard and standard out is a Windows NT window, you can type in MQSC commands, one at a time, in a Windows NT window. The results from the commands are then displayed in the window.

For example, if you have a queue manager called QMAN1, you type:

```
runmqsc QMAN1
```

The queue manager QMAN1 responds with:

```
5697-177 (C) Copyright IBM Corp. 1996, 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
```

You can then type in your commands, as required, for example:

```
DEFINE QLOCAL(TEST) +
REPLACE +
DESCR('This is a test queue') +
TRIGGER +
INITQ(SYSTEM.SAMPLE.TRIGGER)
```

The plus sign indicates that the command is continued on the next line. When you press Enter, the queue manager confirms that the command has been carried out:

```
1: DEFINE QLOCAL(TEST) +
:   REPLACE +
:   DESCR('This is a test queue') +
:   TRIGGER +
:   INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006: MQSeries queue created.
```

If you accidentally add a + to the end of a line when you want to end a command, use the ; character to end the command.

You can then continue to enter additional commands, as required. When you have finished, type END, then press Enter. (Alternatively, you can type the end-of-file

## Issuing Windows NT commands

character, which in Windows NT is Ctrl+Z, then press Enter.) The queue manager then displays a summary report, for example:

```
3 MQSC commands read.  
0 commands have a syntax error.  
0 commands cannot be processed.
```

### Getting help

When you are issuing commands interactively on Windows NT, you can get help by entering command ? (where command is the name of the command you are interested in). MQSeries displays the syntax of the command.

If you enter an invalid command, MQSeries displays the syntax of the command if it can determine what the command was or, alternatively, displays a list of commands for you to select from.

## Issuing MQSC commands from a command file

You can put a sequence of MQSC commands in an ASCII text file, called a command file, using a standard editor. The following **runmqsc** command redirects both stdin and stdout. It takes input from the command file `commnds.in`, processes the commands contained in the file, and sends a report to the file `report.out`. The commands are processed by the default queue manager.

```
runmqsc < commnds.in > report.out
```

The command file `commnds.in` contains the following:

```
* This is a sample MQSC command file  
  
* Step 1 - Create the queue  
DEFINE QLOCAL(TEST) +  
  REPLACE +  
  DESCR('This is a test queue') +  
  TRIGGER +  
  INITQ(SYSTEM.SAMPLE.TRIGGER)  
  
* Step 2 - Display selected queue attributes  
DISPLAY Q(TEST) +  
  DESCR +  
  TRIGGER +  
  INITQ +  
  GET +  
  PUT  
  
* Step 3 - Delete the queue  
DELETE QLOCAL (TEST)
```

Figure 11. Example command input file for Windows NT



The following report is sent to report.out, the output file:

```

5697-177 (C) Copyright IBM Corp. 1996, 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.
: * This is a sample MQSC command file
:
: * Step 1 - Create the queue
1 : DEFINE QLOCAL(TEST) +
:     REPLACE +
:     DESCR('This is a test queue') +
:     TRIGGER +
:     INITQ(SYSTEM.SAMPLE.TRIGGER)
AMQ8006 MQSeries queue created.
:
: * Step 2 - Display selected queue attributes
2 : DISPLAY Q(TEST) +
:     DESCR +
:     TRIGGER +
:     INITQ +
:     GET +
:     PUT
:
AMQ8409 Display Queue details.
DESCR(This is a test queue)      INITQ(SYSTEM.SAMPLE.TRIGGER)
QUEUE(TEST)                     GET(ENABLED)
PUT(ENABLED)                     TRIGGER
:
: * Step 3 - Delete the test queue
3 : DELETE QL(TEST)
AMQ8007 MQSeries queue deleted.
:
: * End of this sample
3 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.

```

Figure 12. Example report file from Windows NT

### Error messages

If the command fails, there might be additional information about the problem in the error log and system event log.

### Verifying MQSC commands

To verify the commands in an input file, use the **runmqsc** command with the v flag:

```
runmqsc /v < commnds.in > report.out
```

This command verifies the MQSC commands in the input file and writes a report to the report file. The report contains the same information as is obtained when the commands are actually run, except that it does not contain any messages indicating the outcome of each command.

## Issuing Windows NT commands

---

## Appendix I. Notices

**The following paragraph does not apply to any 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 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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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 Laboratory Counsel, MP151, IBM United Kingdom Laboratories, Hursley Park, Winchester, Hampshire, England SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

---

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX	IBM	OS/2 Warp
AS/400	IMS	OS/400
BookManager	MQSeries	RACF
CICS	MVS/ESA	

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

## Notices

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other company, product, and service names, which may be denoted by a double asterisk (\*\*), may be trademarks or service marks of others.

# Index

## A

ACTION attribute  
 RESET TPIPE 157  
 RESOLVE CHANNEL 159  
 RESOLVE INDOUBT 160  
 active thread, display 148  
 active trace, display list of 149  
 administrator commands 177  
 alias queue  
 alter attributes 31  
 define 89  
 delete definition 117  
 display attributes 141  
 ALL attribute  
 DISPLAY CHANNEL 124  
 DISPLAY CHSTATUS 129  
 DISPLAY NAMELIST 136  
 DISPLAY PROCESS 137  
 DISPLAY QMGR 138  
 DISPLAY QUEUE 142  
 DISPLAY SECURITY 146  
 DISPLAY STGCLASS 147  
 alter  
 storage class 57  
 alter a namelist 28  
 alter a process definition 29  
 alter queue attributes  
 alias queue 31  
 local queue 33  
 model queue 45  
 remote queue 53  
 alter security attributes 56  
 alter the queue manager 41  
 alter trace 59  
 application identifier  
 See APPLICID attribute  
 application type  
 See APPLTYPE attribute  
 APPLICID attribute  
 ALTER PROCESS 29  
 DEFINE PROCESS 86  
 DISPLAY PROCESS 137  
 APPLTYPE attribute  
 ALTER PROCESS 29  
 DEFINE PROCESS 86  
 DISPLAY PROCESS 137  
 archive a log 60  
 AUTHOREV attribute  
 ALTER QMGR 42  
 DISPLAY QMGR 139

auto-definition exit program 42, 139  
 auto-definition of channels 42, 139  
 AUTOSTART attribute  
 ALTER CHANNEL 17  
 DEFINE CHANNEL 70  
 DISPLAY CHANNEL 124

## B

backing up the log 60  
 backout count, harden  
 See HARDENBO attribute  
 backout requeue name  
 See BOQNAME attribute  
 backout threshold  
 See BOTHRESH attribute  
 BATCHES attribute  
 DISPLAY CHSTATUS 131  
 BATCHINT attribute  
 ALTER CHANNEL 17  
 DEFINE CHANNEL 71  
 DISPLAY CHANNEL 124  
 BATCHSZ attribute  
 ALTER CHANNEL 17  
 DEFINE CHANNEL 71  
 DISPLAY CHANNEL 124  
 DISPLAY CHSTATUS 131  
 bibliography viii  
 BookManager xii  
 bootstrap data set, recover 154  
 BOQNAME attribute  
 ALTER QLOCAL 34  
 ALTER QMODEL 46  
 DEFINE QLOCAL 94  
 DEFINE QMODEL 102  
 DISPLAY QUEUE 144  
 BOTHRESH attribute  
 ALTER QLOCAL 34  
 ALTER QMODEL 46  
 DEFINE QLOCAL 94  
 DEFINE QMODEL 102  
 DISPLAY QUEUE 144  
 BSDS, recover 154  
 buffer pool number  
 See BUFFPOOL attribute  
 buffer pool, defining 63  
 BUFFERS attribute  
 DEFINE BUFFPOOL 63  
 BUFFPOOL attribute  
 DEFINE PSID 88  
 BUFSRCVD attribute  
 DISPLAY CHSTATUS 131

## index

- BUFSSSENT attribute
  - DISPLAY CHSTATUS 131
- building command scripts 3
- building commands
  - characters with special meanings 3
  - rules for 1
- BYTSRCVD attribute
  - DISPLAY CHSTATUS 131
- BYTSSSENT attribute
  - DISPLAY CHSTATUS 131
- C**
- CCSID attribute
  - DISPLAY QMGR 139
- CHAD attribute
  - ALTER QMGR 42
  - DISPLAY QMGR 139
- CHADEV attribute
  - ALTER QMGR 42
  - DISPLAY QMGR 139
- CHADEXIT attribute
  - ALTER QMGR 42
  - DISPLAY QMGR 139
- channel
  - alter attributes 11
  - auto-definition 42, 139
  - define attributes 64
  - delete definition 114
  - display 122
  - ping 152
  - reset 156
  - resolve 159
  - start 162
  - start initiator 163
  - start listener 165
  - stop 171
- channel initiator
  - start 163
  - stop 172
- channel status, displaying 126
- channels, rules for names of 7
- CHLTYPE attribute
  - ALTER CHANNEL 18
  - DEFINE CHANNEL 71
  - DISPLAY CHANNEL 124
- CHSTADA attribute
  - DISPLAY CHSTATUS 131
- CHSTATI attribute
  - DISPLAY CHSTATUS 131
- CLASS attribute
  - ALTER TRACE 59
  - DISPLAY TRACE 150
  - START TRACE 168
  - STOP TRACE 177
- clear a local queue 62
- CMDLEVEL attribute
  - DISPLAY QMGR 139
- coded character set identifier
  - See CCSID attribute
- command input queue name
  - See COMMANDQ attribute
- command prefix string 189
- command scripts, building 3
- command server
  - display status 133
  - start 164
  - stop 173
- command string
  - entering quotes 2
  - preserving case 2
- command summary 179
- COMMANDQ attribute
  - DISPLAY QMGR 139
- commands 177
  - rules for building 1
  - rules for naming objects in 5
  - rules for using 1
  - synonym 2
- COMMENT attribute
  - ALTER TRACE 59
  - DISPLAY TRACE 149
  - START TRACE 167
  - STOP TRACE 176
- CONNNAME attribute
  - ALTER CHANNEL 18
  - DEFINE CHANNEL 71
  - DISPLAY CHANNEL 124
  - DISPLAY CHSTATUS 129
- CONVERT attribute
  - ALTER CHANNEL 19
  - DEFINE CHANNEL 73
  - DISPLAY CHANNEL 124
- CPF 189
- CPILEVEL attribute
  - DISPLAY QMGR 139
- CRDATE attribute
  - DISPLAY QUEUE 144
- create a queue
  - See define a queue
- creation date
  - See CRDATE attribute
- creation time
  - See CRTIME attribute
- CRTIME attribute
  - DISPLAY QUEUE 144
- CURDEPTH attribute
  - DISPLAY QUEUE 144
- CURLUWID attribute
  - DISPLAY CHSTATUS 129

CURMSG attribute  
 DISPLAY CHSTATUS 130  
 CURRENT attribute  
 DISPLAY CHSTATUS 129  
 current queue depth  
 See CURDEPTH attribute  
 CURSEQNO attribute  
 DISPLAY CHSTATUS 130

## D

DATALEN attribute  
 PING CHANNEL 152  
 dead-letter queue name  
 See DEADQ attribute  
 DEADQ attribute  
 ALTER QMGR 42  
 DISPLAY QMGR 139  
 default message persistence  
 See DEFPSIST attribute  
 default message priority  
 See DEFPRTY attribute  
 default queue type  
 See DEFTYPE attribute  
 default share options  
 See DEFSOPT attribute  
 default transmission queue name  
 See DEFXMITQ attribute  
 define  
 buffer pool 63  
 list of queue names 83  
 maximum number of messages 82  
 namelist 83  
 page set 88, 112  
 process definition 85  
 security options 56  
 storage class 112  
 define a queue  
 alias queue 89  
 local queue 92  
 model queue 100  
 queue-manager alias 108  
 remote queue 108  
 reply-to queue alias 108  
 DEFPRTY attribute  
 ALTER QALIAS 31  
 ALTER QLOCAL 34  
 ALTER QMODEL 46  
 ALTER QREMOTE 53  
 DEFINE QALIAS 90  
 DEFINE QLOCAL 93  
 DEFINE QMODEL 101  
 DEFINE QREMOTE 109  
 DISPLAY QUEUE 144  
 DEFPSIST attribute  
 ALTER QALIAS 31

DEFPSIST attribute (*continued*)  
 ALTER QLOCAL 34  
 ALTER QMODEL 46  
 ALTER QREMOTE 54  
 DEFINE QALIAS 90  
 DEFINE QLOCAL 94  
 DEFINE QMODEL 101  
 DEFINE QREMOTE 109  
 DISPLAY QUEUE 144  
 DEFSOPT attribute  
 ALTER QLOCAL 35  
 ALTER QMODEL 47  
 DEFINE QLOCAL 94  
 DEFINE QMODEL 102  
 DISPLAY QUEUE 144  
 DEFTYPE attribute  
 ALTER QMODEL 52  
 DEFINE QMODEL 107  
 DISPLAY QUEUE 144  
 DEFXMITQ attribute  
 ALTER QMGR 42  
 DISPLAY QMGR 139  
 delete  
 list of queue names 115  
 namelist 115  
 process definition 116  
 storage class 121  
 delete a queue definition  
 alias queue 117  
 local queue 118  
 model queue 119  
 remote queue 120  
 DESCR attribute  
 ALTER CHANNEL 19  
 ALTER NAMELIST 28  
 ALTER PROCESS 30  
 ALTER QALIAS 32  
 ALTER QLOCAL 34  
 ALTER QMGR 43  
 ALTER QMODEL 46  
 ALTER QREMOTE 54  
 ALTER STGCLASS 57  
 DEFINE CHANNEL 73  
 DEFINE NAMELIST 84  
 DEFINE PROCESS 87  
 DEFINE QALIAS 90  
 DEFINE QLOCAL 94  
 DEFINE QMODEL 102  
 DEFINE QREMOTE 110  
 DEFINE STGCLASS 112  
 DISPLAY CHANNEL 124  
 DISPLAY NAMELIST 136  
 DISPLAY PROCESS 137  
 DISPLAY QMGR 139  
 DISPLAY QUEUE 144  
 DISPLAY STGCLASS 147

## index

### description

See DESCR attribute

### DEST attribute

DISPLAY TRACE 150  
START TRACE 168  
STOP TRACE 177

### destination

See DEST attribute

### DETAIL attribute

DISPLAY TRACE 149

### Digital OpenVMS

error messages 187  
example command input file 186  
example report file 187  
issuing commands from a command file 186  
issuing commands interactively 185  
runmqsc command 185  
verifying commands 188

### directing MVS/ESA commands 190

### DISCINT attribute

ALTER CHANNEL 19  
DEFINE CHANNEL 73  
DISPLAY CHANNEL 124

### display

command server status 133  
contents of namelist 136  
current state of a page set 151  
distributed queue management 134  
DQM 134  
information about threads 148  
list of active traces 149  
list of queue names 136  
maximum number of messages 135  
process attributes 137  
queue attributes 141  
queue manager attributes 138  
security attributes 146  
storage class 147  
unit-of-work ID 148

### DISTL attribute

ALTER QLOCAL 35  
ALTER QMODEL 47  
DEFINE QLOCAL 94  
DEFINE QMODEL 102  
DISPLAY QMGR 139  
DISPLAY QUEUE 144

### dual BSDS, reestablish 154

## E

### environment information

See ENVRDATA attribute

### ENVPARM attribute

START CHINIT 163  
START QMGR 166

### ENVRDATA attribute

ALTER PROCESS 30  
DEFINE PROCESS 87  
DISPLAY PROCESS 137

### error messages

Digital OpenVMS 187  
OS/2 Warp 193  
Tandem NSK 199  
UNIX systems 203  
Windows NT 207

### example OS/400 MQSeries command 195

### examples

command input file for Digital OpenVMS 186  
command input file for OS/2 Warp 192  
command input file for OS/400 195  
command input file for Tandem NSK 198  
command input file for UNIX systems 202  
command input file for Windows NT 206  
report file from Digital OpenVMS 187  
report file from OS/2 Warp 193  
report file from OS/400 196  
report file from Tandem NSK 199  
report file from UNIX systems 203  
report file from Windows NT 207

### excessive backout requeue name

See BOQNAME attribute

## F

### FIFO queue

ALTER QLOCAL 36  
ALTER QMODEL 48  
DEFINE QLOCAL 96  
DEFINE QMODEL 104

### FORCE attribute

ALTER QALIAS 31  
ALTER QLOCAL 34  
ALTER QMGR 41  
ALTER QREMOTE 53

## G

### GET attribute

ALTER QALIAS 32  
ALTER QLOCAL 35  
ALTER QMODEL 47  
DEFINE QALIAS 91  
DEFINE QLOCAL 94  
DEFINE QMODEL 102  
DISPLAY QUEUE 144

### getting help when issuing commands

OS/2 Warp 192  
UNIX systems 202  
Windows NT 206



**H**

handles indicating queue open for input  
     See IPPROCS attribute

handles indicating queue open for output  
     See OPPROCS attribute

handles, maximum number of open  
     See MAXHANDS attribute

harden backout count  
     See HARDENBO attribute

HARDENBO attribute  
     ALTER QLOCAL 36  
     ALTER QMODEL 48  
     DEFINE QLOCAL 96  
     DEFINE QMODEL 104  
     DISPLAY QUEUE 144

HBINT attribute  
     ALTER CHANNEL 20, 73  
     DISPLAY CHANNEL 124

how to issue commands  
     Digital OpenVMS 185  
     MVS/ESA 189  
     OS/2 Warp 191  
     OS/400 195  
     Tandem NSK 197  
     UNIX systems 201  
     Windows NT 205

HTML (Hypertext Markup Language) xiii  
 Hypertext Markup Language (HTML) xiii

**I**

IFCID attribute  
     ALTER TRACE 59  
     START TRACE 169

IMS Tpipe  
     reset sequence numbers manually 157

in-doubt thread  
     display 148  
     resolve manually 160

INDOUBT attribute  
     DISPLAY CHSTATUS 130

INDXTYPE attribute  
     ALTER QLOCAL 35  
     ALTER QMODEL 47  
     DEFINE QLOCAL 95  
     DEFINE QMODEL 102  
     DISPLAY QUEUE 144

Information Presentation Facility (IPF) xiii

INHIBTEV attribute  
     ALTER QMGR 43  
     DISPLAY QMGR 139

initiation queue name  
     See INITQ attribute

INITQ attribute  
     ALTER QLOCAL 35

INITQ attribute (*continued*)  
     ALTER QMODEL 47  
     DEFINE QLOCAL 95  
     DEFINE QMODEL 103  
     DISPLAY QUEUE 144  
     START CHINIT 163

INTERVAL attribute  
     ALTER SECURITY 56  
     DISPLAY SECURITY 146

IPF (Information Presentation Facility) xiii

IPPROCS attribute  
     DISPLAY QUEUE 144

**J**

JOBNAME attribute  
     DISPLAY CHSTATUS 131

**L**

LIKE attribute  
     DEFINE CHANNEL 74  
     DEFINE NAMELIST 83  
     DEFINE PROCESS 86  
     DEFINE QALIAS 89  
     DEFINE QLOCAL 93  
     DEFINE QMODEL 101  
     DEFINE QREMOTE 109  
     DEFINE STGCLASS 112

list of queue names  
     alter 28  
     define 83  
     delete 115  
     display 136

listener  
     start 165  
     stop 174

local queue  
     alter attributes 33  
     clear 62  
     define 92  
     delete definition 118  
     display attributes 141

LOCALEV attribute  
     ALTER QMGR 43  
     DISPLAY QMGR 140

log, archive 60

LONGRTS attribute  
     DISPLAY CHSTATUS 132

LONGRTY attribute  
     ALTER CHANNEL 20  
     DEFINE CHANNEL 74  
     DISPLAY CHANNEL 124

LONGTMR attribute  
     ALTER CHANNEL 20  
     DEFINE CHANNEL 74

## index

LONGTMR attribute (*continued*)  
  DISPLAY CHANNEL 124  
LSTLUWID attribute  
  DISPLAY CHSTATUS 130  
LSTMSGDA attribute  
  DISPLAY CHSTATUS 132  
LSTMSGTI attribute  
  DISPLAY CHSTATUS 132  
LSTSEQNO attribute  
  DISPLAY CHSTATUS 130

## M

MAXDEPTH attribute  
  ALTER QLOCAL 36  
  ALTER QMODEL 48  
  DEFINE QLOCAL 95  
  DEFINE QMODEL 103  
  DISPLAY QUEUE 144  
MAXHANDS attribute  
  ALTER QMGR 43  
  DISPLAY QMGR 140  
maximum message depth  
  See MAXDEPTH attribute  
maximum message length  
  See MAXMSGL attribute  
maximum number of open handles  
  See MAXHANDS attribute  
maximum priority  
  See MAXPRTY attribute  
MAXMSGL attribute  
  ALTER CHANNEL 20  
  ALTER QLOCAL 36  
  ALTER QMGR 43  
  ALTER QMODEL 48  
  DEFINE CHANNEL 74  
  DEFINE QLOCAL 95  
  DEFINE QMODEL 103  
  DISPLAY CHANNEL 124  
  DISPLAY CHSTATUS 132  
  DISPLAY QMGR 140  
  DISPLAY QUEUE 144  
MAXPRTY attribute  
  DISPLAY QMGR 140  
maxsmsgs  
  define 82  
  display 135  
MAXUMSGS attribute  
  ALTER QMGR 43  
MCANAME attribute  
  ALTER CHANNEL 21  
  DEFINE CHANNEL 75  
  DISPLAY CHANNEL 125  
MCASTAT attribute  
  DISPLAY CHSTATUS 132  
MCATYPE attribute  
  ALTER CHANNEL 21  
  DEFINE CHANNEL 75  
  DISPLAY CHANNEL 125  
MCAUSER attribute  
  ALTER CHANNEL 21  
  DEFINE CHANNEL 75  
  DISPLAY CHANNEL 125  
message delivery sequence  
  See MSGDLVSQ attribute  
message depth, maximum  
  See MAXDEPTH attribute  
message length, maximum  
  See MAXMSGL attribute  
message persistence, default  
  See DEFPSIST attribute  
message priority for triggers, threshold  
  See TRIGMPRI attribute  
message priority, default  
  See DEFPRTY attribute  
MODE attribute  
  ARCHIVE LOG 60  
  STOP CHANNEL 171  
  STOP QMGR 175  
model queue  
  alter attributes 45  
  define 100  
  delete definition 119  
  display attributes 141  
MODENAME attribute  
  ALTER CHANNEL 21  
  DEFINE CHANNEL 75  
  DISPLAY CHANNEL 125  
MQSC commands  
  how to issue on Digital OpenVMS 185  
  how to issue on MVS/ESA 189  
  how to issue on OS/2 Warp 191  
  how to issue on OS/400 195  
  how to issue on Tandem NSK 197  
  how to issue on UNIX systems 201  
  how to issue on Windows NT 205  
MQSeries commands 177  
MQSeries for MVS/ESA commands  
  directing to the correct queue manager 190  
  how to issue 189  
MQSeries for OS/2 Warp  
  how to issue commands 191  
MQSeries for Windows NT  
  how to issue commands 205  
MQSeries on Digital OpenVMS  
  how to issue commands 185  
MQSeries on Tandem NSK  
  how to issue commands 197  
MQSeries on UNIX systems  
  how to issue commands 201

MQSeries publications viii

MRDATA attribute

- ALTER CHANNEL 21
- DEFINE CHANNEL 75
- DISPLAY CHANNEL 125

MREXIT attribute

- ALTER CHANNEL 21
- DEFINE CHANNEL 75
- DISPLAY CHANNEL 125

MRRTY attribute

- ALTER CHANNEL 22
- DEFINE CHANNEL 76
- DISPLAY CHANNEL 125

MRTMR attribute

- ALTER CHANNEL 22
- DEFINE CHANNEL 76
- DISPLAY CHANNEL 125

MSGDATA attribute

- ALTER CHANNEL 22
- DEFINE CHANNEL 76
- DISPLAY CHANNEL 125

MSGDLVSQ attribute

- ALTER QLOCAL 36
- ALTER QMODEL 48
- DEFINE QLOCAL 96
- DEFINE QMODEL 104
- DISPLAY QUEUE 144

MSGEXIT attribute

- ALTER CHANNEL 22
- DEFINE CHANNEL 76
- DISPLAY CHANNEL 125

MSGS

- DISPLAY CHSTATUS 132

MVS/ESA commands

- directing to the correct queue manager 190
- how to issue 189

## N

NAMCOUNT attribute

- DISPLAY NAMELIST 136

name spaces 5

namelist

- alter 28
- define 83
- delete 115
- display contents 136

namelists, rules for names of 7

NAMES attribute

- ALTER NAMELIST 28
- DEFINE NAMELIST 84
- DISPLAY NAMELIST 136

network identifier

- See NID attribute

NID attribute

- RESOLVE INDOUBT 160

NOHARDENBO attribute

- ALTER QLOCAL 36
- ALTER QMODEL 48
- DEFINE QLOCAL 96
- DEFINE QMODEL 104

NOPURGE attribute

- DELETE QLOCAL 118

NOREPLACE

- DEFINE QALIAS 90

NOREPLACE attribute

- DEFINE CHANNEL 77
- DEFINE NAMELIST 83
- DEFINE PROCESS 86
- DEFINE QLOCAL 93
- DEFINE QMODEL 101
- DEFINE QREMOTE 109
- DEFINE STGCLASS 113

NOSHARE attribute

- ALTER QLOCAL 37
- ALTER QMODEL 49
- DEFINE QLOCAL 96
- DEFINE QMODEL 104

NOTRIGGER attribute

- ALTER QLOCAL 37
- ALTER QMODEL 49
- DEFINE QLOCAL 96
- DEFINE QMODEL 104

NPMSPEED attribute

- ALTER CHANNEL 23
- DEFINE CHANNEL 77
- DISPLAY CHANNEL 125

## O

objects, reserved names 7

operator commands 177

OPPROCS attribute

- DISPLAY QUEUE 144

OS/2 Warp

- error messages 193
- example command input file 192
- example report file 193
- getting help when issuing commands 192
- issuing commands from a command file 192
- issuing commands interactively 191
- runmqsc command 191
- verifying commands 193

OS/400

- example command input file 195
- example report file 196
- how to issue commands 195
- verifying commands 196

OS/400 MQSeries command, example 195

## index

### P

page set  
  define 88, 112  
  display usage 151  
page set identifier  
  See PSID attribute  
PARM attribute  
  START CHINIT 163  
  START QMGR 166  
PASSWORD attribute  
  ALTER CHANNEL 23  
  DEFINE CHANNEL 78  
  DISPLAY CHANNEL 125  
PERFMEV attribute  
  ALTER QMGR 43  
  DISPLAY QMGR 140  
PLATFORM attribute  
  DISPLAY QMGR 140  
PostScript format xii  
priority queue  
  ALTER QLOCAL 36  
  ALTER QMODEL 48  
  DEFINE QLOCAL 96  
  DEFINE QMODEL 104  
priority, maximum  
  See MAXPRTY attribute  
PROCESS attribute  
  ALTER QLOCAL 37  
  ALTER QMODEL 49  
  DEFINE QLOCAL 96  
  DEFINE QMODEL 104  
  DISPLAY QUEUE 144  
process definition  
  alter 29  
  define 85  
  delete 116  
  display 137  
processes, rules for names of 7  
PSID attribute  
  ALTER STGCLASS 57  
  DEFINE STGCLASS 113  
  DISPLAY STGCLASS 147  
  DISPLAY USAGE 151  
publications  
  MQSeries viii  
PURGE attribute  
  DELETE QLOCAL 118  
PUT attribute  
  ALTER QALIAS 32  
  ALTER QLOCAL 34  
  ALTER QMODEL 46  
  ALTER QREMOTE 54  
  DEFINE QALIAS 90  
  DEFINE QLOCAL 94  
  DEFINE QMODEL 102

PUT attribute (*continued*)  
  DEFINE QREMOTE 110  
  DISPLAY QUEUE 144  
PUTAUT attribute  
  ALTER CHANNEL 24  
  DEFINE CHANNEL 78  
  DISPLAY CHANNEL 125

### Q

QDEPTHHI attribute  
  ALTER QLOCAL 37  
  ALTER QMODEL 49  
  DEFINE QLOCAL 97  
  DEFINE QMODEL 105  
  DISPLAY QUEUE 144  
QDEPTHLO attribute  
  ALTER QLOCAL 37  
  ALTER QMODEL 49  
  DEFINE QLOCAL 97  
  DEFINE QMODEL 105  
  DISPLAY QUEUE 144  
QDPHIEV attribute  
  ALTER QLOCAL 37  
  ALTER QMODEL 49  
  DEFINE QLOCAL 97  
  DEFINE QMODEL 105  
  DISPLAY QUEUE 144  
QDPLOEV attribute  
  ALTER QLOCAL 38  
  ALTER QMODEL 50  
  DEFINE QLOCAL 97  
  DEFINE QMODEL 105  
  DISPLAY QUEUE 144  
QDPMAXEV attribute  
  ALTER QLOCAL 38  
  ALTER QMODEL 50  
  DEFINE QLOCAL 97  
  DEFINE QMODEL 105  
  DISPLAY QUEUE 145  
QMNAME attribute  
  ALTER CHANNEL 24  
  DEFINE CHANNEL 78  
  DISPLAY CHANNEL 125  
  DISPLAY QMGR 140  
QSVCIIEV attribute  
  ALTER QLOCAL 38  
  ALTER QMODEL 50  
  DEFINE QLOCAL 98  
  DEFINE QMODEL 106  
  DISPLAY QUEUE 145  
QSVCIINT attribute  
  ALTER QLOCAL 38  
  ALTER QMODEL 50  
  DEFINE QLOCAL 98  
  DEFINE QMODEL 106

QSVICINT attribute (*continued*)  
 DISPLAY QUEUE 145  
 QTYPE attribute  
 DISPLAY QUEUE 145  
 queue attributes, display 141  
 queue creation date  
 See CRDATE attribute  
 queue creation time  
 See CRTIME attribute  
 queue depth  
 See CURDEPTH attribute  
 queue manager  
 alter attributes 41  
 directing MVS/ESA commands to 190  
 display attributes 138  
 ping 153  
 start 166  
 stop 175  
 queue manager name  
 See QMNAME attribute  
 queue names 6  
 queue type, default  
 See DEFTYPE attribute  
 queue-manager alias, defining 108  
 queues  
 reserved names 6  
 rules for names of 6

## R

railroad diagrams, how to read 8  
 RCVDATA attribute  
 ALTER CHANNEL 24  
 DEFINE CHANNEL 78  
 DISPLAY CHANNEL 125  
 RCVEXIT attribute  
 ALTER CHANNEL 24  
 DEFINE CHANNEL 78  
 DISPLAY CHANNEL 125  
 RCVSEQ attribute  
 RESET TPIPE 158  
 rebuild security 155  
 recover BSDS 154  
 reestablish dual BSDS 154  
 refresh security 155  
 remote queue  
 alter attributes 53  
 define 108  
 delete definition 120  
 display attributes 141  
 remote queue manager name  
 See RQMNAME attribute  
 remote queue name  
 See RNAME attribute  
 REMOTEEV attribute  
 ALTER QMGR 44

REMOTEEV attribute (*continued*)  
 DISPLAY QMGR 140  
 REPLACE attribute  
 DEFINE CHANNEL 77  
 DEFINE NAMELIST 83  
 DEFINE PROCESS 86  
 DEFINE QALIAS 90  
 DEFINE QLOCAL 93  
 DEFINE QMODEL 101  
 DEFINE QREMOTE 109  
 DEFINE STGCLASS 113  
 reply-to queue alias, defining 108  
 reserved names  
 objects 7  
 queues 6  
 reset Tpipe manually 157  
 resolve in-doubt thread manually 160  
 resource manager identifier  
 See RMID attribute  
 restart, message persistence  
 See DEFPSIST attribute  
 retention interval  
 See RETINTVL attribute  
 RETINTVL attribute  
 ALTER QLOCAL 38  
 ALTER QMODEL 50  
 DEFINE QLOCAL 98  
 DEFINE QMODEL 106  
 DISPLAY QUEUE 145  
 RMID attribute  
 DISPLAY TRACE 150  
 START TRACE 169  
 STOP TRACE 177  
 RNAME attribute  
 ALTER QREMOTE 54  
 DEFINE QREMOTE 110  
 DISPLAY QUEUE 145  
 RQMNAME attribute  
 ALTER QREMOTE 54  
 DEFINE QREMOTE 110  
 DISPLAY QUEUE 145  
 rules for using commands 1  
 runmqsc command  
 using on Digital OpenVMS 185  
 using on OS/2 Warp 191  
 using on Tandem NSK 197  
 using on UNIX systems 201  
 using on Windows NT 205

## S

SAVED attribute  
 DISPLAY CHSTATUS 129  
 SCOPE attribute  
 ALTER QALIAS 32  
 ALTER QLOCAL 39

## index

- SCOPE attribute (*continued*)
  - ALTER QREMOTE 55
  - DEFINE QALIAS 91
  - DEFINE QLOCAL 98
  - DEFINE QREMOTE 111
  - DISPLAY QUEUE 145
- SCYDATA attribute
  - ALTER CHANNEL 24
  - DEFINE CHANNEL 79
  - DISPLAY CHANNEL 125
- SCYEXIT attribute
  - ALTER CHANNEL 24
  - DEFINE CHANNEL 79
  - DISPLAY CHANNEL 125
- security
  - alter attributes 56
  - display attributes 146
  - rebuild 155
  - refresh 155
  - reverify 161
- SENDDATA attribute
  - ALTER CHANNEL 25
  - DEFINE CHANNEL 79
  - DISPLAY CHANNEL 125
- SENDEXIT attribute
  - ALTER CHANNEL 25
  - DEFINE CHANNEL 79
  - DISPLAY CHANNEL 125
- SENDSEQ attribute
  - RESET TPIPE 157
- SEQNUM attribute
  - RESET CHANNEL 156
- sequence numbers
  - resetting on an IMS Tpipe 157
- SEQWRAP attribute
  - ALTER CHANNEL 25
  - DEFINE CHANNEL 80
  - DISPLAY CHANNEL 125
- set queue attributes
  - alias queue 89
  - local queue 92
  - model queue 100
  - queue-manager alias 108
  - remote queue 108
  - reply-to queue alias 108
- SHARE attribute
  - ALTER QLOCAL 37
  - ALTER QMODEL 49
  - DEFINE QLOCAL 96
  - DEFINE QMODEL 104
  - DISPLAY QUEUE 145
- share options, default
  - See DEFSOPT attribute
- SHORTRTS attribute
  - DISPLAY CHSTATUS 132
- SHORTRTY attribute
  - ALTER CHANNEL 25
  - DEFINE CHANNEL 80
  - DISPLAY CHANNEL 125
- SHORTTMR attribute
  - ALTER CHANNEL 26
  - DEFINE CHANNEL 80
  - DISPLAY CHANNEL 125
- softcopy books xii
- start
  - channel 162
  - channel initiator 163
  - command server 164
  - listener 165
  - queue manager 166
  - trace 167
- STATUS attribute
  - DISPLAY CHSTATUS 130
- STGCLASS attribute
  - ALTER QLOCAL 39
  - ALTER QMODEL 51
  - DEFINE QLOCAL 99
  - DEFINE QMODEL 106
  - DISPLAY QUEUE 142, 145
- stop
  - channel 171
  - channel initiator 172
  - command server 173
  - listener 174
  - queue manager 175
  - trace 176
- STOPREQ attribute
  - DISPLAY CHSTATUS 132
- storage class
  - alter 57
  - define 112
  - delete 121
  - display 147
- storage class name
  - See STGCLASS attribute
- storage classes, rules for names of 7
- STRSTPEV attribute
  - ALTER QMGR 44
  - DISPLAY QMGR 140
- summary of commands 179
- SWITCHES attribute
  - DISPLAY SECURITY 146
- syncpoint support
  - See SYNCPT attribute
- SYNCPT attribute
  - DISPLAY QMGR 140
- syntax diagrams, how to read 8
- system-command input queue name
  - See COMMANDQ attribute

**T**

## Tandem NSK

- error messages 199
- example command input file 198
- example report file 199
- issuing commands from a command file 198
- issuing commands interactively 197
- runmqsc command 197
- verifying commands 200

## target queue name

See TARGQ attribute

## TARGQ attribute

- ALTER QALIAS 32
- DEFINE QALIAS 91
- DISPLAY QUEUE 145

## TDATA attribute

- START TRACE 170

## thread

- display information about 148
- resolving in-doubt manually 160

## TIME attribute

- ARCHIVE LOG 60

## TIMEOUT attribute

- ALTER SECURITY 56
- DISPLAY SECURITY 146

## TNO attribute

- ALTER TRACE 59
- DISPLAY TRACE 150
- STOP TRACE 177

## TPNAME attribute

- ALTER CHANNEL 26
- DEFINE CHANNEL 80
- DISPLAY CHANNEL 125

## trace

- alter events being traced 59
- classes 168
- destination 168
- display list of active 149
- start 167
- stop 176

## trace event identifier

See IFCID attribute

## trace number

See TNO attribute

## transmission queue name

See XMITQ attribute

## transmission queue name, default

See DEFXMITQ attribute

## TRIGDATA attribute

- ALTER QLOCAL 39
- ALTER QMODEL 51
- DEFINE QLOCAL 99
- DEFINE QMODEL 106
- DISPLAY QUEUE 145

## TRIGDPTH attribute

- ALTER QLOCAL 39
- ALTER QMODEL 51
- DEFINE QLOCAL 99
- DEFINE QMODEL 106
- DISPLAY QUEUE 145

## TRIGGER attribute

- ALTER QLOCAL 37
- ALTER QMODEL 49
- DEFINE QLOCAL 96
- DEFINE QMODEL 104
- DISPLAY QUEUE 145

## trigger depth

See TRIGDPTH attribute

## trigger interval

See TRIGINT attribute

## trigger message data

See TRIGDATA attribute

## trigger message priority

See TRIGMPRI attribute

## trigger message priority threshold

See TRIGMPRI attribute

## trigger type

See TRIGTYPE attribute

## TRIGINT attribute

- ALTER QMGR 44
- DISPLAY QMGR 140

## TRIGMPRI attribute

- ALTER QLOCAL 39
- ALTER QMODEL 51
- DEFINE QLOCAL 99
- DEFINE QMODEL 107
- DISPLAY QUEUE 145

## TRIGTYPE attribute

- ALTER QLOCAL 40
- ALTER QMODEL 51
- DEFINE QLOCAL 99
- DEFINE QMODEL 107
- DISPLAY QUEUE 145

## TRPTYPE attribute

- ALTER CHANNEL 26
- DEFINE CHANNEL 81
- DISPLAY CHANNEL 125

## TYPE attribute

- DISPLAY CHANNEL 124
- DISPLAY QUEUE 142
- DISPLAY THREAD 148

**U**

## undelivered-message queue name

See DEADQ attribute

## unit-of-work ID, display 148

## UNIX systems

- error messages 203
- example command input file 202

## index

### UNIX systems (*continued*)

- example report file 203
- getting help when issuing commands 202
- issuing commands from a command file 202
- issuing commands interactively 201
- runmqsc command 201
- verifying commands 203

### USAGE attribute

- ALTER QLOCAL 40
- ALTER QMODEL 51
- DEFINE QLOCAL 99
- DEFINE QMODEL 107
- DISPLAY QUEUE 145

### usage, page set

- display 151

### USERDATA attribute

- ALTER PROCESS 30
- DEFINE PROCESS 87
- DISPLAY PROCESS 137

### USERID attribute

- ALTER CHANNEL 26
- DEFINE CHANNEL 81
- DISPLAY CHANNEL 125
- DISPLAY TRACE 150
- START TRACE 170
- STOP TRACE 177

### using commands

- rules for 1

## V

### verifying commands

- Digital OpenVMS 188
- OS/2 Warp 193
- OS/400 196
- Tandem NSK 200
- UNIX systems 203
- Windows NT 207

## W

### WAIT attribute

- ARCHIVE LOG 60

### Windows Help

xiii

### Windows NT

- error messages 207
- example command input file 206
- example report file 207
- getting help when issuing commands 206
- issuing commands from a command file 206
- issuing commands interactively 205
- runmqsc command 205
- verifying commands 207

## X

### XCFGNAME attribute

- ALTER STGCLASS 57
- DEFINE STGCLASS 113
- DISPLAY STGCLASS 147
- RESET TPIPE 158

### XCFMNAME attribute

- ALTER STGCLASS 58
- DEFINE STGCLASS 113
- DISPLAY STGCLASS 147
- RESET TPIPE 157

### XMITQ attribute

- ALTER CHANNEL 27
- ALTER QREMOTE 55
- DEFINE CHANNEL 81
- DEFINE QREMOTE 111
- DISPLAY CHANNEL 125
- DISPLAY CHSTATUS 128
- DISPLAY QUEUE 145



---

## **Sending your comments to IBM**

**MQSeries**

**Command Reference**

**SC33-1369-09**

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form
- By fax:
  - From outside the U.K., after your international access code use 44 1962 870229
  - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: WINVMD(IDRCF)
  - Internet: idrcf@winvmd.vnet.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.



# Readers' Comments

## MQSeries

### Command Reference

#### SC33-1369-09

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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

Be sure to print your name and address below if you would like a reply.

---

Name

---

Address

---

Company or Organization

---

Telephone

---

Email



**You can send your comments POST FREE on this form from any one of these countries:**

Australia	Finland	Iceland	Netherlands	Singapore	United States
Belgium	France	Israel	New Zealand	Spain	of America
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

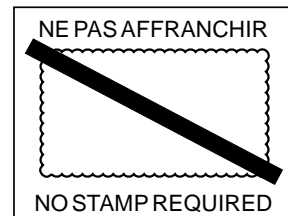
If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

1 Cut along this line

2 Fold along this line

**By air mail**  
*Par avion*

IBRS/CCR NUMBER: PHQ - D/1348/SO



**REPONSE PAYEE**  
**GRANDE-BRETAGNE**

IBM United Kingdom Laboratories  
Information Development Department (MP095)  
Hursley Park,  
WINCHESTER, Hants  
SO21 2ZZ United Kingdom

3 Fold along this line

*From:* Name \_\_\_\_\_  
Company or Organization \_\_\_\_\_  
Address \_\_\_\_\_  
\_\_\_\_\_

EMAIL \_\_\_\_\_  
Telephone \_\_\_\_\_

1 Cut along this line

4 Fasten here with adhesive tape



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SC33-1369-09





**MQSeries**

**Command Reference**