

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 245.

Eleventh edition (January 1999)

This edition applies to the following products:

- MQSeries for AIX® Version 5.1
- MQSeries for AS/400® Version 4 Release 2.1
- 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 5.1
- MQSeries for OS/2® Warp Version 5.1
- MQSeries for OS/390® Version 2 Release 1
- MQSeries for SINIX and DC/OSx Version 2 Release 2
- MQSeries for Sun Solaris Version 5.1
- MQSeries for Tandem NonStop Kernel Version 2 Release 2
- MQSeries for Windows NT® Version 5.1

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 1993,1999. All rights reserved.**

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

Contents

About this book	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	xi
MQSeries Level 1 product publications	xiii
Softcopy books	xiii
MQSeries information available on the Internet	xiv
Summary of changes	xv
Changes for this edition	xv
MQSeries for OS/390 V2.1	xv
MQSeries V5.1	xvi
MQSeries for AS/400 V4R2M1	xix
Changes for the tenth edition	xix
Chapter 1. Using MQSeries Commands	1
Rules for using MQSeries commands	1
Rules for naming MQSeries objects	5
How to read syntax diagrams	9
Chapter 2. The MQSeries commands	11
ALTER CHANNEL	14
ALTER NAMELIST	35
ALTER PROCESS	36
ALTER QALIAS	38
ALTER QLOCAL	41
ALTER QMGR	50
ALTER QMODEL	55
ALTER QREMOTE	63
ALTER SECURITY	67
ALTER STGCLASS	68
ALTER TRACE	70
ARCHIVE LOG	71
CLEAR QLOCAL	73
DEFINE BUFFPOOL	74
DEFINE CHANNEL	75
DEFINE MAXSMSGS	97
DEFINE NAMELIST	98
DEFINE PROCESS	100
DEFINE PSID	103
DEFINE QALIAS	104
DEFINE QLOCAL	108
DEFINE QMODEL	118
DEFINE QREMOTE	127
DEFINE STGCLASS	132
DELETE CHANNEL	134
DELETE NAMELIST	135
DELETE PROCESS	136

Contents

DELETE QALIAS	137
DELETE QLOCAL	138
DELETE QMODEL	139
DELETE QREMOTE	140
DELETE STGCLASS	141
DISPLAY CHANNEL	142
DISPLAY CHSTATUS	147
DISPLAY CLUSQMGR	154
DISPLAY CMDSERV	158
DISPLAY DQM	159
DISPLAY MAXSMSGS	160
DISPLAY NAMELIST	161
DISPLAY PROCESS	162
DISPLAY QMGR	164
DISPLAY QUEUE	168
DISPLAY SECURITY	175
DISPLAY STGCLASS	176
DISPLAY THREAD	178
DISPLAY TRACE	180
DISPLAY USAGE	182
PING CHANNEL	183
PING QMGR	184
RECOVER BSDS	185
REFRESH CLUSTER	186
REFRESH SECURITY	187
RESET CHANNEL	188
RESET CLUSTER	189
RESET TPIPE	190
RESOLVE CHANNEL	192
RESOLVE INDOUBT	193
RESUME QMGR	194
RVERIFY SECURITY	195
START CHANNEL	196
START CHINIT	197
START CMDSERV	198
START LISTENER	199
START QMGR	200
START TRACE	201
STOP CHANNEL	205
STOP CHINIT	207
STOP CMDSERV	208
STOP LISTENER	209
STOP QMGR	210
STOP TRACE	211
SUSPEND QMGR	213
Appendix A. Command summary	215
Appendix B. How to issue MQSC commands on Digital OpenVMS	221
Using the runmqsc command	221
Appendix C. How to issue MQSC commands on OS/2 Warp	225
Using the runmqsc command	225

Appendix D. How to issue MQSC commands on OS/390	229
Directing the command to the correct queue manager	230
Appendix E. How to issue MQSC commands on OS/400	231
Example OS/400 MQSeries command file and report	231
Appendix F. How to issue MQSC commands on Tandem NSK	233
Using the runmqsc command	233
Appendix G. How to issue MQSC commands on UNIX systems	237
Using the runmqsc command	237
Appendix H. How to issue MQSC commands on Windows NT	241
Using the runmqsc command	241
Appendix I. Notices	245
Trademarks	247
Index	249

Figures

1.	Example command input file for Digital OpenVMS	222
2.	Example report file from Digital OpenVMS	223
3.	Example command input file for OS/2 Warp	226
4.	Example report file from OS/2 Warp	227
5.	Example command input file for OS/400	231
6.	Example report file from OS/400	232
7.	Example command input file for Tandem NSK	234
8.	Example report file from Tandem NSK	235
9.	Example command input file for UNIX systems	239
10.	Example report file from UNIX systems	240
11.	Example command input file for Windows NT	243
12.	Example report file from Windows NT	244

Tables

1.	How to read syntax diagrams	9
2.	MQSeries operator and administrator commands	11
3.	Attributes that can be returned by the DISPLAY QUEUE command	171
4.	Destinations allowed for each trace type	202
5.	Constraints allowed for each trace type	202
6.	IFCID descriptions for IFCID trace events and classes	203
7.	Resource Manager identifiers that are allowed	203
8.	Commands for queue manager administration	215
9.	Commands for queue administration	216
10.	Commands for process definition administration	216
11.	Commands for namelist administration	217
12.	Commands for channel administration	217
13.	Commands for cluster administration	218
14.	Commands for security administration	218
15.	Commands for system-dependent function	219
16.	Other control commands in MQSeries for Tandem NonStop Kernel	220

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
- OS/2 Warp
- OS/390
- OS/400
- Tandem NSK
- UNIX operating systems
- Windows NT

The commands are listed in alphabetic order in Chapter 2, “The MQSeries commands” on page 11. A matrix at the start of each command description identifies platforms on which the command is valid. Table 2 on page 11 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 11. 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 221
- Appendix C, “How to issue MQSC commands on OS/2 Warp” on page 225
- Appendix D, “How to issue MQSC commands on OS/390” on page 229
- Appendix E, “How to issue MQSC commands on OS/400” on page 231
- Appendix F, “How to issue MQSC commands on Tandem NSK” on page 233
- Appendix G, “How to issue MQSC commands on UNIX systems” on page 237
- Appendix H, “How to issue MQSC commands on Windows NT” on page 241

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

- AIX
- AT&T GIS UNIX¹
- HP-UX
- SINIX and DC/OSx
- Sun Solaris

¹ 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 9. 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 V5.1
- MQSeries for AS/400 V4R2M1
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Digital OpenVMS V2.2
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for SINIX and DC/OSx V2.2
- MQSeries for Sun Solaris V5.1
- MQSeries for Tandem NonStop Kernel V2.2
- MQSeries for VSE/ESA V2.1
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT V5.1

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 xiii. 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, and the dead-letter queue handler. It also includes the syntax of the MQSeries control commands.

This book applies to the following MQSeries products only:

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

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 Format (PCF) messages, and installable services.

MQSeries Messages

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

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

MQSeries publications

- 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 these MQSeries products:

- MQSeries for AIX V5.1
- MQSeries for AS/400 V4R2M1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

MQSeries C++ is also supported by MQSeries clients supplied with these products and installed in the following environments:

- AIX
- HP-UX
- OS/2
- Sun Solaris
- Windows NT
- Windows 3.1
- Windows 95 and Windows 98

MQSeries Using Java®

MQSeries Using Java, SC34-5456, provides both guidance and reference information for users of the MQSeries Bindings for Java and the MQSeries Client for Java. MQSeries Java is supported by these MQSeries products:

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

MQSeries Administration Interface Programming Guide and Reference

The *MQSeries Administration Interface Programming Guide and Reference*, SC34-5390, provides information for users of the MQAI. The MQAI is a

programming interface that simplifies the way in which applications manipulate Programmable Command Format (PCF) messages and their associated data structures.

This book applies to the following MQSeries products only:

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

MQSeries Queue Manager Clusters

MQSeries Queue Manager Clusters, SC34-5349, describes MQSeries clustering. It explains the concepts and terminology and shows how you can benefit by taking advantage of clustering. It details changes to the MQI, and summarizes the syntax of new and changed MQSeries commands. It shows a number of examples of tasks you can perform to set up and maintain clusters of queue managers.

This book applies to the following MQSeries products only:

- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

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 Version 5 Release 1 Quick Beginnings, GC33-1867

MQSeries for AS/400

MQSeries for AS/400 Version 4 Release 2.1 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 Release 2 System Management Guide, SC33-1642

MQSeries for Digital OpenVMS

MQSeries for Digital OpenVMS Version 2 Release 2 System Management Guide, GC33-1791

MQSeries for Digital UNIX

MQSeries for Digital UNIX Version 2 Release 2.1 System Management Guide, GC34-5483

MQSeries for HP-UX

MQSeries for HP-UX Version 5 Release 1 Quick Beginnings, GC33-1869

MQSeries publications

MQSeries for OS/2 Warp

MQSeries for OS/2 Warp Version 5 Release 1 Quick Beginnings, GC33-1868

MQSeries for OS/390

MQSeries for OS/390 Version 2 Release 1 Licensed Program Specifications, GC34-5377

MQSeries for OS/390 Version 2 Release 1 Program Directory

MQSeries for OS/390 Version 2 Release 1 System Management Guide, SC34-5374

MQSeries for OS/390 Version 2 Release 1 Messages and Codes, GC34-5375

MQSeries for OS/390 Version 2 Release 1 Problem Determination Guide, GC34-5376

MQSeries link for R/3

MQSeries link for R/3 Version 1 Release 2 User's Guide, GC33-1934

MQSeries for SINIX and DC/OSx

MQSeries for SINIX and DC/OSx Version 2 Release 2 System Management Guide, GC33-1768

MQSeries for Sun Solaris

MQSeries for Sun Solaris Version 5 Release 1 Quick Beginnings, GC33-1870

MQSeries for Tandem NonStop Kernel

MQSeries for Tandem NonStop Kernel Version 2 Release 2 System Management Guide, GC33-1893

MQSeries for VSE/ESA™

MQSeries for VSE/ESA Version 2 Release 1 Licensed Program Specifications, GC34-5365

MQSeries for VSE/ESA Version 2 Release 1 System Management Guide, GC34-5364

MQSeries for Windows

MQSeries for Windows Version 2 Release 0 User's Guide, GC33-1822

MQSeries for Windows Version 2 Release 1 User's Guide, GC33-1965

MQSeries for Windows NT

MQSeries for Windows NT Version 5 Release 1 Quick Beginnings, GC34-5389

MQSeries for Windows NT Using the Component Object Model Interface, SC34-5387

MQSeries LotusScript Extension, SC34-5404

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 UnixWare Version 1 Release 4.1 User's Guide, SC33-1379

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

HTML format

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

- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1 (compiled HTML)
- MQSeries link for R/3 V1.2

The MQSeries books are also available in HTML format from the MQSeries product family Web site at:

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

Portable Document Format (PDF)

PDF files can be viewed and printed using the Adobe Acrobat Reader.

If you need to obtain the Adobe Acrobat Reader, or would like up-to-date information about the platforms on which the Acrobat Reader is supported, visit the Adobe Systems Inc. Web site at:

<http://www.adobe.com/>

PDF versions of relevant MQSeries books are supplied with these MQSeries products:

- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1

MQSeries on the Internet

- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1
- MQSeries link for R/3 V1.2

PDF versions of all current MQSeries books are also available from the MQSeries product family Web site at:

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

PostScript format

The MQSeries library is provided in PostScript (.PS) format with many MQSeries Version 2 products. Books in PostScript format can be printed on a PostScript printer or viewed with a suitable viewer.

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 and PDF formats.
- 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

This edition of *MQSeries Command Reference* applies to these new versions and releases of MQSeries products:

- MQSeries for AIX V5.1
- MQSeries for AS/400 V4R2M1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

Major new function supplied with each of these MQSeries products is summarized here.

MQSeries for OS/390 V2.1

MQSeries for OS/390 V2.1 is a new product for the OS/390 platform that offers functional enhancements over MQSeries for MVS/ESA™ V1.2. Those functional enhancements specific to MQSeries for OS/390 are summarized here. As a general rule, other function described in this book as supported by MQSeries for OS/390 is also supported by MQSeries for MVS/ESA V1.2.

MQSeries queue manager clusters

MQSeries queue managers can be connected to form a *cluster* of queue managers. Within a cluster, queue managers can make the queues they host available to every other queue manager. Any queue manager can send a message to any other queue manager in the same cluster without the need for explicit channel definitions, remote queue definitions, or transmission queues for each destination. The main benefits of MQSeries clusters are:

- Fewer system administration tasks
- Increased availability
- Workload balancing

Clusters are supported by these MQSeries products:

- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

See the book *MQSeries Queue Manager Clusters*, SC34-5349, for a complete description of this function.

Summary of changes

OS/390 Automatic Restart Manager (ARM)

If an MQSeries queue manager or channel initiator fails, the OS/390 Automatic Restart Manager (ARM) can restart it automatically on the same OS/390 image. If the OS/390 image itself fails, ARM can restart that image's subsystems and applications automatically on another OS/390 image in the sysplex, provided that the LU 6.2 communication protocol is being used. By removing the need for operator intervention, OS/390 ARM improves the availability of your MQSeries subsystems.

OS/390 Resource Recovery Services (RRS)

MQSeries Batch and TSO applications can participate in two-phase commit protocols with other RRS-enabled products, such as DB2®, coordinated by the OS/390 RRS facility.

MQSeries Workflow

MQSeries Workflow allows applications on various network clients to perform business functions through System/390® by driving one or more CICS®, IMS™, or MQSeries applications. This is achieved through format, rule, and table definition, rather than through application programming.

Support for C++

MQSeries for OS/390 V2.1 applications can be written in C++.

Euro support

MQSeries supports new and changed code pages that use the euro currency symbol. Details of code pages that include the euro symbol are provided in the *MQSeries Application Programming Reference* book.

MQSeries V5.1

The MQSeries Version 5 Release 1 products are:

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

The following new function is provided in all of the V5.1 products:

MQSeries queue manager clusters

MQSeries queue managers can be connected to form a *cluster* of queue managers. Within a cluster, queue managers can make the queues they host available to every other queue manager. Any queue manager can send a message to any other queue manager in the same cluster without the need for explicit channel definitions, remote queue definitions, or transmission queues for each destination. The main benefits of MQSeries clusters are:

- Fewer system administration tasks
- Increased availability
- Workload balancing

Clusters are supported by these MQSeries products:

- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for Sun Solaris V5.1

- MQSeries for Windows NT V5.1

See the book *MQSeries Queue Manager Clusters*, SC34-5349, for a complete description of this function.

MQSeries Administration Interface (MQAI)

The MQSeries Administration Interface is an MQSeries programming interface that simplifies manipulation of MQSeries PCF messages for administrative tasks. It is described in a new book, *MQSeries Administration Interface Programming Guide and Reference*, SC34-5390.

Support for Windows 98 clients

A Windows 98 client can connect to any MQSeries V5.1 server.

Message queue size

A message queue can be up to 2 GB.

Controlled, synchronous shutdown of a queue manager

A new option has been added to the **endmqm** command to allow controlled, synchronous shutdown of a queue manager.

Java support

The MQSeries Client for Java and MQSeries Bindings for Java are provided with all MQSeries V5.1 products. The client, bindings, and common files have been packaged into .jar files for ease of installation.

Euro support

MQSeries supports new and changed code pages that use the euro currency symbol. Details of code pages that include the euro symbol are provided in the *MQSeries Application Programming Reference* book.

Conversion of the EBCDIC new-line character

You can control the conversion of EBCDIC new-line characters to ensure that data transmitted from EBCDIC systems to ASCII systems and back to EBCDIC is unaltered by the ASCII conversion.

Client connections via MQCONN

A client application can specify the definition of the client-connection channel at run time in the MQCNO structure of the MQCONN call.

Additional new function in MQSeries for AIX V5.1

- The UDP transport protocol is supported.
- Sybase databases can participate in global units of work.
- Multithreaded channels are supported.

Additional new function in MQSeries for HP-UX V5.1

- MQSeries for HP-UX V5.1 runs on both HP-UX V10.20 and HP-UX V11.0.
- Multithreaded channels are supported.
- Both HP-UX kernel threads and DCE threads are supported.

Additional new function in MQSeries for OS/2 Warp V5.1

OS/2 high memory support is provided.

Additional new function in MQSeries for Sun Solaris V5.1

- MQSeries for Sun Solaris V5.1 runs on both Sun Solaris V2.6 and Sun Solaris 7.
- Sybase databases can participate in global units of work.
- Multithreaded channels are supported.

Additional new function in MQSeries for Windows NT V5.1

MQSeries for Windows NT V5.1 is part of the IBM Enterprise Suite for Windows NT. New function in this release includes:

- Close integration with Microsoft® Windows NT Version 4.0, including exploitation of extra function provided by additional Microsoft offerings. The main highlights are:
 - Graphical tools and applications for managing, controlling, and exploring MQSeries:
 - MQSeries Explorer—a snap-in for the Microsoft management console (MMC) that allows you to query, change, and create the local, remote, and cluster objects across an MQSeries network.
 - MQSeries Services—an MMC snap-in that controls the operation of MQSeries components, either locally or remotely within the Windows NT domain. It monitors the operation of MQSeries servers and provides extensive error detection and recovery functions.
 - MQSeries API Exerciser—a graphical application for exploring the messaging and queuing programming functions that MQSeries provides. It can also be used in conjunction with the MQSeries Explorer to gain a deeper understanding of the effects of MQSeries operations on objects and messages.
 - MQSeries Postcard—a sample application that can be used to verify an MQSeries installation, for either local or remote messaging.
 - Support for the following features of Windows NT has been added:
 - Windows NT performance monitor—used to access and display MQSeries information, such as the current depth of a queue and the rate at which message data is put onto and taken off queues.
 - ActiveDirectory—programmable access to MQSeries objects is available through the Active Directory Service Interfaces (ADSI).
 - Windows NT user IDs—previous MQSeries restrictions on the validity of Windows NT user IDs have been removed. All valid Windows NT user IDs are now valid identifiers for MQSeries operations. MQSeries uses the associated Windows NT Security Identifier (SID) and the Security Account Manager (SAM). The SID allows the MQSeries Object Authority Manager (OAM) to identify the specific user for an authorization request.
 - Windows NT registry—now used to hold all configuration and related data. The contents of any configuration (.INI) files from previous MQSeries installations of MQSeries for Windows NT products are migrated into the registry; the .INI files are then deleted.

- A set of Component Object Model (COM) classes, which allow ActiveX applications to access the MQSeries Message Queue Interface (MQI) and the MQSeries Administration Interface (MQAI).
- An online Quick Tour of the product concepts and functions.
- An online Information Center that gives you quick access to task help information, reference information, and Web-based online books and home pages.
- Simplified installation of MQSeries for Windows NT, with default options and automatic configuration.
- Support for web-based administration of an MQSeries network, which provides a simplified way of using the MQSC commands and scripts and allows you to create powerful macros for standard administration tasks.
- Support for MQSeries LotusScript Extension (MQLSX), which allows Lotus® Notes® applications that are written in LotusScript to communicate with applications that run in non-Notes environments.
- Support for Microsoft Visual Basic for Windows Version 5.0.
- Performance improvements over the MQSeries for Windows NT Version 5.0 product.
- Information and examples on how MQSeries applications can interface with and exploit the lightweight directory access protocol (LDAP) directories.
- Support for Sybase participation in global units of work.

MQSeries for AS/400 V4R2M1

New function in MQSeries for AS/400 V4R2M1 includes:

- Support for the MQSeries dead-letter queue handler
- Improvements to installation and migration procedures

Changes for the tenth 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 9.

Summary of changes

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 OS/390) see Part 2, "Programmable Command Formats" in 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 9

The general format of the commands is shown in Chapter 2, "The MQSeries commands" on page 11.

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 221
- Appendix C, "How to issue MQSC commands on OS/2 Warp" on page 225
- Appendix D, "How to issue MQSC commands on OS/390" on page 229
- Appendix E, "How to issue MQSC commands on OS/400" on page 231
- Appendix F, "How to issue MQSC commands on Tandem NSK" on page 233
- Appendix G, "How to issue MQSC commands on UNIX systems" on page 237
- Appendix H, "How to issue MQSC commands on Windows NT" on page 241

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 usually 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 OS/390, 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 OS/390.

Rules for using commands

- 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.

- 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 OS/390 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 OS/390
- 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 OS/390, 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 OS/390 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 Part 2, "Programmable Command Formats" in 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 OS/390 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 OS/390 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 OS/390, 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 OS/390 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.
5. On AS/400 systems, lowercase a-z, forward slash (/), and percent (%) are special characters. If you use any of these characters in a name, the name must be enclosed in quotation marks. Lowercase a-z characters are changed to uppercase if the name is not enclosed in quotation marks.

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. 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 OS/390)
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 OS/390 using CICS
SYSTEM.CHANNEL.INITQ	Queue used for distributed queuing (without CICS on OS/390)
SYSTEM.CHANNEL.REPLY.INFO	Queue used for distributed queuing on OS/390 without CICS
SYSTEM.CHANNEL.SEQNO	Queue used for distributed queuing on OS/390 using CICS
SYSTEM.CHANNEL.SYNCQ	Queue used for distributed queuing (without CICS on OS/390)
SYSTEM.CICS.INITIATION.QUEUE	Queue used for triggering (not for OS/390)
SYSTEM.CLUSTER.COMMAND.QUEUE	Queue used to communicate repository changes between queue managers (AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only)
SYSTEM.CLUSTER.REPOSITORY.QUEUE	Queue used to hold information about the repository (AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only)
SYSTEM.CLUSTER.TRANSMIT.QUEUE	Transmission queue for all destinations managed by cluster support (AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only)
SYSTEM.COMMAND.INPUT	Queue to which command messages are sent on OS/390
SYSTEM.COMMAND.REPLY.MODEL	Model queue definition for command replies (for OS/390)
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter queue (not for OS/390)
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue definition
SYSTEM.DEFAULT.INITIATION.QUEUE	Queue used for distributed queuing (not for OS/390)
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue definition
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue definition

Rules for naming objects

SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue definition
SYSTEM.MQSC.REPLY.QUEUE	Model queue definition for MQSC command replies (not for OS/390)

Other object names

Processes, namelists, and clusters 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. The following names are defined for MQSeries:

SYSTEM.ADMIN.SVRCONN	Server-connection channel used for remote administration of a queue manager by the MQSeries Explorer (remote administration is not available on OS/390)
SYSTEM.AUTO.RECEIVER	Default receiver channel for auto definition (not for AT&T GIS UNIX, Digital OpenVMS, OS/390, SINIX and DC/OSx, or Tandem NSK)
SYSTEM.AUTO.SVRCONN	Default server-connection channel for auto definition (not for AT&T GIS UNIX, Digital OpenVMS, OS/390, SINIX and DC/OSx, or Tandem NSK)
SYSTEM.DEF.CLNTCONN	Default client-connection channel definition
SYSTEM.DEF.CLUSRCVR	Default cluster-receiver channel definition (AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only)
SYSTEM.DEF.CLUSSDR	Default cluster-sender channel definition (AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only)
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 (AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only)
SYSTEM.DEFAULT.PROCESS	Default process definition
SYSTEMST	Default storage class definition (OS/390 only)

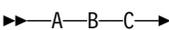
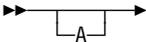
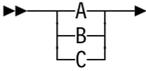
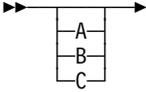
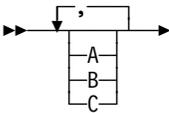
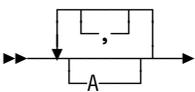
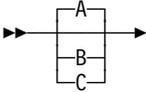
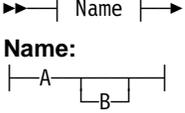
How to read syntax diagrams

This book 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. Name:
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.

Reading syntax diagrams

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	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT	Windows ¹
ALTER CHANNEL	√	√ ²	√	√	√	√	√	√
ALTER NAMELIST		√		√		√ ³	√	
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	√	√ ²	√	√	√	√	√	√
DEFINE MAXSMSGS	√ ⁴	√	√ ⁴					
DEFINE NAMELIST		√		√		√ ³	√	
DEFINE PROCESS	√	√	√	√	√	√	√	
DEFINE PSID		√						
DEFINE QALIAS	√	√	√	√	√	√	√	√
DEFINE QLOCAL	√	√	√	√	√	√	√	√
DEFINE QMODEL	√	√	√	√	√	√	√	√
DEFINE QREMOTE	√	√	√	√	√	√	√	√
DEFINE STGCLASS		√						
DELETE CHANNEL	√	√ ²	√	√	√	√	√	√
DELETE NAMELIST		√		√		√ ³	√	
DELETE PROCESS	√	√	√	√	√	√	√	
DELETE QALIAS	√	√	√	√	√	√	√	√
DELETE QLOCAL	√	√	√	√	√	√	√	√
DELETE QMODEL	√	√	√	√	√	√	√	√
DELETE QREMOTE	√	√	√	√	√	√	√	√
DELETE STGCLASS		√						
DISPLAY CHANNEL	√	√ ²	√	√	√	√	√	√ ⁵
DISPLAY CHSTATUS	√	√ ²	√	√	√	√	√	
DISPLAY CLUSQMGR		√		√		√ ³	√	
DISPLAY CMDSERV		√						

MQSeries commands

Table 2 (Page 2 of 3). MQSeries operator and administrator commands								
Command	Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT	Windows ¹
DISPLAY DQM		√ ²						
DISPLAY MAXSMSGS	√ ⁴	√	√ ⁴					
DISPLAY NAMELIST		√		√		√ ³	√	
DISPLAY PROCESS	√	√	√	√	√	√	√	
DISPLAY QALIAS ⁶		√	√	√		√ ³	√	
DISPLAY QCLUSTER ⁶		√		√		√ ³	√	
DISPLAY QLOCAL ⁶		√	√	√		√ ³	√	
DISPLAY QMGR	√	√	√	√	√	√	√	√ ⁵
DISPLAY QMODEL ⁶		√	√	√		√ ³	√	
DISPLAY QREMOTE ⁶		√	√	√		√ ³	√	
DISPLAY QUEUE	√	√	√	√	√	√	√	√ ⁵
DISPLAY SECURITY		√						
DISPLAY STGCLASS		√						
DISPLAY THREAD		√						
DISPLAY TRACE		√						
DISPLAY USAGE		√						
PING CHANNEL	√	√ ²	√	√	√	√	√	
PING QMGR	√		√	√	√	√	√	
RECOVER BSDS		√						
REFRESH CLUSTER		√		√		√ ³	√	
REFRESH SECURITY		√						
RESET CHANNEL	√	√ ²	√	√	√	√	√	√
RESET CLUSTER		√		√		√ ³	√	
RESET TPIPE		√						
RESOLVE CHANNEL	√	√ ²	√	√	√	√	√	√
RESOLVE INDOUBT		√						
RESUME QMGR		√		√		√ ³	√	
RVERIFY SECURITY		√						
START CHANNEL	√	√ ²	√	√	√	√	√	√
START CHINIT	√	√ ²	√	√		√	√	
START CMDSERV		√						
START LISTENER		√ ²	√	√			√	
START QMGR		√						
START TRACE		√						
STOP CHANNEL	√	√ ²	√	√	√	√	√	√
STOP CHINIT		√ ²						
STOP CMDSERV		√						
STOP LISTENER		√ ²						
STOP QMGR		√						
STOP TRACE		√						

Table 2 (Page 3 of 3). MQSeries operator and administrator commands

Command	Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT	Windows ¹
SUSPEND QMGR		√		√		√ ³	√	

Notes:

1. For a description of the syntax for commands on this platform, see the online *MQSeries for Windows Command Reference*.
2. On OS/390, 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 CKMC transaction. See Chapter 25, "Monitoring and controlling channels in OS/390 with CICS" in the *MQSeries Intercommunication* manual for information about this.
3. AIX, HP-UX, and Sun Solaris only.
4. This command is valid only on OS/390. For other platforms, use the MAXUMSGS keyword of the ALTER QMGR command instead.
5. MQSeries for Windows Version 2.1 only.
6. See "DISPLAY QUEUE" on page 168.

ALTER CHANNEL

ALTER CHANNEL

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use ALTER CHANNEL to alter the attributes of a channel.

Notes:

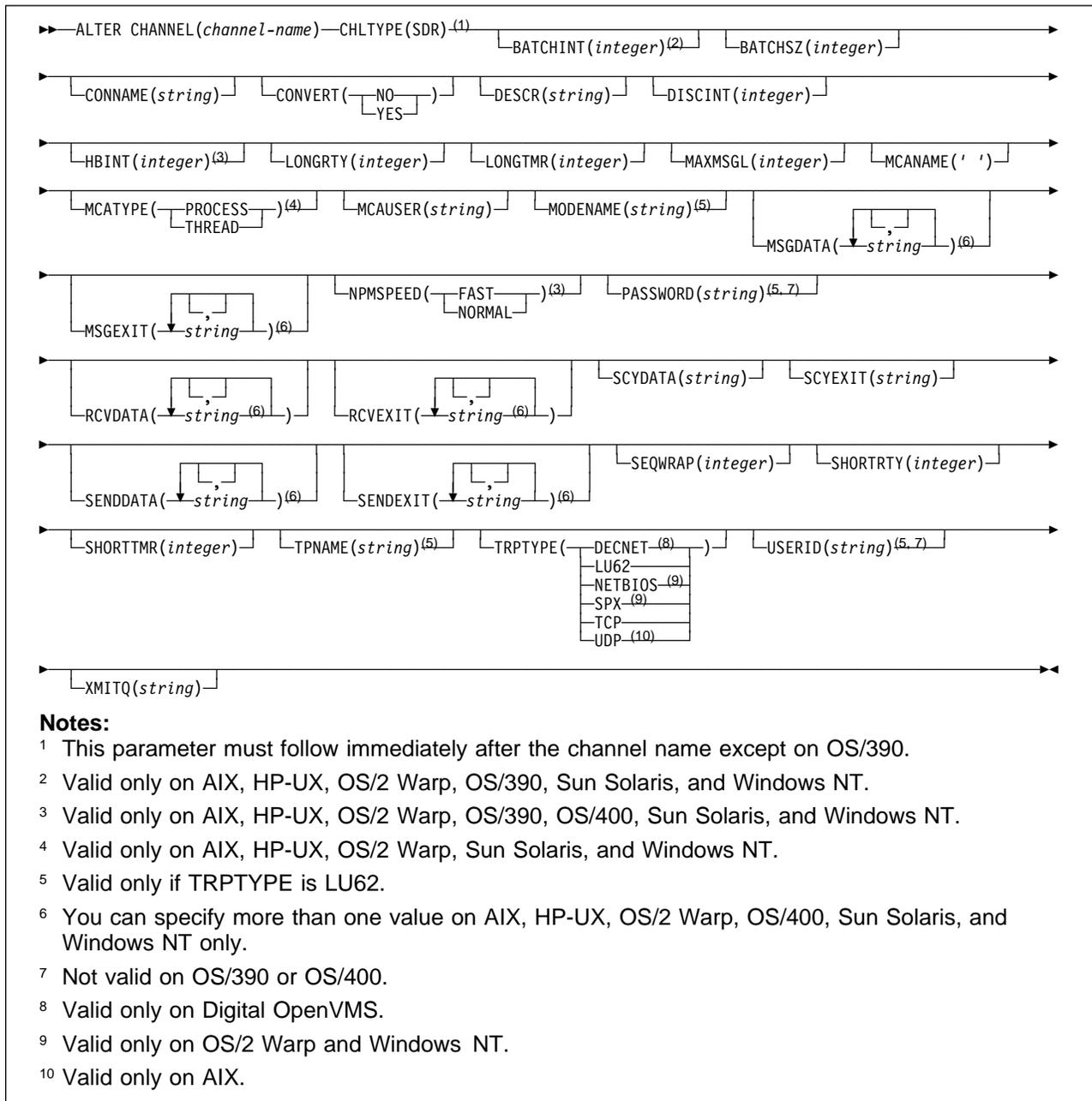
1. On OS/390, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 13.
2. For cluster-sender channels, you can only alter channels that have been created manually.

Synonym: ALT CHL

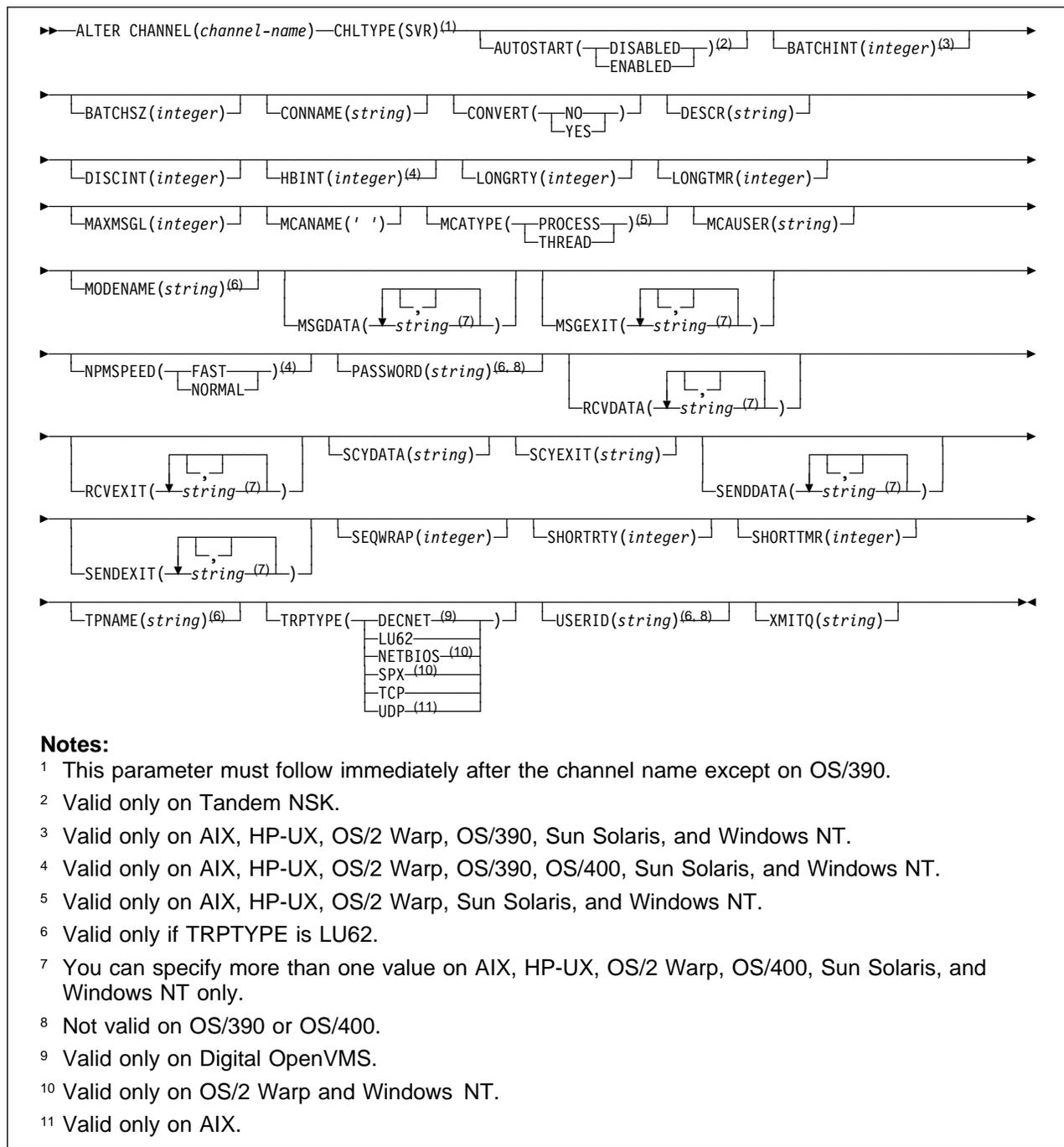
There is a separate syntax diagram for each type of channel:

- “Sender channel” on page 15
- “Server channel” on page 16
- “Receiver channel” on page 17
- “Requester channel” on page 18
- “Client-connection channel” on page 19
- “Server-connection channel” on page 20
- “Cluster-sender channel” on page 21
- “Cluster-receiver channel” on page 22

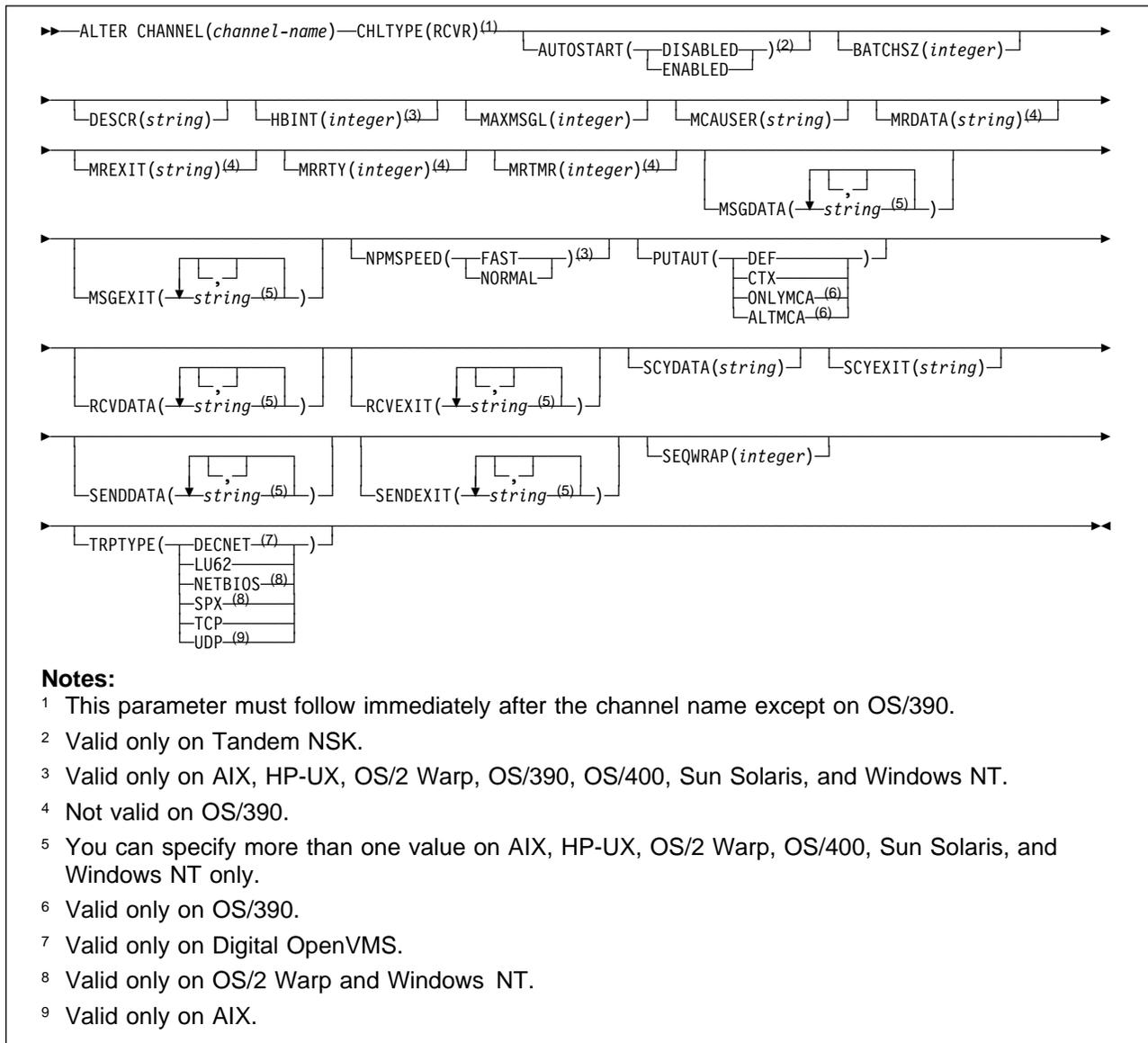
Sender channel



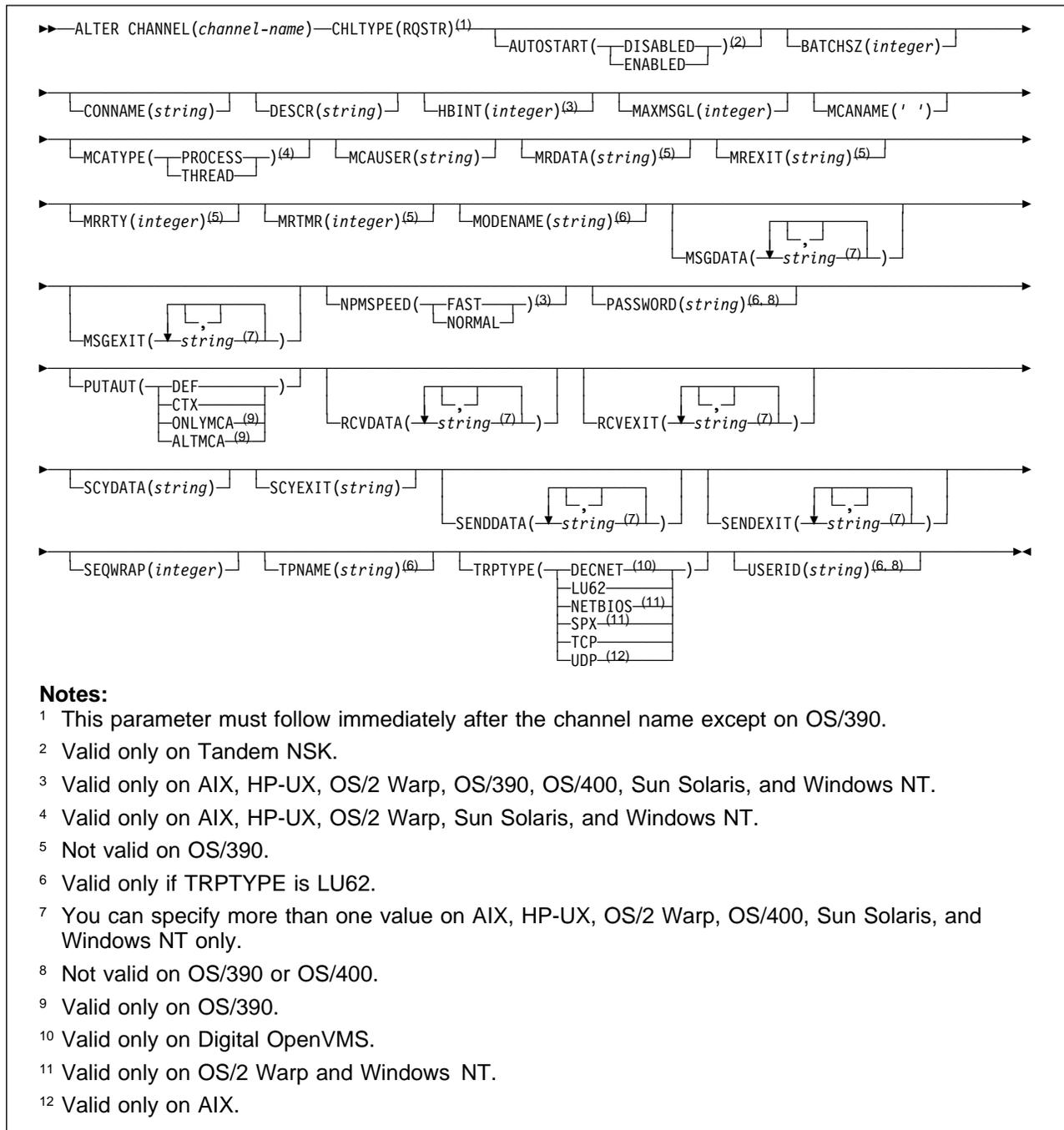
Server channel



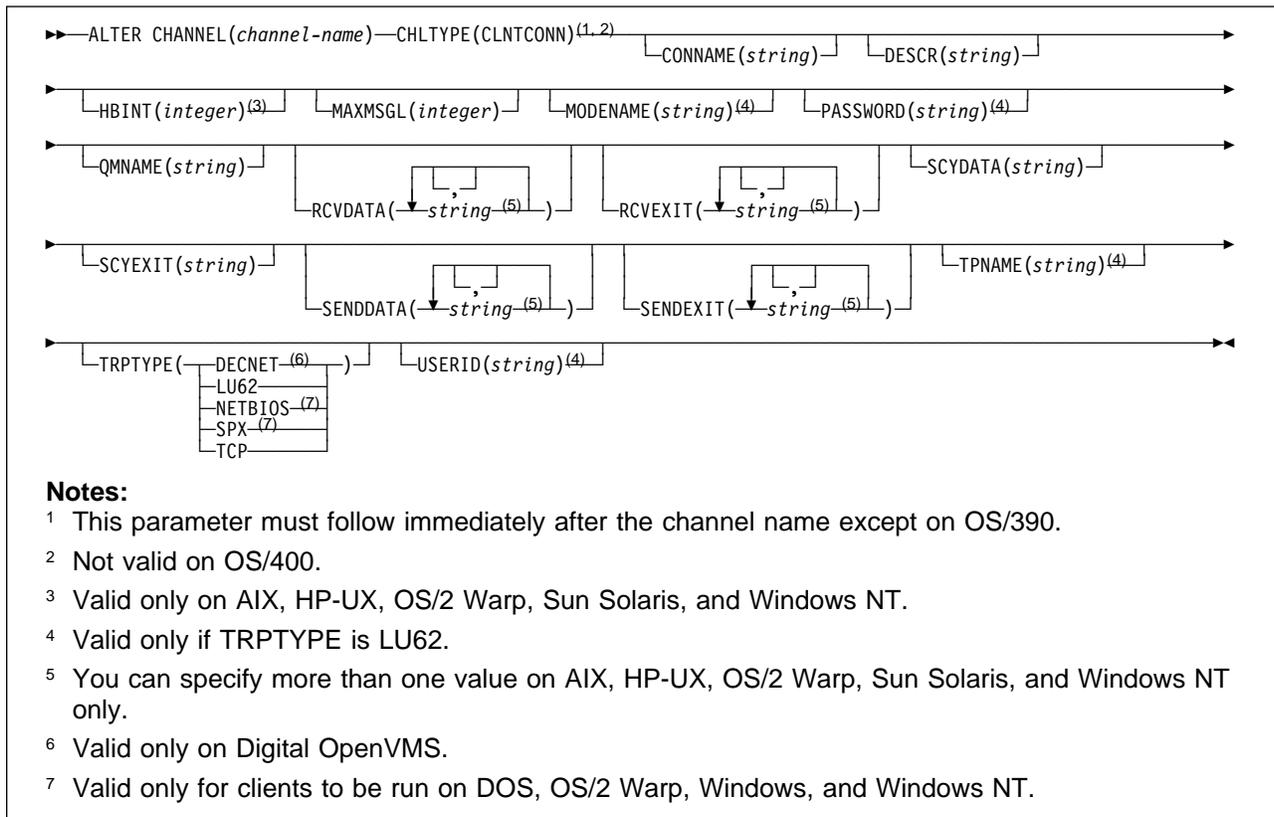
Receiver channel



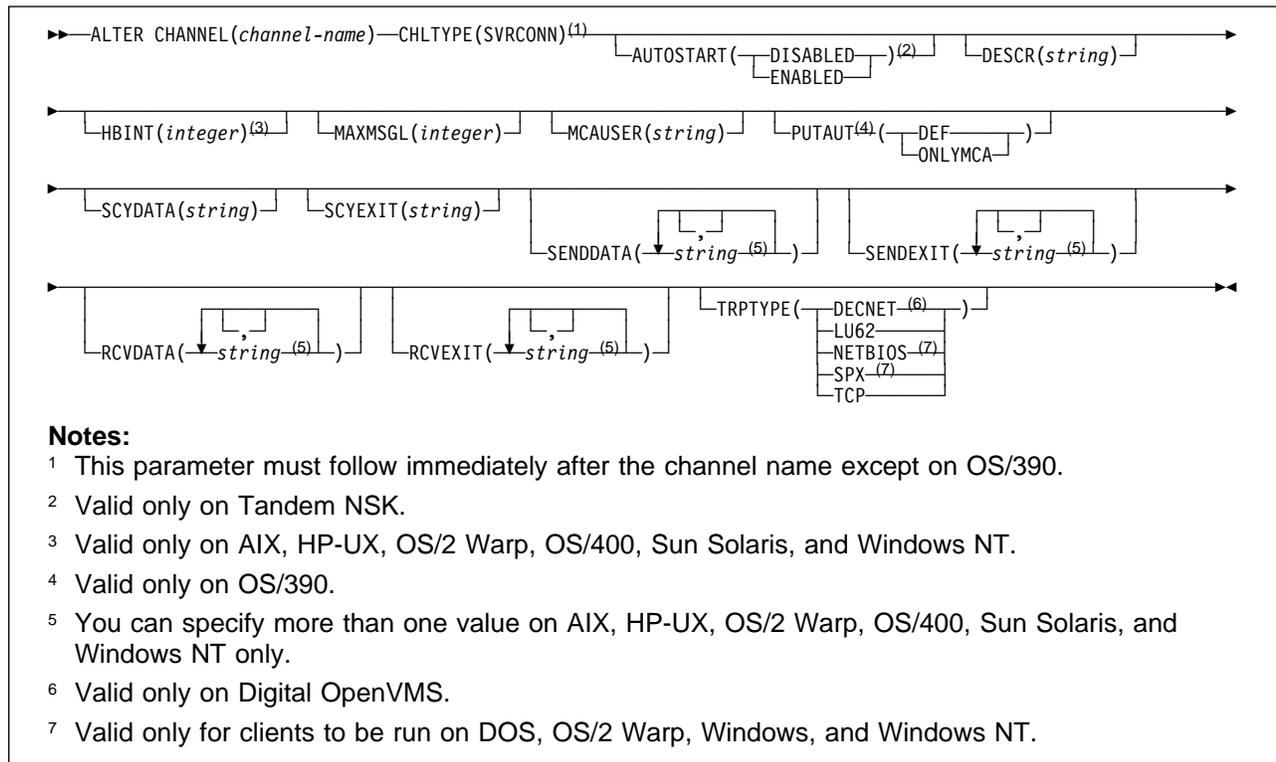
Requester channel



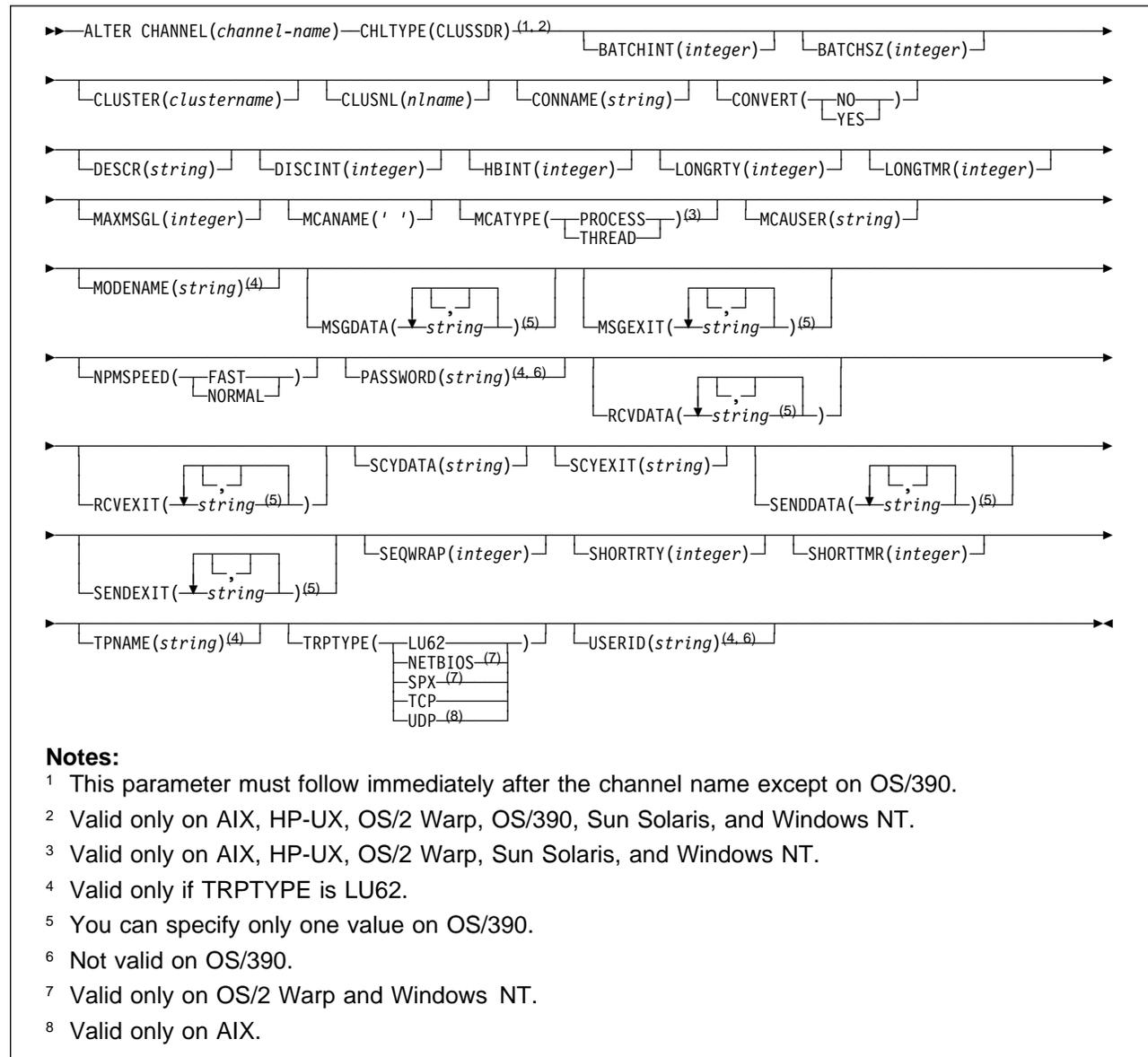
Client-connection channel



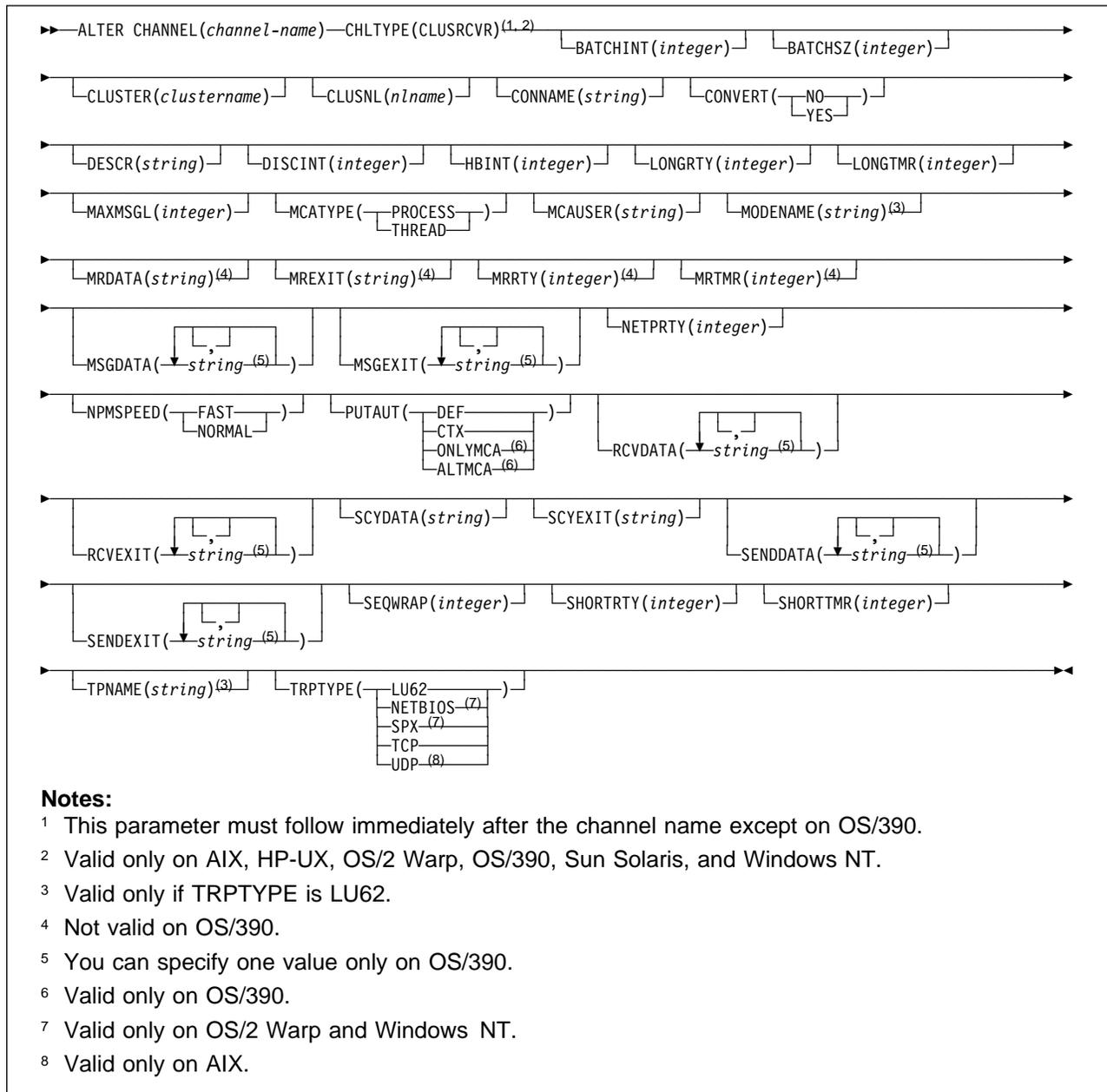
Server-connection channel



Cluster-sender channel



Cluster-receiver 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, SVR, CLUSSDR, or CLUSRCVR. It is valid only on AIX, HP-UX, OS/2 Warp, OS/390, 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 OS/390.

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

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

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 OS/390.

SDR	Sender channel
SVR	Server channel
RCVR	Receiver channel
RQSTR	Requester channel
CLNTCONN	Client-connection channel (not valid on OS/400)
SVRCONN	Server-connection channel
CLUSSDR	Cluster-sender channel (valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT)
CLUSRCVR	Cluster-receiver channel (valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT)

ALTER CHANNEL

CLUSTER(*clustername*)

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQSeries objects.

This parameter is valid only for CLUSSDR and CLUSRCVR channels. Only one of the resultant values of CLUSTER or CLUSNL can be nonblank, the other must be blank.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*nlname*)

The name of the namelist that specifies a list of clusters to which the channel belongs.

This parameter is valid only for CLUSSDR and CLUSRCVR channels. Only one of the resultant values of CLUSTER or CLUSNL can be nonblank, the other must be blank.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CONNAME(*string*)

Connection name.

For cluster-receiver channels it relates to the local queue manager, and for other channels it relates to the target queue manager. (The maximum length is 48 characters on OS/390, and 264 characters on other platforms.)

The value you specify depends on the transport type (TRPTYPE) to be used:

DECnet Phase IV

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

```
CONNAME('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:

```
CONNAME('gateway_node.access_name(tpname)')
```

- On OS/390 there are two forms in which to specify the value:

Logical unit name

The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. This can be specified in one of 3 forms:

Form	Example
luname	IGY12355
luname/TPname	IGY12345/APING
luname/TPname/modename	IGY12345/APINGD/#INTER

For the first form, the TP name and mode name must be specified for the TPNAME and MODENAME attributes; otherwise these attributes must be blank.

Note: For client-connection channels, only the first form is allowed.

Symbolic name

The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The TPNAME and MODENAME attributes must be blank.

Note: For cluster-receiver channels, the side information is on the other queue managers in the cluster. Alternatively, in this case it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

- On OS/2 Warp it is the fully-qualified name of the partner LU, or an LU alias.
- On OS/400, Windows NT, and UNIX systems, this is the name of the CPI-C communications side object or, if the TPNAME is not blank, this is the fully-qualified name of the partner logical unit.

See the information about configuration parameters for an LU 6.2 connection for your platform in the *MQSeries Intercommunication* manual for more information.

- 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('PPPP.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('PPPP.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:

```
CONNNAME('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:

```
CONNNAME('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:

```
CONNNAME('0a0b0c0d.804abcde23a1(5e86)')
```

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

TCP

Either the host name, or the network address of the remote machine (or the local machine for cluster-receiver channels). This can be followed by an optional port number, enclosed in parentheses.

This parameter is required for channels with a channel type (CHLTYPE) of SDR, RQSTR, CLNTCONN, CLUSSDR, and CLUSRCVR. 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.

ALTER CHANNEL

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, SVR, CLUSSDR, or CLUSRCVR.

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, SVR, CLUSSDR, or CLUSRCVR.

HBINT(*integer*)

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

- For a channel type of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR, 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, OS/2 Warp, OS/390, 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, server, or cluster-sender 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).

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, SVR, CLUSSDR, or CLUSRCVR.

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, SVR, CLUSSDR, or CLUSRCVR.

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, RQSTR, CLUSSDR, or CLUSRCVR. It is supported only on AIX, HP/UX, OS/2 Warp, Sun Solaris, and Windows NT. On OS/390 it is supported only for channels with a channel type of CLUSRCVR.

MCAUSER(*string*)

Message channel agent user identifier.

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.

ALTER CHANNEL

The maximum length of *string* is 64 characters on Windows NT and 12 characters on other platforms. On Windows NT, you can optionally qualify a user identifier with the domain name in the following format:

user@domain

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 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, if specified this should be set to the SNA mode name unless the CONNAME contains a side-object name, in which case it should be set to blanks. The actual name is then taken from the CPI-C Communications Side Object, or APPC side information data set.

See the information about configuration parameters for an LU 6.2 connection for your platform in the *MQSeries Intercommunication* manual for more information.

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, RQSTR, or CLUSRCVR. It is not supported on OS/390.

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, RQSTR, or CLUSRCVR. It is not supported on OS/390.

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, RQSTR, or CLUSRCVR. It is not supported on OS/390.

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, RQSTR, or CLUSRCVR. It is not supported on OS/390.

MSGDATA(string)

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. The total length of the field must not exceed 999 characters. 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(string)

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.

ALTER CHANNEL

- 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 OS/390, it is a load module name, maximum length 8 characters (128 characters are allowed for exit names for client-connection channels).

NETPRTY(*integer*)

The priority for the network connection. Distributed queuing chooses the path with the highest priority if there are multiple paths available. The value must be in the range 0 through 9; 0 is the lowest priority.

This parameter is valid only for CLUSRCVR channels.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

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, RQSTR, CLUSSDR, or CLUSRCVR. It is valid only on AIX, HP-UX, OS/2 Warp, OS/390, 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, CLNTCONN, or CLUSSDR. It is not supported on OS/400 and is only supported on OS/390 for client-connection channels.

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

PUTAUT

Specifies which user identifiers should be used to establish authority to put messages to the destination queue (for message channels) or to execute an MQI call (for MQI channels).

DEF The default user ID is used. On OS/390 this might involve using both the user ID received from the network and that derived from MCAUSER.

CTX	The user ID from the <i>UserIdentifier</i> field of the message descriptor is used. On OS/390 this might involve also using the user ID received from the network or that derived from MCAUSER, or both.
ONLYMCA	The default user ID is used. Any user ID received from the network is not used. This value is supported only on OS/390.
ALTMCA	The user ID from the <i>UserIdentifier</i> field of the message descriptor is used. Any user ID received from the network is not used. This value is supported only on OS/390.

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR, RQSTR, CLUSRCVR, or SVRCONN. CTX and ALTMCA are not valid for SVRCONN channels.

QMNAME(*string*)

Queue manager name.

For channels with a channel type (CHLTYPE) of CLNTCONN, this 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(*string*)

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. The total length of the field must not exceed 999 characters. 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(*string*)

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(*string*)

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

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

ALTER CHANNEL

SCYEXIT(*string*)

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(*string*)

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. The total length of the field must not exceed 999 characters. 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(*string*)

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, RQSTR, CLUSSDR, or CLUSRCVR.

SHORTRTY(*integer*)

The maximum number of attempts that are made by a sender, server, or cluster-sender channel to connect to the remote queue manager, at intervals specified by SHORTTMR, before the (normally longer) LONGRTY and LONGTMR are used.

Retry attempts are made if the channel fails to connect initially (whether it is started automatically by the channel initiator or by an explicit command), and also if the connection fails after the channel has successfully connected. However, if the cause of the failure is such that 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, SVR, CLUSSDR, or CLUSRCVR.

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, SVR, CLUSSDR, or CLUSRCVR.

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 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, this should be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it should be set to blanks. The actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set.

See the information about configuration parameters for an LU 6.2 connection for your platform in the *MQSeries Intercommunication* manual for more information.

On Windows NT SNA Server, and in the side object on OS/390, the TPNAME is wrapped to upper case.

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

ALTER CHANNEL

TRPTYPE

Transport type to be used.

This is not required on AIX, HP-UX, OS/2 Warp, OS/390, 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 OS/390, 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 OS/390 for defining client-connection channels that will connect to servers on the platforms supporting NetBIOS)

SPX Sequenced packet exchange (supported only on OS/2 Warp, Windows, Windows NT, and DOS; it also applies to OS/390 for defining client-connection channels that will connect to servers on the platforms supporting SPX)

TCP Transmission Control Protocol - part of the TCP/IP protocol suite

UDP User Datagram Protocol - part of the TCP/IP protocol suite (supported only on AIX)

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, CLNTCONN, or CLUSSDR. It is not supported on OS/400 and is only supported on OS/390 for CLNTCONN 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.

ALTER NAMELIST

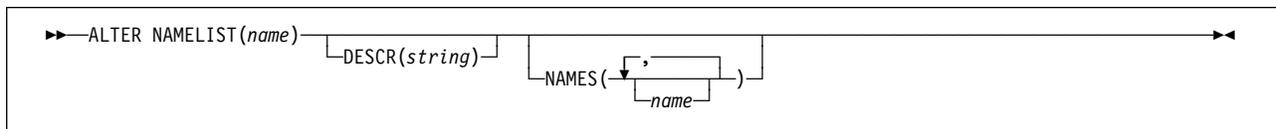
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use ALTER NAMELIST to alter a list of names. This is most commonly a list of cluster names or queue names.

Notes:

1. On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.

Synonym: ALT NL



Keyword and parameter descriptions

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(*name, ...*)

List of names. This is optional.

The names can be of any type but must conform to the rules for naming MQSeries objects, with a maximum length of 48 characters.

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

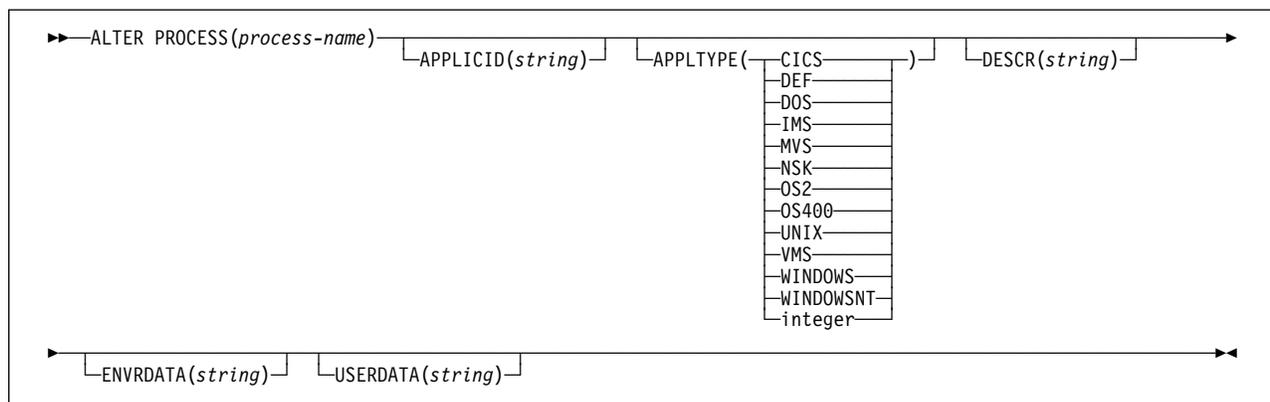
ALTER PROCESS

ALTER PROCESS

Digital OpenVMS	OS/390	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 OS/390, 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:

CICS	Represents a CICS transaction.
DOS	Represents a DOS application.
IMS	Represents an IMS transaction.
MVS	Represents an OS/390 application (batch or TSO).
NSK	Represents a Tandem NSK application.
OS2	Represents an OS/2 Warp application.
OS400	Represents an OS/400 application.
UNIX	Represents a UNIX application.
VMS	Represents a Digital OpenVMS application.

WINDOWS	Represents a Windows application.
WINDOWSNT	Represents a Windows NT application.
integer	User-defined application type in the range 65 536 through 999 999 999.
DEF	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 OS/390, CICS (default), DOS, IMS, MVS, OS2, UNIX, WINDOWS, WINDOWSNT, 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 "Triggering channels" in the *MQSeries Intercommunication* manual for information about what these need as APPLICID.

DEFPSIST

Specifies the message persistence to be used when applications specify the MQPER_PERSISTENCE_AS_Q_DEF option.

NO Messages on this queue are lost across a restart of the queue manager. 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**CLUSTER**(*clustername*)

The name of the cluster to which the queue belongs. The maximum length is 48 characters conforming to the rules for naming MQSeries objects. Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*nlname*)

The name of the namelist that specifies a list of clusters to which the queue belongs. Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

DEFBIND

Specifies the binding to be used when the application specifies MQOO_BIND_AS_Q_DEF on the **MQOPEN** call, and the queue is a cluster queue.

OPEN The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

NOTFIXED The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using **MQPUT**, and to change that selection subsequently should the need arise.

The **MQPUT1** call always behaves as if NOTFIXED had been specified.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

ALTER QALIAS

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 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 model queue)
- A local definition of a remote queue
- A cluster queue

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

Usage notes

1. `DEFINE QALIAS(otherqname) TARGQ(aliasqueue) CLUSTER(c)` has the effect of advertising queue *aliasqueue* by the name *otherqname*.
2. `DEFINE QALIAS(otherqname) TARGQ(aliasqueue)` has the effect of allowing a queue advertised by the name *otherqname* to be used on this queue manager by the name *aliasqueue*, provided that *aliasqueue* is also defined.

ALTER QLOCAL

Digital OpenVMS	OS/390	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

```

▶▶ ALTER QLOCAL (q-name) [FORCE] [common q attrs] [local q attrs]
Common q attrs:
  [DEFPRTY(integer)] [DEFPSIST(NO|YES)] [DESCR(string)] [PUT(ENABLED|DISABLED)]
Local q attrs:
  [BOQNAME(string)(1)] [BOTHRESH(integer)(1)] [CLUSTER(clustername)(2)] [CLUSNL(nlname)(2)]
  [DEFBIND(OPEN|NOTFIXED)(2)] [DEFSOPT(EXCL|SHARED)] [DISTL(NO|YES)(3)]
  [GET(ENABLED|DISABLED)] [INDXTYPE(CORRELID|MSGID|MSGTOKEN|NONE)(4)] [INITQ(string)] [MAXDEPTH(integer)]
  [MAXMSGL(integer)] [MSGDLVSQ(PRIORITY|FIFO)] [NOHARDENBO(1)|HARDENBO] [NOSHARE|SHARE]
  [NOTRIGGER|TRIGGER] [PROCESS(string)] [QDEPTHHI(integer)] [QDEPTHLO(integer)]
  [QDPHIEV(ENABLED|DISABLED)] [QDPLOEV(ENABLED|DISABLED)] [QDPMAEV(ENABLED|DISABLED)]
  [QSVCI EV(NONE|HIGH|OK)] [QSVCI NT(integer)] [RETINTVL(integer)] [SCOPE(QMGR|CELL)(5)]
  [STGCLASS(string)(4)] [TRIGDATA(string)] [TRIGDPH(integer)] [TRIGMPRI(integer)]
  [TRIGTYPE(FIRST|EVERY|DEPTH|NONE)] [USAGE(NORMAL|XMITQ)]
Notes:
1 Ignored on OS/400.
2 Valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
3 Valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
4 Valid only on OS/390.
5 Valid only on Digital OpenVMS, OS/2 Warp, UNIX systems, and Windows NT.

```

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.

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

Specifies the message persistence to be used when applications specify the MQPER_PERSISTENCE_AS_Q_DEF option.

NO Messages on this queue are lost across a restart of the queue manager. 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.

CLUSTER(*clustername*)

The name of the cluster to which the queue belongs. The maximum length is 48 characters conforming to the rules for naming MQSeries objects. Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter cannot be set for dynamic, transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx, or SYSTEM.COMMAND.xx queues.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*nlname*)

The name of the namelist that specifies a list of clusters to which the queue belongs. Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter cannot be set for dynamic, transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx, or SYSTEM.COMMAND.xx queues.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

DEFBIND

Specifies the binding to be used when the application specifies MQOO_BIND_AS_Q_DEF on the **MQOPEN** call, and the queue is a cluster queue.

OPEN The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

NOTFIXED The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using **MQPUT**, and to change that selection subsequently should the need arise.

The **MQPUT1** call always behaves as if NOTFIXED had been specified.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

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

ALTER QLOCAL

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.

MSGTOKEN An index of message tokens is maintained. Use this when the queue is a WLM managed queue that you are using with the Workload Manager functions of OS/390.

Note: If the queue is a transmission queue or a temporary-dynamic queue, you cannot set **INDXTYPE** to **MSGTOKEN**.

The **INDXTYPE** attribute can be changed to **NONE**, **MSGID**, or **CORRELID** 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 can be changed to **MSGTOKEN** only when there are no messages on the queue. If you attempt to change this attribute to **MSGTOKEN** while there are messages on the queue, the command fails.

This attribute is supported only on OS/390. 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 OS/390
- 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.

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.

ALTER QLOCAL

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 “Queue Depth High” 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 “Queue Depth Low” 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 “Queue Full” 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 “Queue Service Interval High” 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

ALTER QLOCAL

the queue is no longer needed. The CRDATE and CRTIME can be displayed using DISPLAY QUEUE on page 168.

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 OS/390. See the *MQSeries for OS/390 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 164 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.

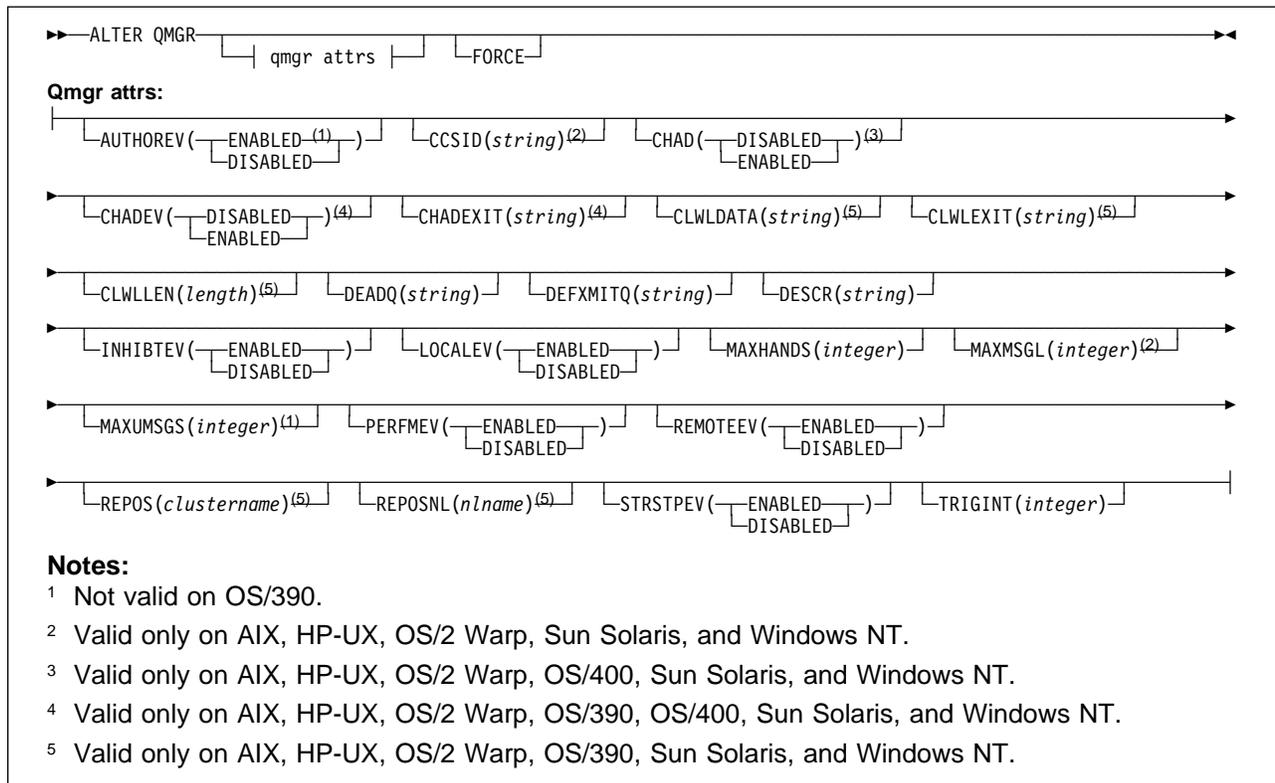
If you specify this option, do not specify values for CLUSTER and CLUSNL, and do not specify INDXTYPE(MSGTOKEN).

ALTER QMGR

Digital OpenVMS	OS/390	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



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 OS/390.

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

CCSID(*string*)

The coded character set identifier for the queue manager. The CCSID is the identifier used with all character string fields defined by the API. It does not apply to application data carried in the text of messages unless the CCSID in the message descriptor is set to the value MQCCSI_Q_MGR when the message is put to a queue.

Specify a value in the range 1 through 65 535. The CCSID must specify a value that is defined for use on your platform, and use a character set that is appropriate to the platform.

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

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.

Cluster-sender channels can always be defined automatically, regardless of the setting of this parameter.

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/390, OS/400, Sun Solaris, and Windows NT. On OS/390 it applies only to cluster-sender and cluster-receiver channels.

CHADEXIT(*string*)

Auto-definition exit name.

If this name is nonblank, the exit is called when an inbound request for an undefined receiver, server-connection, or cluster-sender channel is received. It is also called when starting a cluster-receiver channel.

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

- On 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.

ALTER QMGR

- On AIX, HP-UX, and Sun Solaris, it is of the form *libraryname(functionname)*. The maximum length of the string is 128 characters.
- On OS/390, it is a load module name, maximum length 8 characters.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT. On OS/390, it applies only to cluster-sender and cluster-receiver channels.

CLWLDATA(*string*)

Cluster workload exit data (maximum length 32 characters).

This is passed to the cluster workload exit when it is called.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLWLEXIT(*string*)

Cluster workload exit name.

If this name is nonblank, the exit is called when a message is put to a cluster queue. The format and maximum length of the name depends on the environment:

- On UNIX systems, it is of the form *libraryname(functionname)*. The maximum length is 128 characters.
- On OS/2 Warp and Windows NT, it is of the form *dllname(functionname)*, where *dllname* is specified without the suffix (".DLL"). The maximum length is 128 characters.
- On OS/390, it is a load module name, maximum length 8 characters.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLWLLEN(*length*)

The maximum number of bytes of message data that is passed to the cluster workload exit.

Specify a value between zero and 4 MB for OS/390, or between zero and 999 999 999 on other platforms. The initial default value is 100.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, 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 OS/390. 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.

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.

ALTER QMGR

REPOS(*clustername*)

The name of a cluster for which this queue manager is to provide a repository manager service.
The maximum length is 48 characters conforming to the rules for naming MQSeries objects.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

REPOSNL(*nlname*)

The name of a namelist of clusters for which this queue manager is to provide a repository manager service.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

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 108 for details). In this case trigger messages are normally generated only when a suitable message arrives on the queue, and the queue was previously empty. Under certain circumstances, however an additional trigger message can be generated with FIRST triggering even if the queue was not empty. These additional trigger messages are not generated more often than every TRIGINT milliseconds. See "Special case of trigger type FIRST" in the *MQSeries Application Programming Guide* for more information.

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

Usage notes

1. No more than one of the resultant values of REPOS and REPOSNL can be nonblank.
2. If both REPOS and REPOSNL are blank, or REPOS is blank and the namelist specified by REPOSNL is empty, this queue manager does not have a full repository, but might be a client of other repository services that are defined in the cluster.

ALTER QMODEL

Digital OpenVMS	OS/390	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

```

▶ ALTER QMODEL(q-name)
┌── common q attrs ─┘ ┌── local q attrs ─┘ ┌── model q attr ─┘
┌── DEFPRTY(integer) ─┘ ┌── DEFPSIST(┌── NO ─┘) ┌── DESCR(string) ─┘ ┌── PUT(┌── ENABLED ─┘) ─┘
└── YES ─┘ └── DISABLED ─┘
Local q attrs:
┌── BOQNAME(string)(1) ─┘ ┌── BOTHRESH(integer)(1) ─┘ ┌── DEFSOPT(┌── EXCL ─┘) ─┘ ┌── DISTL(┌── NO ─┘)(2) ─┘
└── SHARED ─┘ └── YES ─┘
┌── GET(┌── ENABLED ─┘) ─┘ ┌── INDXTYPE(┌── CORRELID ─┘)(3) ─┘ ┌── INITQ(string) ─┘ ┌── MAXDEPTH(integer) ─┘
└── DISABLED ─┘ └── MSGID ─┘
└── MSGTOKEN ─┘
└── NONE ─┘
┌── MAXMSGL(integer) ─┘ ┌── MSGDLVSQ(┌── PRIORITY ─┘) ─┘ ┌── NOHARDENBO ─┘(1) ─┘ ┌── NOSHARE ─┘
└── FIFO ─┘ └── HARDENBO ─┘ └── SHARE ─┘
┌── NOTRIGGER ─┘ ┌── PROCESS(string) ─┘ ┌── QDEPTHHI(integer) ─┘ ┌── QDEPTHLO(integer) ─┘
└── TRIGGER ─┘
┌── QDPHIEV(┌── ENABLED ─┘) ─┘ ┌── QDPLOEV(┌── ENABLED ─┘) ─┘ ┌── QDPMAXEV(┌── ENABLED ─┘) ─┘
└── DISABLED ─┘ └── DISABLED ─┘ └── DISABLED ─┘
┌── QSVCIEV(┌── NONE ─┘) ─┘ ┌── QSVCINT(integer) ─┘ ┌── RETINTVL(integer) ─┘ ┌── STGCLASS(string)(3) ─┘
└── HIGH ─┘
└── OK ─┘
┌── TRIGDATA(string) ─┘ ┌── TRIGDPH(integer) ─┘ ┌── TRIGMPRI(integer) ─┘ ┌── TRIGTYPE(┌── FIRST ─┘) ─┘
└── EVERY ─┘
└── DEPTH ─┘
└── NONE ─┘
┌── USAGE(┌── NORMAL ─┘) ─┘
└── XMITQ ─┘
Model q attr:
┌── DEFTYPE(┌── TEMPDYN ─┘) ─┘
└── PERMDYN ─┘
Notes:
1 Ignored on OS/400.
2 Valid only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT.
3 Valid only on OS/390.

```

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

Specifies the message persistence to be used when applications specify the MQPER_PERSISTENCE_AS_Q_DEF option.

NO Messages on this queue are lost across a restart of the queue manager. 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.

MSGTOKEN An index of message tokens is maintained. Use this when the queue is a WLM managed queue that you are using with the Workload Manager functions of OS/390.

Note: If the queue is a transmission queue or a temporary-dynamic queue, you cannot set **INDXTYPE** to **MSGTOKEN**.

The **INDXTYPE** attribute can be changed to **NONE**, **MSGID**, or **CORRELID** 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.

ALTER QMODEL

This attribute can be changed to MSGTOKEN only when there are no messages on the queue. If you attempt to change this attribute to MSGTOKEN while there are messages on the queue, the command fails.

This attribute is supported only on OS/390. 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 OS/390
- 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.

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.

ALTER QMODEL

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 “Queue Depth High” 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 “Queue Depth Low” 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 “Queue Full” in the *MQSeries Programmable System Management* manual for more details.

ENABLED Queue Full events are generated

DISABLED Queue Full events are not generated

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 “Queue Service Interval High” 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 168.

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.

STGLCLASS(*string*)

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

This attribute is valid only on OS/390. See the *MQSeries for OS/390 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 164 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.

ALTER QMODEL

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.

If you specify this option, do not specify INDXTYPE(MSGTOKEN).

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.

If you specify this option, do not specify INDXTYPE(MSGTOKEN).

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

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

Specifies the message persistence to be used when applications specify the MQPER_PERSISTENCE_AS_Q_DEF option.

NO Messages on this queue are lost across a restart of the queue manager. 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

CLUSTER(*clustername*)

The name of the cluster to which the queue belongs. The maximum length is 48 characters conforming to the rules for naming MQSeries objects. Changes to this parameter do not affect instances of the queue that are already open. Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*nlname*)

The name of the namelist that specifies a list of clusters to which the queue belongs. Changes to this parameter do not affect instances of the queue that are already open. Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

DEFBIND

Specifies the binding to be used when the application specifies MQOO_BIND_AS_Q_DEF on the **MQOPEN** call, and the queue is a cluster queue.

OPEN The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

NOTFIXED The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using **MQPUT**, and to change that selection subsequently should the need arise.

The **MQPUT1** call always behaves as if NOTFIXED had been specified.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

RNAME(*string*)

Name of remote queue. This is the local name of the queue as defined on the queue manager specified by RQMNAME.

- 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.

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).

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 MQSeries object 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.

ALTER QREMOTE

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.

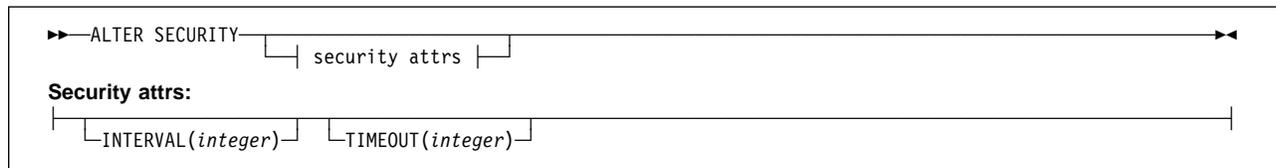
It is also ignored if the definition is used as a reply-to queue alias definition.

ALTER SECURITY

Digital OpenVMS	OS/390	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.

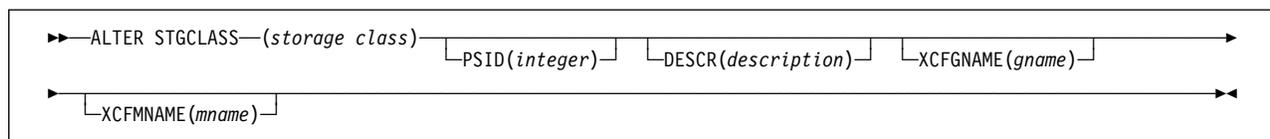
ALTER STGCLASS

ALTER STGCLASS

Digital OpenVMS	OS/390	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.

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).

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).

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.

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(' ')
```

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

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

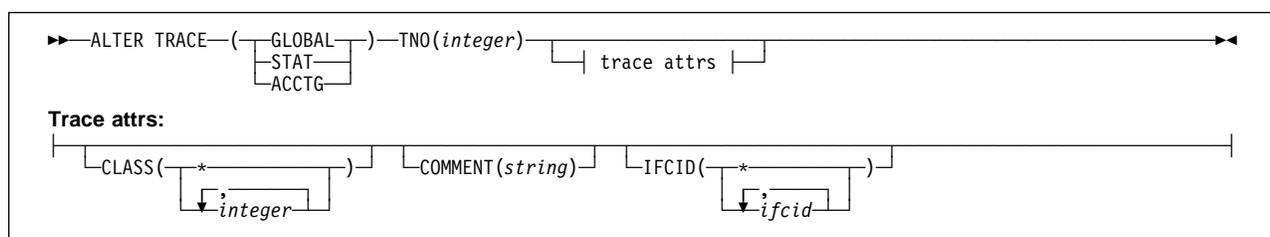
ALTER TRACE

Digital OpenVMS	OS/390	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 201.

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 201 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)

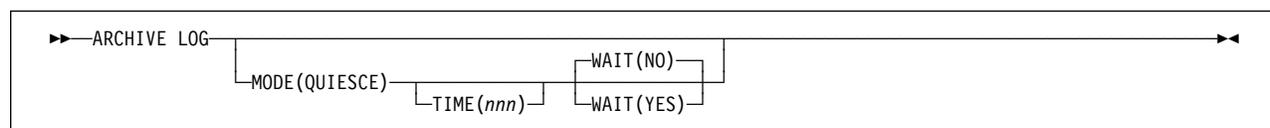
The events to be traced. This specifies the optional IFCIDs to activate. All IFCIDs and classes specified are activated for the trace type specified.

ARCHIVE LOG

Digital OpenVMS	OS/390	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.

ARCHIVE LOG

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.

Usage 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

Digital OpenVMS	OS/390	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

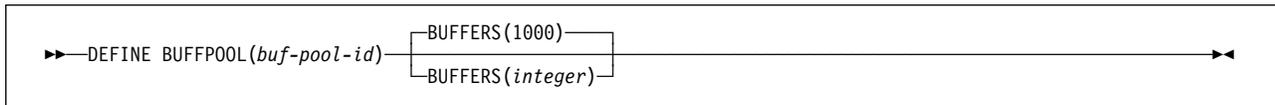
DEFINE BUFFPOOL

Digital OpenVMS	OS/390	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.

Note: 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

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

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

Notes:

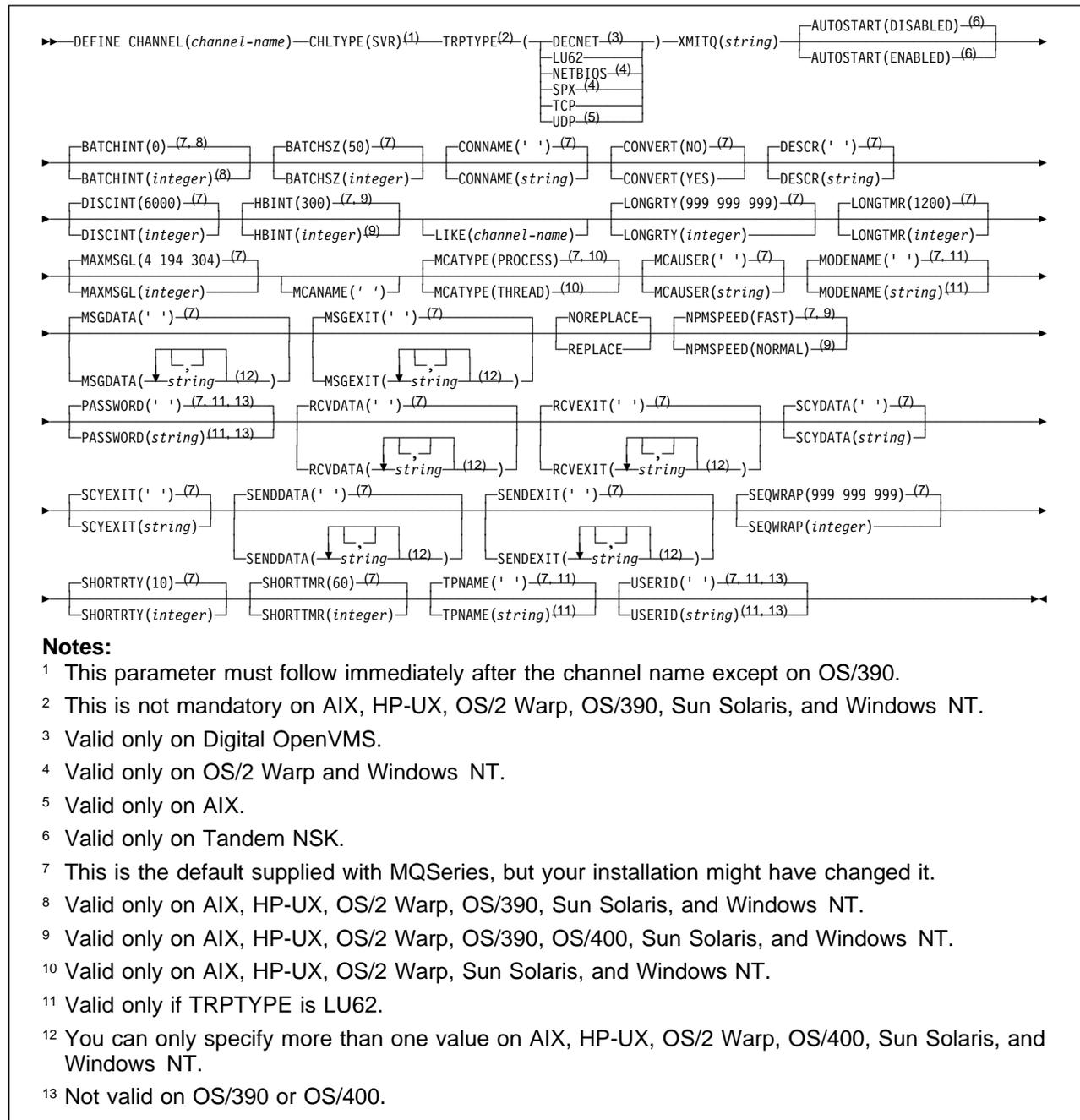
1. On OS/390, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 13.
2. For cluster-sender channels, you can only specify the REPLACE option for channels that have been created manually.

Synonym: DEF CHL

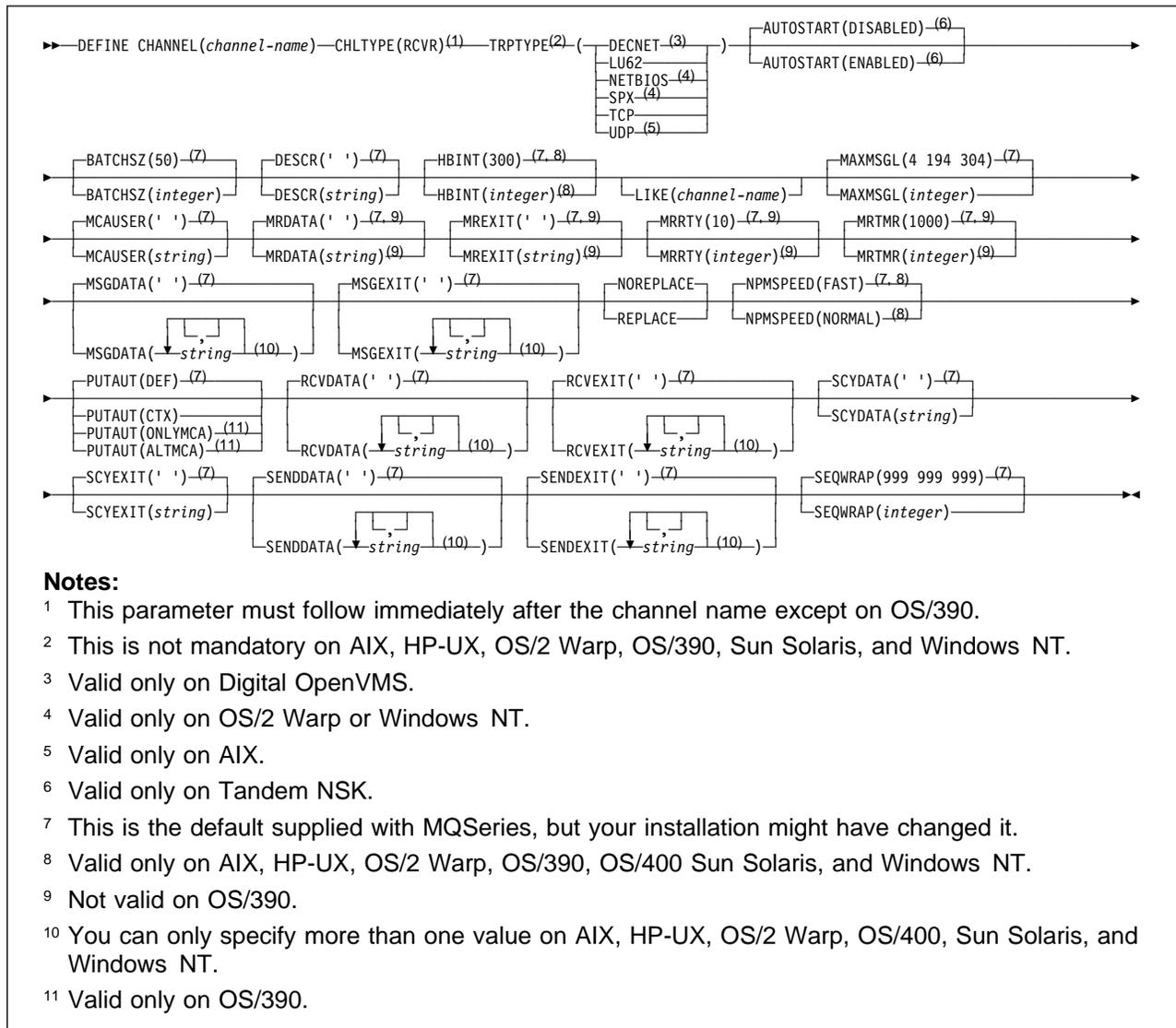
There is a separate syntax diagram for each type of channel:

- “Sender channel” on page 76
- “Server channel” on page 77
- “Receiver channel” on page 78
- “Requester channel” on page 79
- “Client-connection channel” on page 80
- “Server-connection channel” on page 81
- “Cluster-sender channel” on page 82
- “Cluster-receiver channel” on page 83

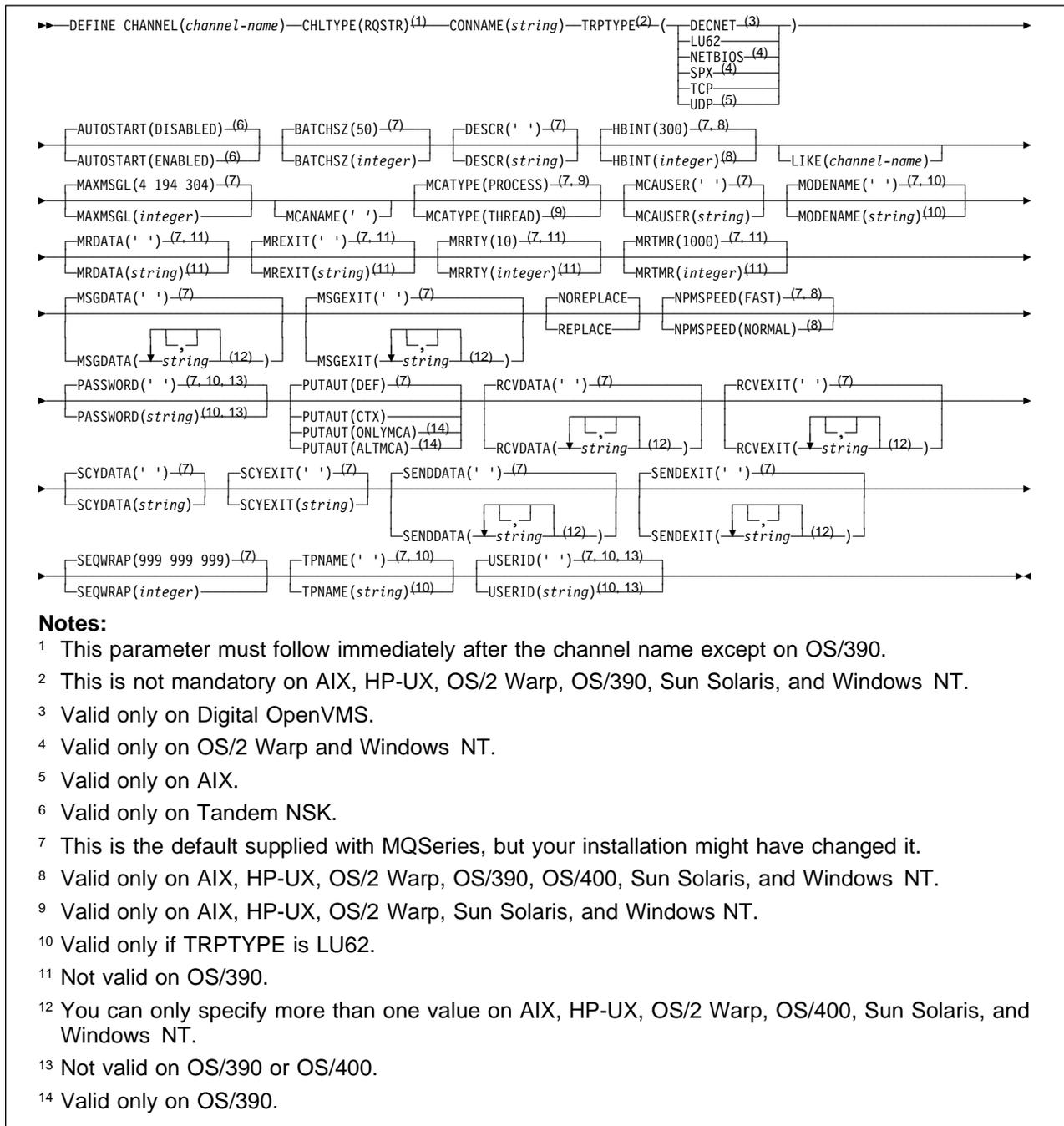
Server channel



Receiver channel

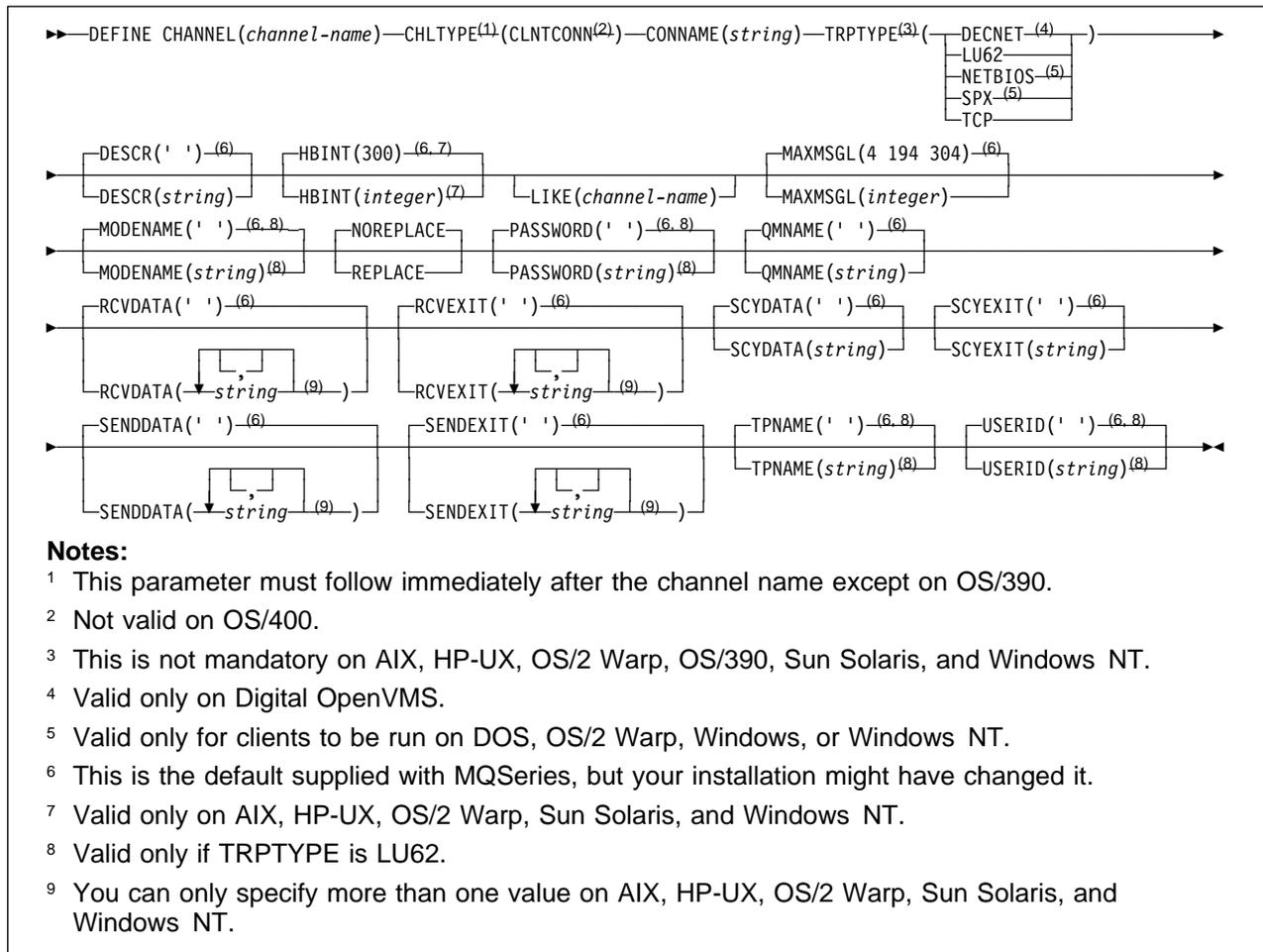


Requester channel

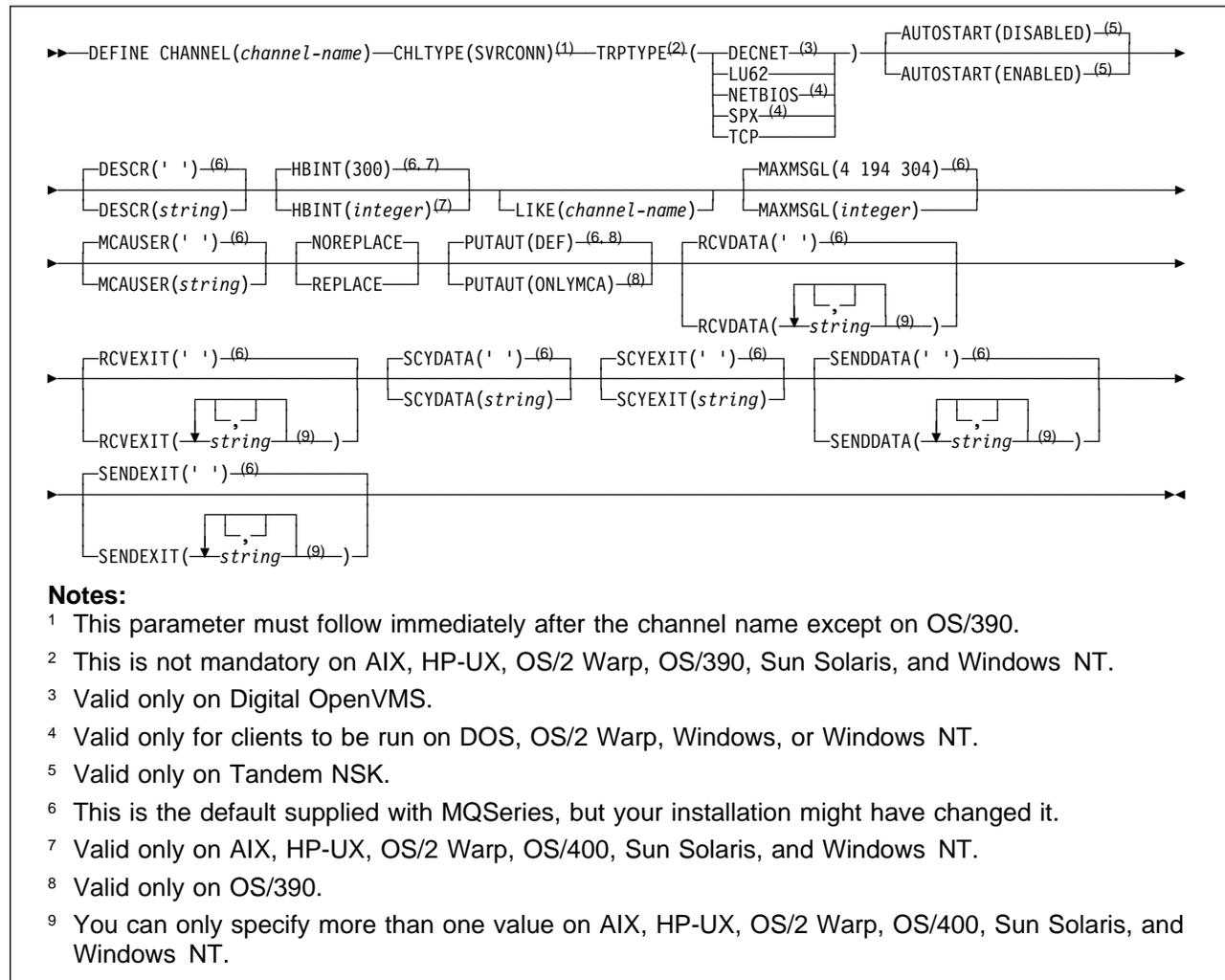


DEFINE CHANNEL

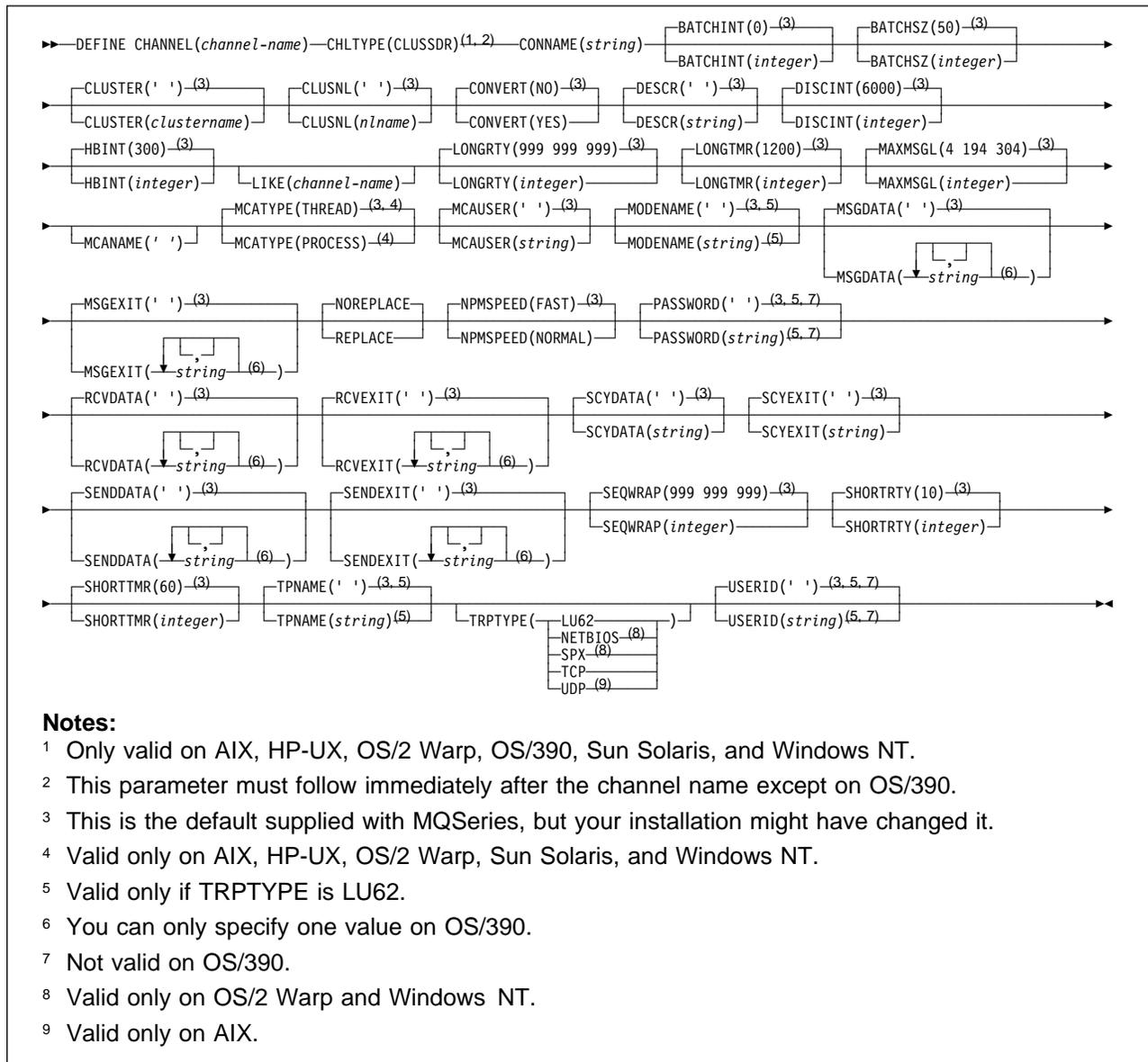
Client-connection channel



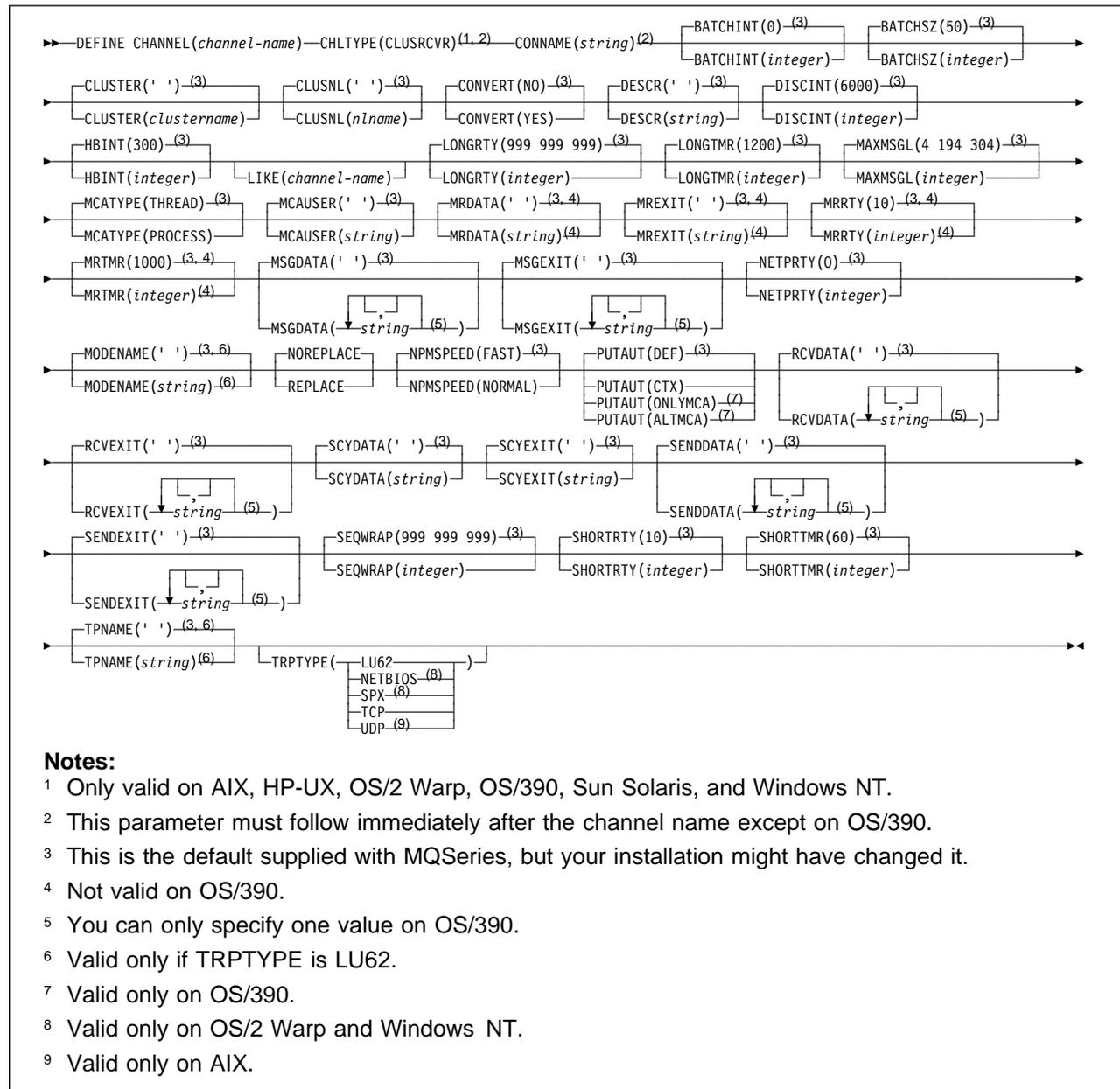
Server-connection channel



Cluster-sender channel



Cluster-receiver channel



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 OS/390, 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.

DEFINE CHANNEL

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, SVR, CLUSSDR, or CLUSRCVR. It is valid only on AIX, HP-UX, OS/2 Warp, OS/390, 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 OS/390.

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

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 OS/390.

SDR	Sender channel
SVR	Server channel
RCVR	Receiver channel
RQSTR	Requester channel
CLNTCONN	Client-connection channel (not valid on OS/400)
SVRCONN	Server-connection channel
CLUSSDR	Cluster-sender channel (valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT)
CLUSRCVR	Cluster-receiver channel (valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT)

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

CLUSTER(*clustername*)

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQSeries objects.

This parameter is valid only for CLUSSDR and CLUSRCVR channels. Only one of the resultant values of CLUSTER or CLUSNL can be nonblank, the other must be blank.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*nlname*)

The name of the namelist that specifies a list of clusters to which the channel belongs.

This parameter is valid only for CLUSSDR and CLUSRCVR channels. Only one of the resultant values of CLUSTER or CLUSNL can be nonblank, the other must be blank.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CONNAME(*string*)

Connection name.

For cluster-receiver channels it relates to the local queue manager, and for other channels it relates to the target queue manager. (The maximum length is 48 characters on OS/390, and 264 characters on other platforms.)

The value you specify depends on the transport type (TRPTYPE) to be used:

DECnet Phase IV

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

CONNAME('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:

CONNAME('gateway_node.access_name(tpname)')

- On OS/390 there are two forms in which to specify the value:

Logical unit name

The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. This can be specified in one of 3 forms:

Form	Example
luname	IGY12355
luname/TPname	IGY12345/APING
luname/TPname/modename	IGY12345/APINGD/#INTER

For the first form, the TP name and mode name must be specified for the TPNAME and MODENAME attributes; otherwise these attributes must be blank.

Note: For client-connection channels, only the first form is allowed.

Symbolic name

The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The TPNAME and MODENAME attributes must be blank.

Note: For cluster-receiver channels, the side information is on the other queue managers in the cluster. Alternatively, in this case it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

DEFINE CHANNEL

- On OS/2 Warp it is the fully-qualified name of the partner LU, or an LU alias.
- On OS/400, Windows NT, and UNIX systems, this is the name of the CPI-C communications side object or, if the TPNAME is not blank, this is the fully-qualified name of the partner logical unit.

See the information about configuration parameters for an LU 6.2 connection for your platform in the *MQSeries Intercommunication* manual for more information.

- 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('PPPP.LOCALLU.REMOTELU')
```

- If SNAX is used:
 - 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('PPPP.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:

```
CONNNAME('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:

```
CONNNAME('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:

```
CONNNAME('0a0b0c0d.804abcde23a1(5e86)')
```

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

TCP

Either the host name, or the network address of the remote machine (or the local machine for cluster-receiver channels). This can be followed by an optional port number, enclosed in parentheses.

This parameter is required for channels with a channel type (CHLTYPE) of SDR, RQSTR, CLNTCONN, CLUSSDR, and CLUSRCVR. 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, SVR, CLUSSDR, or CLUSRCVR.

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, SVR, CLUSSDR, or CLUSRCVR.

HBINT(*integer*)

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

- For a channel type of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR, 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, OS/2 Warp, OS/390, 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.

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

DEFINE CHANNEL

SYSTEM.DEF.SERVER	Server channel
SYSTEM.DEF.RECEIVER	Receiver channel
SYSTEM.DEF.REQUESTER	Requester channel
SYSTEM.DEF.SVRCONN	Server-connection channel
SYSTEM.DEF.CLNTCONN	Client-connection channel
SYSTEM.DEF.CLUSSDR	Cluster-sender channel
SYSTEM.DEF.CLUSRCVR	Cluster-receiver channel

This 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, server, or cluster-sender 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).

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**, **SVR**, **CLUSSDR**, or **CLUSRCVR**.

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 are treated as 999 999.

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

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, RQSTR, CLUSSDR, or CLUSRCVR. It is supported only on AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT. On OS/390 it is supported only for channels with a channel type of CLUSRCVR.

MCAUSER(string)

Message channel agent user identifier.

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.

The maximum length of *string* is 64 characters on Windows NT and 12 characters on other platforms. On Windows NT, you can optionally qualify a user identifier with the domain name in the following format:

user@domain

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 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, if specified this should be set to the SNA mode name unless the CONNAME contains a side-object name, in which case it should be set to blanks. The actual name is then taken from the CPI-C Communications Side Object, or APPC side information data set.

See the information about configuration parameters for an LU 6.2 connection for your platform in the *MQSeries Intercommunication* manual for more information.

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, RQSTR, or CLUSRCVR. It is not supported on OS/390.

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, RQSTR, or CLUSRCVR. It is not supported on OS/390.

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, RQSTR, or CLUSRCVR. It is not supported on OS/390.

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, RQSTR, or CLUSRCVR. It is not supported on OS/390.

MSGDATA(*string*)

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. The total length of the field must not exceed 999 characters. 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(*string*)

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

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 OS/390, it is a load module name, maximum length 8 characters (128 characters are allowed for exit names for client-connection channels).

NETPRTY(*integer*)

The priority for the network connection. Distributed queuing chooses the path with the highest priority if there are multiple paths available. The value must be in the range 0 through 9; 0 is the lowest priority.

This parameter is valid only for CLUSRCVR channels.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390 Sun Solaris, and Windows NT.

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.

DEFINE CHANNEL

This parameter is valid only for channels with a CHLTYPE of SDR, SVR, RCVR, RQSTR, CLUSSDR, or CLUSRCVR. It is valid only on AIX, HP-UX, OS/2 Warp, OS/390, 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, CLNTCONN, or CLUSSDR. It is not supported on OS/400 and is only supported on OS/390 for client-connection channels.

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

PUTAUT

Specifies which user identifiers should be used to establish authority to put messages to the destination queue (for messages channels) or to execute an MQI call (for MQI channels).

DEF The default user ID is used. On OS/390 this might involve using both the user ID received from the network and that derived from MCAUSER.

CTX The user ID from the *UserIdentifier* field of the message descriptor is used. On OS/390 this might involve also using the user ID received from the network or that derived from MCAUSER, or both.

ONLYMCA The default user ID is used. Any user ID received from the network is not used. This value is supported only on OS/390.

ALTMCA The user ID from the *UserIdentifier* field of the message descriptor is used. Any user ID received from the network is not used. This value is supported only on OS/390.

This parameter is valid only for channels with a channel type (CHLTYPE) of RCVR, RQSTR, CLUSRCVR, or SVRCONN. CTX and ALTMCA are not valid for SVRCONN channels.

QMNAME(*string*)

Queue manager name.

For channels with a channel type (CHLTYPE) of CLNTCONN, this 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(*string*)

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. The total length of the field must not exceed 999 characters. 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(*string*)

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(*string*)

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

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

SCYEXIT(*string*)

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(*string*)

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. The total length of the field must not exceed 999 characters. 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.

DEFINE CHANNEL

SENDEXIT(*string*)

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, RQSTR, CLUSSDR, or CLUSRCVR.

SHORTRTY(*integer*)

The maximum number of attempts that are made by a sender, server, or cluster-sender channel to connect to the remote queue manager, at intervals specified by SHORTTMR, before the (normally longer) LONGRTY and LONGTMR are used.

Retry attempts are made if the channel fails to connect initially (whether it is started automatically by the channel initiator or by an explicit command), and also if the connection fails after the channel has successfully connected. However, if the cause of the failure is such that 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, SVR, CLUSSDR, or CLUSRCVR.

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, SVR, CLUSSDR, or CLUSRCVR.

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 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, this should be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it should be set to blanks. The actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set.

See the information about configuration parameters for an LU 6.2 connection for your platform in the *MQSeries Intercommunication* manual for more information.

On Windows NT SNA Server, and in the side object on OS/390, 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, OS/2 Warp, OS/390, 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 OS/390, 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 OS/390 for defining client-connection channels that will connect to servers on the platforms supporting NetBIOS)

SPX Sequenced packet exchange (supported only on OS/2 Warp, Windows, Windows NT, and DOS; it also applies to OS/390 for defining client-connection channels that will connect to servers on the platforms supporting SPX)

TCP Transmission Control Protocol - part of the TCP/IP protocol suite

UDP User Datagram Protocol - part of the TCP/IP protocol suite (supported only on AIX)

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, CLNTCONN, or CLUSSDR. It is not supported on OS/400 and is only supported on OS/390 for CLNTCONN channels.

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

DEFINE CHANNEL

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

Digital OpenVMS	OS/390	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.

Notes:

1. You can issue the DEFINE MAXSMSGS command at any time to change the number of messages allowed.
2. This command is valid only on OS/390. 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

DEFINE NAMELIST

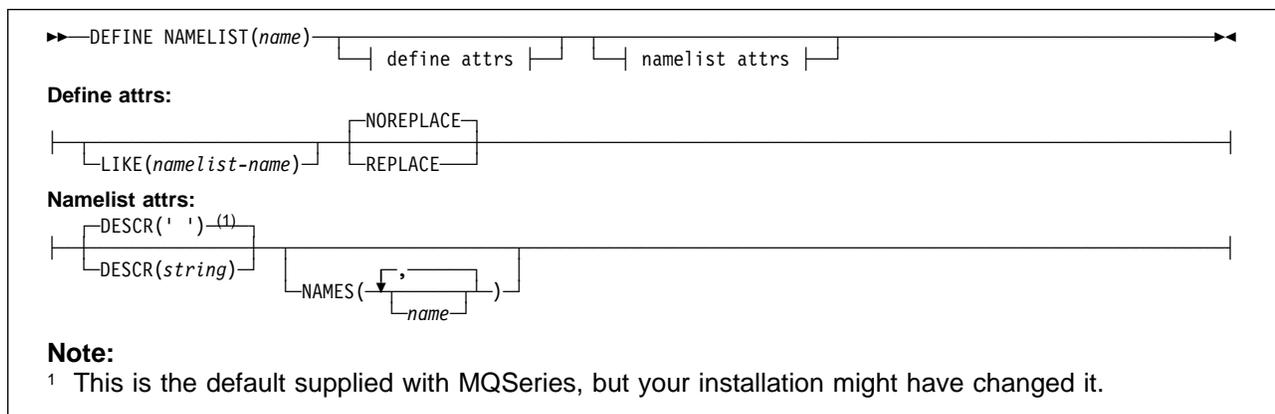
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use DEFINE NAMELIST to define a list of names. This is most commonly a list of cluster names or queue names.

Notes:

1. On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.

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.

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 161).

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**(*name, ...*)

| List of names.

| The names can be of any type, but must conform to the rules for naming MQSeries objects, with
| a maximum length of 48 characters.

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

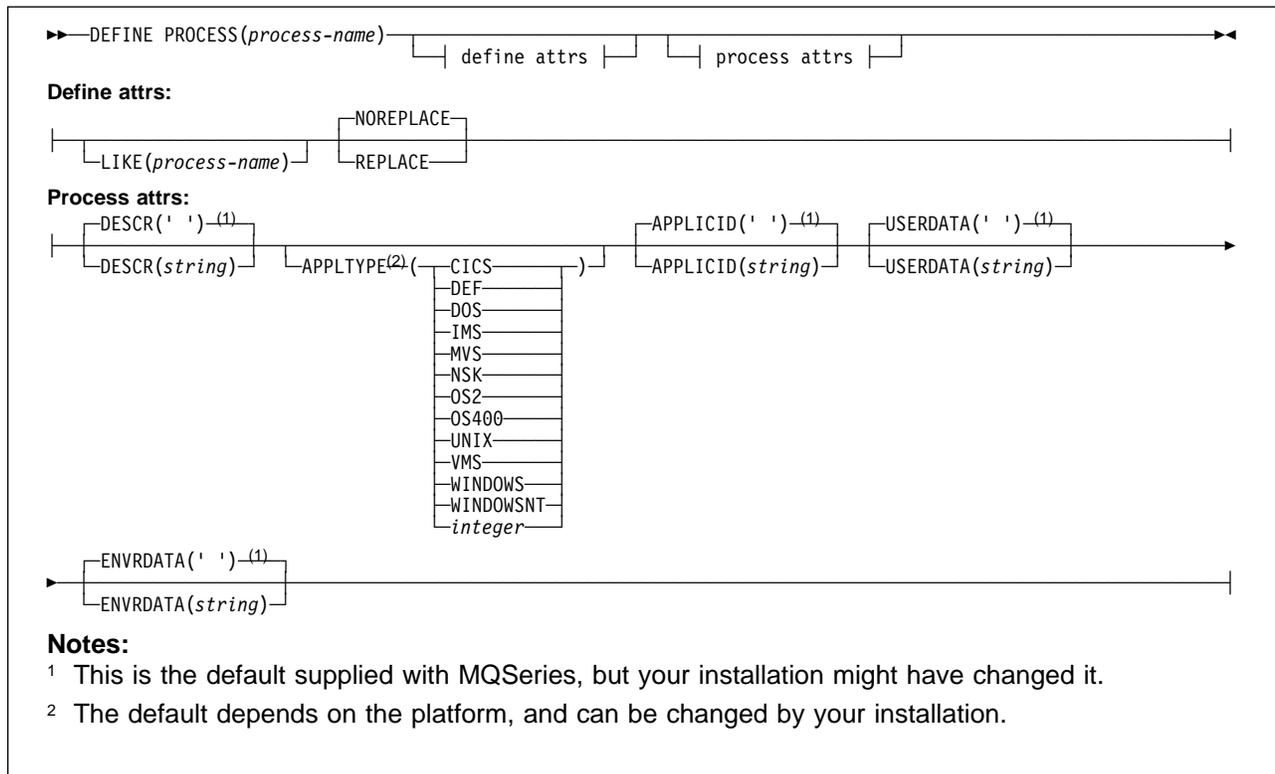
DEFINE PROCESS

DEFINE PROCESS

Digital OpenVMS	OS/390	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 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 OS/390, 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:

CICS	Represents a CICS transaction.
DOS	Represents a DOS application.
IMS	Represents an IMS transaction.
MVS	Represents an OS/390 application (batch or TSO).
NSK	Represents a Tandem NSK application.
OS2	Represents an OS/2 Warp application.
OS400	Represents an OS/400 application.
UNIX	Represents a UNIX application.
VMS	Represents a Digital OpenVMS application.
WINDOWS	Represents a Windows application.
WINDOWSNT	Represents a Windows NT application.
integer	User-defined application type in the range 65 536 through 999 999 999.
DEF	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 OS/390, CICS (default), DOS, IMS, MVS, OS2, UNIX, WINDOWS, WINDOWSNT, 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

DEFINE PROCESS

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 "Triggering channels" in the *MQSeries Intercommunication* manual for information about what these need as APPLICID.

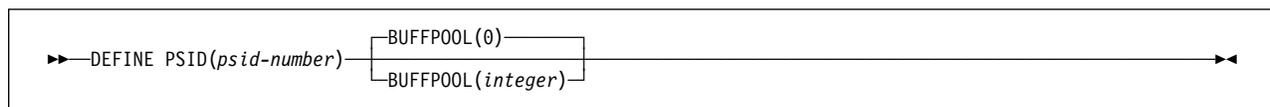
DEFINE PSID

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DEFINE PSID to define a page set and associated buffer pool.

Note: 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 OS/390 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 74.

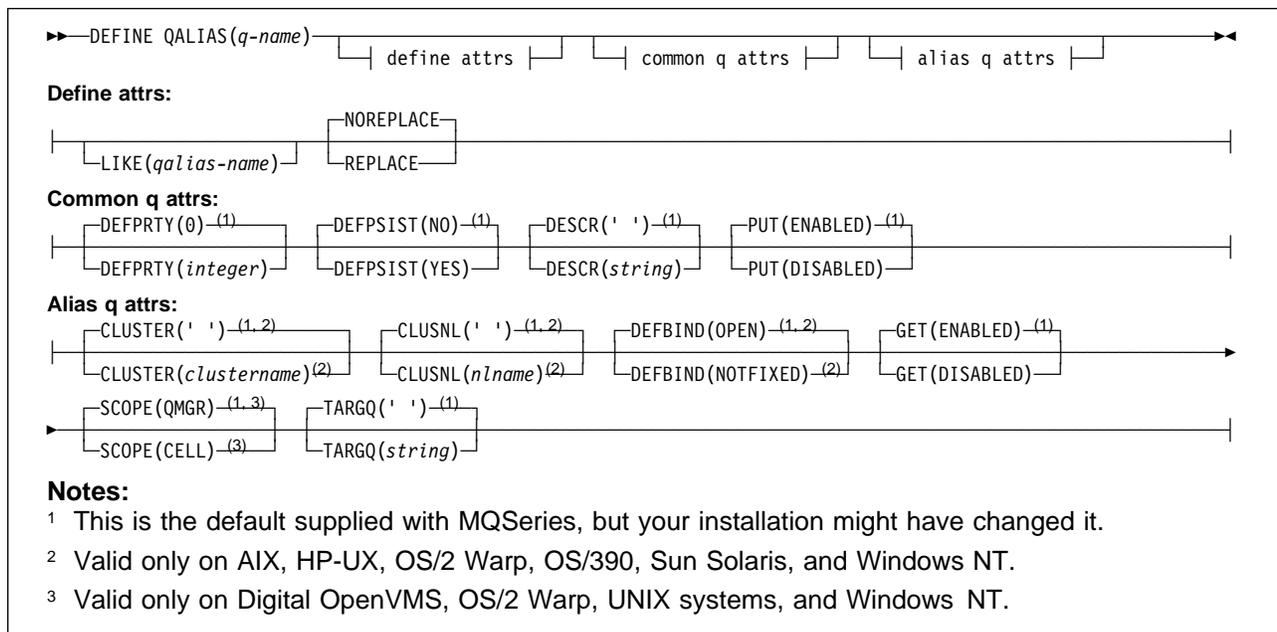
DEFINE QALIAS

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DEFINE QALIAS to define a new alias queue, and set its attributes.

Note: 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.

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 option, 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 option if you were using the ALTER command
- The object is open

The ALTER command with the FORCE option 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

Specifies the message persistence to be used when applications specify the MQPER_PERSISTENCE_AS_Q_DEF option.

NO Messages on this queue are lost across a restart of the queue manager. 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.

DEFINE QALIAS

DISABLED Messages cannot be added to the queue.

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

Alias queue attributes

CLUSTER(*clustername*)

The name of the cluster to which the queue belongs. The maximum length is 48 characters conforming to the rules for naming MQSeries objects. Changes to this parameter do not affect instances of the queue that are already open. Only one of the resultant values of **CLUSTER** or **CLUSNL** can be nonblank; you cannot specify a value for both.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*nlname*)

The name of the namelist that specifies a list of clusters to which the queue belongs. Changes to this parameter do not affect instances of the queue that are already open. Only one of the resultant values of **CLUSTER** or **CLUSNL** can be nonblank; you cannot specify a value for both.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

DEFBIND

Specifies the binding to be used when the application specifies **MQOO_BIND_AS_Q_DEF** on the **MQOPEN** call, and the queue is a cluster queue.

OPEN The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

NOTFIXED The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using **MQPUT**, and to change that selection subsequently should the need arise.

The **MQPUT1** call always behaves as if **NOTFIXED** had been specified.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

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 model queue)
- A cluster queue
- A local definition of a remote queue

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

Usage notes

1. DEFINE QALIAS(*otherqname*) TARGQ(*aliasqueue*) CLUSTER(*c*) has the effect of advertising queue *aliasqueue* by the name *otherqname*.
2. DEFINE QALIAS(*otherqname*) TARGQ(*aliasqueue*) has the effect of allowing a queue advertised by the name *otherqname* to be used on this queue manager by the name *aliasqueue*.

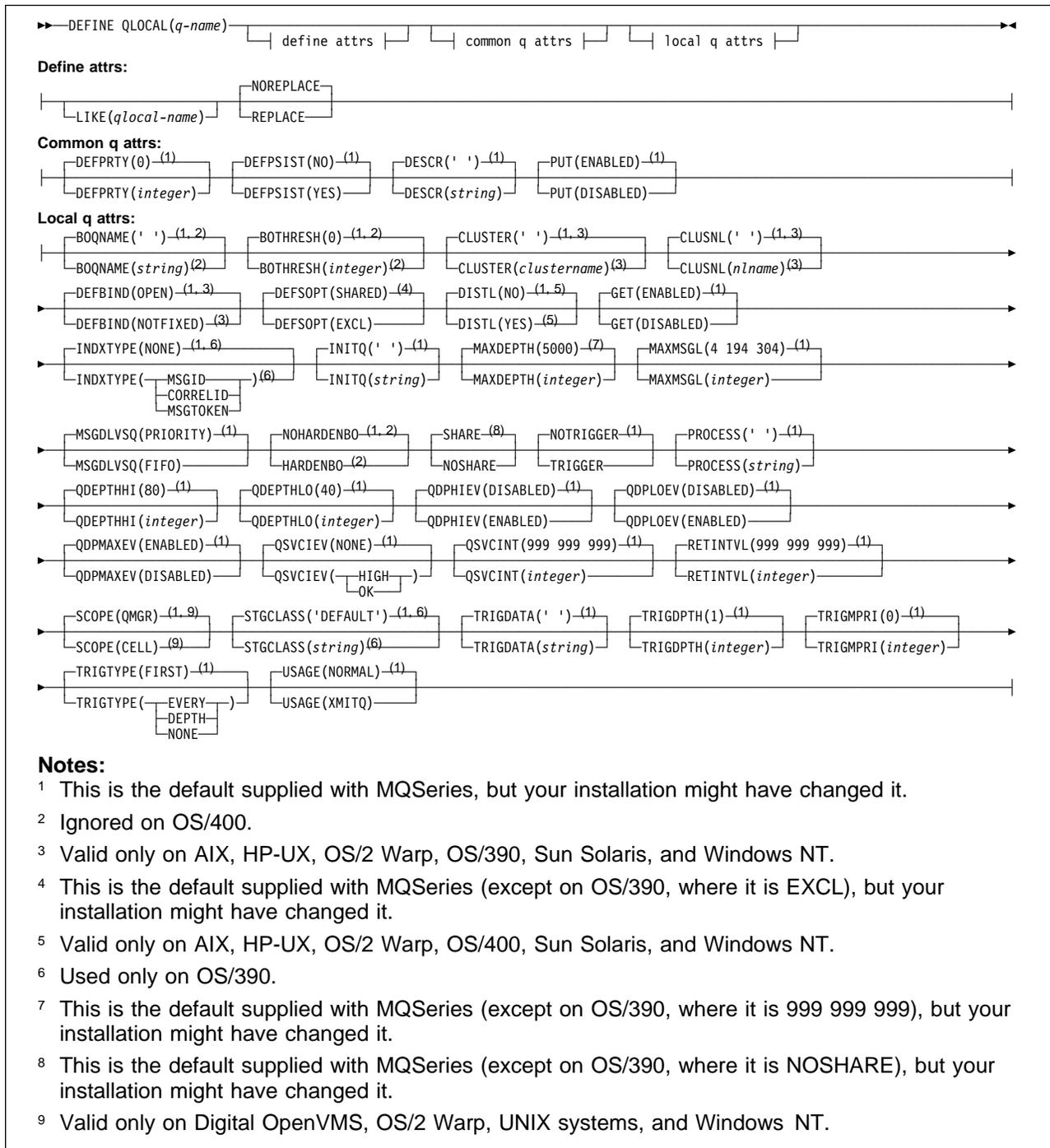
DEFINE QLOCAL

DEFINE QLOCAL

Digital OpenVMS	OS/390	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



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 QLOCAL

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 option, 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 option if you were using the ALTER command
- The object is open

The ALTER command with the FORCE option 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.)

DEFPSIST

| Specifies the message persistence to be used when applications specify the
| MQPER_PERSISTENCE_AS_Q_DEF option.

NO Messages on this queue are lost across a restart of the queue manager. 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.

CLUSTER(*clustername*)

The name of the cluster to which the queue belongs. The maximum length is 48 characters conforming to the rules for naming MQSeries objects. Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter cannot be set for transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx, or SYSTEM.COMMAND.xx queues.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*nlname*)

The name of the namelist that specifies a list of clusters to which the queue belongs. Changes to this parameter do not affect instances of the queue that are already open.

Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter cannot be set for transmission, SYSTEM.CHANNEL.xx, SYSTEM.CLUSTER.xx, or SYSTEM.COMMAND.xx queues.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

DEFINE QLOCAL

DEFBIND

Specifies the binding to be used when the application specifies MQOO_BIND_AS_Q_DEF on the **MQOPEN** call, and the queue is a cluster queue.

OPEN The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

NOTFIXED The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using **MQPUT**, and to change that selection subsequently should the need arise.

The **MQPUT1** call always behaves as if NOTFIXED had been specified.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

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.

MSGTOKEN An index of message tokens is maintained. Use this when the queue is a WLM managed queue that you are using with the Workload Manager functions of OS/390.

Note: If the queue is a transmission queue you cannot set INDXTYPE to MSGTOKEN.

The INDXTYPE attribute can be changed to NONE, MSGID, or CORRELID 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 can be changed to MSGTOKEN only when there are no messages on the queue. If you attempt to change this attribute to MSGTOKEN while there are messages on the queue, the command fails.

This attribute is supported only on OS/390. 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 OS/390
- 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.

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 QLOCAL

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 “Queue Depth High” 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 “Queue Depth Low” 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 “Queue Full” 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 “Queue Service Interval High” 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 168.

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 option 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 OS/390. See the *MQSeries for OS/390 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 OS/390, 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 164 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.

If you specify this option, do not specify values for CLUSTER and CLUSNL and do not specify INDXTYPE(MSGTOKEN).

DEFINE QMODEL

Digital OpenVMS	OS/390	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) ─┤
├── DEFPRTY(integer) ─┤
├── DEFPSIST(NO)-(1) ─┤
├── DEFPSIST(YES) ─┤
├── DESCR(' ')-(1) ─┤
├── DESCR(string) ─┤
├── PUT(ENABLED)-(1) ─┤
└── PUT(DISABLED) ─┤
└──────────────────▶

Local q attrs:
├── BOQNAME(' ')-(1,2) ─┤
├── BOQNAME(string)(2) ─┤
├── BOTHRESH(0)-(1,2) ─┤
├── BOTHRESH(integer)(2) ─┤
├── DEFSOPT(EXCL)-(1) ─┤
├── DEFSOPT(SHARED) ─┤
├── DISTL(NO)-(1,3) ─┤
├── DISTL(YES)-(3) ─┤
├── GET(ENABLED)-(1) ─┤
├── GET(DISABLED) ─┤
├── INDXTYPE(NONE)-(1,4) ─┤
├── INDXTYPE(MSGID)-(4) ─┤
├── INDXTYPE(CORRELID)-(4) ─┤
├── INDXTYPE(MSGTOKEN)-(4) ─┤
├── INITQ(' ')-(1) ─┤
├── INITQ(string) ─┤
├── MAXDEPTH(5000)-(5) ─┤
├── MAXDEPTH(integer) ─┤
├── MAXMSGL(4 194 304)-(1) ─┤
├── MAXMSGL(integer) ─┤
├── MSGDLVSVQ(PRIORITY)-(1) ─┤
├── MSGDLVSVQ(FIFO) ─┤
├── NOHARDENBO(1,2) ─┤
├── HARDENBO(2) ─┤
├── NOSHARE-(1) ─┤
├── SHARE ─┤
├── NOTRIGGER(1) ─┤
├── TRIGGER ─┤
├── PROCESS(' ')-(1) ─┤
├── PROCESS(string) ─┤
├── QDEPTHHI(80)-(1) ─┤
├── QDEPTHHI(integer) ─┤
├── QDEPTHLO(40)-(1) ─┤
├── QDEPTHLO(integer) ─┤
├── QDPHIEV(DISABLED)-(1) ─┤
├── QDPHIEV(ENABLED) ─┤
├── QDPLOEV(DISABLED)-(1) ─┤
├── QDPLOEV(ENABLED) ─┤
├── QDPMAXEV(ENABLED)-(1) ─┤
├── QDPMAXEV(DISABLED) ─┤
├── QSVCIEV(NONE)-(1) ─┤
├── QSVCIEV(HIGH) ─┤
├── QSVCIEV(OK) ─┤
├── QSVCINT(999 999 999)-(1) ─┤
├── QSVCINT(integer) ─┤
├── RETINTVL(999 999 999)-(1) ─┤
├── RETINTVL(integer) ─┤
├── STGCLASS(' DEFAULT')-(1,4) ─┤
├── STGCLASS(string)(4) ─┤
├── TRIGDATA(' ')-(1) ─┤
├── TRIGDATA(string) ─┤
├── TRIGDPHT(1)-(1) ─┤
├── TRIGDPHT(integer) ─┤
├── TRIGMPRI(0)-(1) ─┤
├── TRIGMPRI(integer) ─┤
├── TRIGTYPE(FIRST)-(1) ─┤
├── TRIGTYPE(EVERY) ─┤
├── TRIGTYPE(DEPTH) ─┤
├── TRIGTYPE(NONE) ─┤
├── USAGE(NORMAL)-(1) ─┤
└── 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 OS/390.
5 This is the default supplied with MQSeries (except on OS/390, 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

Specifies the message persistence to be used when applications specify the **MQPER_PERSISTENCE_AS_Q_DEF** option.

NO Messages on this queue are lost across a restart of the queue manager. 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 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.
- MSGTOKEN** An index of message tokens is maintained. Use this when the queue is a WLM managed queue that you are using with the Workload Manager functions of OS/390.

Note: If the queue is a transmission queue or a temporary-dynamic queue, you cannot set **INDXTYPE** to **MSGTOKEN**.

The **INDXTYPE** attribute can be changed to **NONE**, **MSGID**, or **CORRELID** 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 can only be changed to **MSGTOKEN** when there are no messages on the queue. If you attempt to change this attribute to **MSGTOKEN** while there are messages on the queue, the command fails.

This attribute is supported only on OS/390. 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 OS/390
- 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.

DEFINE QMODEL

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.

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 “Queue Depth High” 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

DEFINE QMODEL

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 “Queue Depth Low” 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 “Queue Full” 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 “Queue Service Interval High” 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 168.

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 OS/390. See the *MQSeries for OS/390 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 OS/390, 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 164 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.

If you specify this option, do not specify INDXTYPE(MSGTOKEN).

DEFINE QMODEL

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.

If you specify this option, do not specify INDXTYPE(MSGTOKEN).

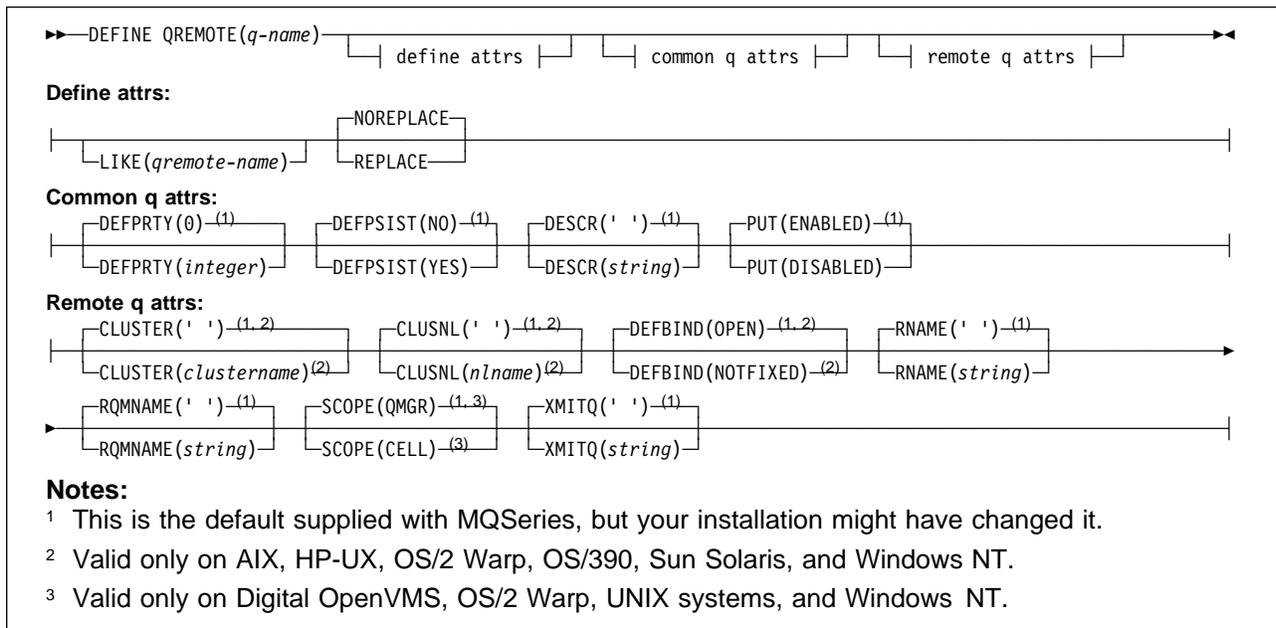
PERMDYN A permanent dynamic queue is created when an application issues an **MQOPEN** API call with the name of this model queue specified in the object descriptor (MQOD).

DEFINE QREMOTE

Digital OpenVMS	OS/390	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

DEFINE QREMOTE

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 option, 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 option if you were using the ALTER command
- The object is open

The ALTER command with the FORCE option 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

| Specifies the message persistence to be used when applications specify the
| MQPER_PERSISTENCE_AS_Q_DEF option.

NO Messages on this queue are lost across a restart of the queue manager. 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**CLUSTER**(*clustname*)

The name of the cluster to which the queue belongs. The maximum length is 48 characters conforming to the rules for naming MQSeries objects. Changes to this parameter will not affect instances of the queue that are already open.

Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*nlname*)

The name of the namelist that specifies a list of clusters to which the queue belongs. Changes to this parameter will not affect instances of the queue that are already open.

Only one of the resultant values of CLUSTER or CLUSNL can be nonblank; you cannot specify a value for both.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

DEFBIND

Specifies the binding to be used when the application specifies MQOO_BIND_AS_Q_DEF on the **MQOPEN** call, and the queue is a cluster queue.

OPEN The queue handle is bound to a specific instance of the cluster queue when the queue is opened.

NOTFIXED The queue handle is not bound to any particular instance of the cluster queue. This allows the queue manager to select a specific queue instance when the message is put using **MQPUT**, and to change that selection subsequently should the need arise.

The **MQPUT1** call always behaves as if NOTFIXED had been specified.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

DEFINE QREMOTE

RNAME(*string*)

Name of remote queue. This is the local name of the queue as defined on the queue manager specified by RQMNAME.

- 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.

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).

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 MQSeries object 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.

Usage notes

1. DEFINE QREMOTE(*rqueue*) RNAME(*otherq*) RQMNAME(*otherqm*) CLUSTER(*cl*) has the effect of advertising this queue manager as a store and forward gateway to which messages for queue *rqueue* can be sent. It has no effect as a reply-to queue alias, except on the local queue manager.
 DEFINE QREMOTE(*otherqm*) RNAME() RQMNAME(*anotherqm*) XMITQ(*xq*) CLUSTER has the effect of advertising this queue manager as a store and forward gateway to which messages for *anotherqm* can be sent.
2. RQMNAME can itself be the name of a cluster queue manager within the cluster, thus (as with QALIAS definitions) you can map the advertised queue manager name to another name locally.
3. It is possible for the values of RQMNAME and QREMOTE to be the same if RQMNAME is itself a cluster queue manager. If this definition is also advertised using a CLUSTER attribute, care should be taken not to choose the local queue manager in the cluster workload exit because a cyclic definition will result.

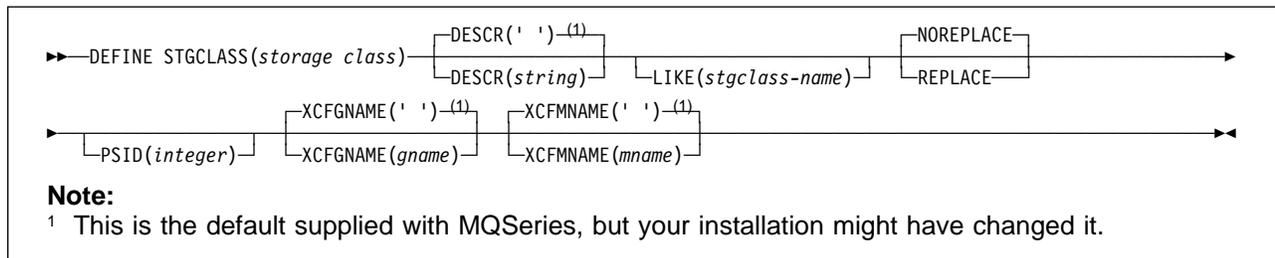
DEFINE STGCLASS

DEFINE STGCLASS

Digital OpenVMS	OS/390	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 103.

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

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

DELETE CHANNEL

DELETE CHANNEL

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
✓	✓	✓	✓	✓	✓	✓

Use DELETE CHANNEL to delete a channel definition.

Notes for OS/390 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 13.
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 can only delete cluster-sender channels that have been created manually.

Synonym: DELETE CHL

```
▶▶ DELETE CHANNEL(channel-name) [CHLTABLE(QMGR) (1) | CHLTABLE(CLNT) (1)] ▶▶
```

Note:

¹ 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 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 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 OS/390, 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	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use DELETE NAMELIST to delete a namelist definition.

Notes:

1. On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.

Synonym: DELETE NL

▶▶—DELETE NAMELIST (*name*)—————▶▶

Keyword and parameter descriptions

You must specify which namelist definition you want to delete.

(name) The name of the namelist definition to be deleted. The name must be defined to the local queue manager.

If an application has this namelist open, the command fails.

DELETE PROCESS

DELETE PROCESS

Digital OpenVMS	OS/390	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.

(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	OS/390	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(<i>q-name</i>)——▶▶

Keyword and parameter descriptions

You must specify which alias queue you want to delete.

(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.

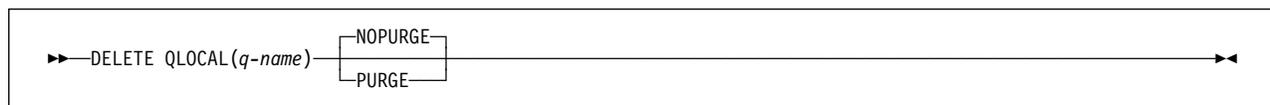
DELETE QLOCAL

DELETE QLOCAL

Digital OpenVMS	OS/390	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.

(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	OS/390	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.

(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	OS/390	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.

(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	OS/390	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.

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

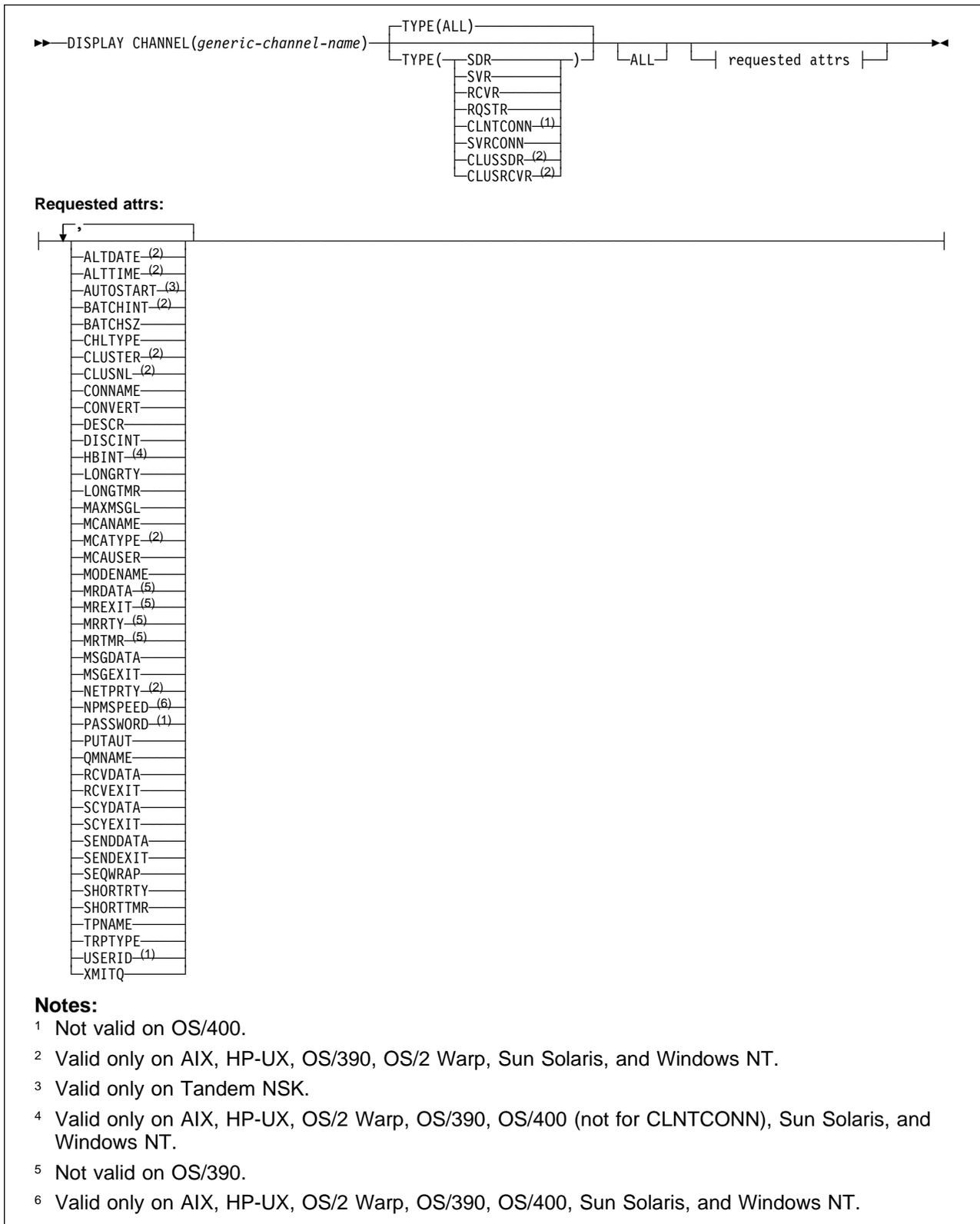
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use DISPLAY CHANNEL to display a channel definition.

Notes:

1. On OS/390, 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 13.
2. You can only display cluster-sender channels if they were created manually.

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.

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 OS/390, 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.

CLUSSDR Cluster-sender channels only are displayed (valid on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only).

CLUSRCVR Cluster-receiver channels only are displayed (valid on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only).

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, OS/2 Warp, OS/390, 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 OS/390, 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.

ALTDATE The date on which the definition was last altered, in the form *yyyy-mm-dd*

ALTTIME The time at which the definition was last altered, in the form *hh.mm.dd*

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 OS/390, the channel 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.
CLUSTER	The name of the cluster to which the channel belongs
CLUSNL	The name of the namelist that specifies the list of clusters to which the channel belongs
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
MCANAME	Message channel agent name
MCATYPE	Whether message channel agent runs as a separate process or a separate thread
MCAUSER	Message channel agent user identifier
MODENAME	LU 6.2 mode name
MRDATA	Channel message-retry exit user data
MREXIT	Channel message-retry exit name
MRRTY	Channel message-retry exit retry count
MRTMR	Channel message-retry exit retry time
MSGDATA	Channel message exit user data
MSGEXIT	Channel message exit names
NETPRTY	The priority for the network connection
NPMSPEED	Nonpersistent message speed
PASSWORD	Password for initiating LU 6.2 session (if nonblank, this is displayed as asterisks)
PUTAUT	Put authority
QMNAME	Queue manager name
RCVDATA	Channel receive exit user data
RCVEXIT	Channel receive exit names
SCYDATA	Channel security exit user data
SCYEXIT	Channel security exit names
SENDDATA	Channel send exit user data
SENDEXIT	Channel send exit names

DISPLAY CHANNEL

SEQWRAP	Sequence number wrap value
SHORTRTY	Short retry count
SHORTTMR	Short retry timer
TPNAME	LU 6.2 transaction program name
TRPTYPE	Transport type
USERID	User identifier for initiating LU 6.2 session
XMITQ	Transmission queue name

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.

| Status for all channels that meet the selection criteria is given, whether the channels were defined
| manually or automatically.

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.

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, cluster-sender, cluster-receiver, or server-connection channel current at the same time (the requester is acting as a receiver). This occurs if several senders, at different queue managers, each initiate a session with this receiver, using the same channel name. For channels of other types, there can only be one instance current at any time.

For all channel types, however, there can be more than one set of saved status information available for a 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.

DISPLAY CHSTATUS

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 might not be the same as in the channel definition, and might differ between the current channel status and the saved channel status. (Using CONNNAME for limiting the number of sets of status is therefore not recommended.) For example, if CONNNAME is blank in the channel definition or (when using TCP) 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), 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 OS/390, 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.

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.

CURMSGs For a sending channel, this is the number of messages that have been sent in the current batch. It is incremented as each message is sent, and when the channel becomes in doubt it is the number of messages that are in doubt.

For a saved channel instance, this 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.

DISPLAY CHSTATUS

INITIALIZING

The channel initiator is attempting to start a channel. This is valid only on AIX, Digital OpenVMS, HP-UX, OS/390, OS/2 Warp, Sun Solaris, and Windows NT. On OS/390, 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 OS/390.

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 OS/390, 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.
 - On Tandem NSK, this is the CPU ID and PID, displayed in decimal.
- This information is not available on OS/390. 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.
- MAXMSGL** The maximum message length being used for this session (valid only on OS/390).
- 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 OS/390. 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 CLUSQMGR

DISPLAY CLUSQMGR

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use DISPLAY CLUSQMGR to display a cluster information about queue managers in a cluster.

Notes:

1. On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.
2. On OS/390, the command fails if the channel initiator has not been started.

Synonym: DIS CLUSQMGR

```

▶ DISPLAY CLUSQMGR(generic-qmname) [ALL] [CHANNEL (generic name)] [CLUSTER (generic name)]
▶ [requested attrs] [channel attrs]

```

Requested attrs:

- CLUSDATE
- CLUSTIME
- DEFTYPE
- QMID
- QMTYPE
- STATUS
- SUSPEND

Channel attrs:

- ALTDATA
- ALTTIME
- BATCHINT
- BATCHSZ
- CONNNAME
- CONVERT
- DESCR
- DISCINT
- HBINT
- LONGRTY
- LONGTMR
- MAXMSGL
- MCANAME
- MCATYPE
- MCAUSER
- MODENAME
- MRDATA (1)
- MREXIT (1)
- MRRTY (1)
- MRTMR (1)
- MSGDATA
- MSGEXIT
- NETPRTY
- NPMSPEED
- PASSWORD (1)
- PUTAUT
- RCVDATA
- RCVEXIT
- SCYDATA
- SCYEXIT
- SENDDATA
- SENDEXIT
- SEQWRAP
- SHORTRTY
- SHORTTMR
- TPNAME
- TRPTYPE
- USERID (1)

Note:
¹ Not valid on OS/390.

Keyword and parameter descriptions

(*generic qmname*)

The name of the cluster queue manager to be displayed.

A trailing asterisk(*) matches all cluster queue managers with the specified stem followed by zero or more characters. An asterisk (*) on its own specifies all cluster queue managers.

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. This is the default if you do not specify a generic name and do not request any specific attributes.

CHANNEL(*generic-name*)

This is optional, and limits the information displayed to cluster queue managers with the specified channel name. The value can be a generic name.

CLUSTER(*generic-name*)

This is optional, and limits the information displayed to cluster queue managers with the specified cluster name. The value can be a generic name.

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.

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, and do not cause an error.

CLUSDATE The date on which the definition became available to the local queue manager, in the form yyyy-mm-dd.

CLUSTIME The time at which the definition became available to the local queue manager, in the form hh.mm.ss.

DEFTYPE How the cluster queue manager was defined:

CLUSSDR As a cluster-sender channel from an explicit definition.

CLUSSDRA As a cluster-sender channel by auto-definition alone.

CLUSSDRB As a cluster-sender channel by auto-definition and an explicit definition.

CLUSRCVR As a cluster-receiver channel from an explicit definition.

QMTYPE The function of the queue manager in the cluster:

REPOS Provides a full repository service.

NORMAL Does not provide a full repository service.

QMID The internally generated unique name of the queue manager.

STATUS The current status of the channel for this queue manager. 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

The channel is performing channel negotiation and is not yet ready to transfer messages.

INACTIVE

The channel is not active.

INITIALIZING

The channel initiator is attempting to start a channel. On OS/390, this is displayed as INITIALIZI.

DISPLAY CLUSQMGR

RUNNING

The channel is either transferring messages at this moment, or is waiting for messages to arrive on the transmission queue so that they can be transferred.

STOPPING

The channel is stopping, or 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.

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 is made to establish a connection automatically.

A channel in this state can be restarted only by issuing the **START CHANNEL** command, or starting the MCA program in an operating-system dependent manner.

REQUESTING

A local requester channel is requesting services from a remote MCA.

SUSPEND Whether this queue manager is suspended from the cluster or not (as a result of the **SUSPEND QMGR** command). This is either **YES** or **NO**.

Channel attributes

ALTDATE The date on which the definition or information was last altered, in the form **yyyy-mm-dd**

ALTTIME The time at which the definition or information was last altered, in the form **hh.mm.ss**

BATCHINT Minimum batch duration

BATCHSZ Batch size

CONNAME Connection name

CONVERT Whether the 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

MCANAME Message channel agent name

MCATYPE Whether the message channel agent runs as a separate process or a separate thread

MCAUSER Message channel agent user identifier

MODENAME	LU 6.2 mode name
MRDATA	Channel message-retry exit user data
MREXIT	Channel message-retry exit name
MRRTY	Channel message-retry exit retry count
MRTMR	Channel message-retry exit retry time
MSGDATA	Channel message exit user data
MSGEXIT	Channel message exit names
NETPRTY	The priority for the network connection
NPMSPEED	Nonpersistent message speed
PASSWORD	Password for initiating LU 6.2 session (if nonblank, this is displayed as asterisks)
PUTAUT	Put authority
RCVDATA	Channel receive exit user data
RCVEXIT	Channel receive exit names
SCYDATA	Channel security exit user data
SCYEXIT	Channel security exit name
SENDDATA	Channel send exit user data
SENDEXIT	Channel send exit names
SEQWRAP	Sequence number wrap value
SHORTRTY	Short retry count
SHORTTMR	Short retry timer
TRPTYPE	Transport type
TPNAME	LU 6.2 transaction program name
USERID	User identifier for initiating LU 6.2 session

DISPLAY CMDSERV

Digital OpenVMS	OS/390	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

```
▶—DISPLAY CMDSERV—▶
```

Usage notes

1. The command server takes messages from the system command input queue and processes them. DISPLAY CMDSERV displays the status of the command server.
2. The response to this command is a message showing the current status of the command server, which is one of the following:

- ENABLED** Available to process 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

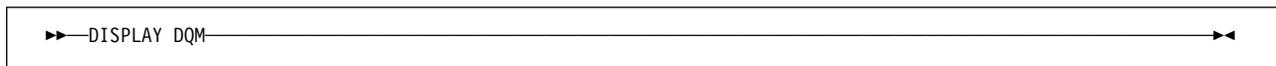
DISPLAY DQM

Digital OpenVMS	OS/390	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 13.

Synonym: DIS 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 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 system name
 - How many channel connections are current, and whether they are active, stopped, or retrying
 - The maximum number of current connections

DISPLAY MAXSMSGS

DISPLAY MAXSMSGS

Digital OpenVMS	OS/390	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.

Notes:

1. This command is valid only on OS/390. For other platforms, use the MAXUMSGS keyword of the DISPLAY QMGR command instead.
2. You can issue the DISPLAY MAXSMSGS command at any time to see the number of messages allowed.

Synonym: DIS MAXSM

▶—DISPLAY MAXSMSGS—◀

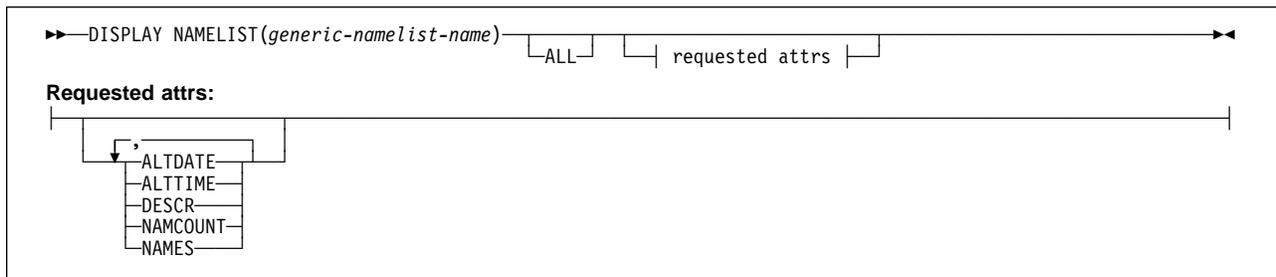
DISPLAY NAMELIST

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use DISPLAY NAMELIST to display the names in a namelist.

Note: On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.

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:

- ALTDATE** The date on which the definition was last altered, in the form yyyy-mm-dd
- ALTTIME** The time at which the definition was last altered, in the form hh.mm.ss
- DESCR** Description
- NAMCOUNT** Number of names in the list
- NAMES** List of names

See DEFINE NAMELIST on page 98 for more information about the DESCR and NAMES attributes.

DISPLAY PROCESS

DISPLAY PROCESS

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
✓	✓	✓	✓	✓	✓	✓

Use DISPLAY PROCESS to display the attributes of one or more MQSeries processes.

Synonym: DIS PRO

▶—DISPLAY PROCESS(*generic-process-name*)—▶

ALL requested attrs

Requested attrs:

- ALTDATA (1)
- ALLTIME (1)
- APPLICID
- APPLTYPE
- DESCR
- ENVRDATA
- USERDATA

Note:
1 Valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

Keyword and parameter descriptions

You must specify the name of the process you want to display. This can be a specific process name or a generic process name. By using a generic process name, you can display either:

- All process definitions
- One or more processes that match the specified name

(generic-process-name)

The name of the process definition to be displayed (see “Rules for naming MQSeries objects” on page 5). A trailing asterisk (*) matches all processes with the specified stem followed by zero or more characters. An asterisk (*) on its own specifies all processes. The names must all be defined to the local queue manager.

ALL Specify this to display all the attributes. 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, OS/390, Sun Solaris, and Windows NT, this is the default if you do not specify a generic name, and do not request any specific attributes.

On other platforms, if no attributes are specified (and the ALL keyword is not specified), the default is that the process names are returned.

Requested attributes: You can request the following information for the process name:

- ALTDATA** The date on which the definition was last altered, in the form *yyyy-mm-dd*
- ALLTIME** The time at which the definition was last altered, in the form *hh.mm.ss*
- APPLICID** Application identifier
- APPLTYPE** Application type

DESCR	Description
ENVRDATA	Environment data
USERDATA	User data

See DEFINE PROCESS on page 100 for more information about individual attributes.

DISPLAY QMGR

Digital OpenVMS	OS/390	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:
ALTDATE (1)
ALTTIME (1)
AUTHOREV
CCSID
CHAD (2)
CHADEV (3)
CHADEXIT (3)
CLWLEXIT (1)
CLWLDATA (1)
CLWLEN (1)
CMDLEVEL
COMMANDQ
CPILEVEL (4)
DEADQ
DEFXMITQ
DESCR
DISTL (2)
INHIBTEV
LOCALEV
MAXHANDS
MAXMSGL
MAXPRTY
MAXUMSGS (5)
PERFMEV
PLATFORM
QMID (1)
QMNAME
REMOETEV
REPOS (1)
REPOSNL (1)
STRSTPEV
SYNCPT
TRIGINT
    
```

Notes:

- 1 Valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
- 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, OS/390, OS/400, Sun Solaris, and Windows NT.
- 4 Valid only on OS/390.
- 5 Not valid on OS/390.

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:

ALTDATE	The date on which the definition was last altered, in the form yyyy-mm-dd. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
ALTTIME	The time at which the definition was last altered, in the form hh.mm.ss. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
AUTHOREV	Whether authorization events are generated.
CCSID	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.
CHAD	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.
CHADEV	Whether auto-definition events are enabled. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT.
CHADEXIT	The name of the channel auto-definition exit. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT.
CLWLEXIT	The name of the cluster workload exit. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
CLWLDATA	The data passed to the cluster workload exit. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
CLWLLEN	The maximum number of bytes of message data that is passed to the cluster workload exit. This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
CMDLEVEL	Command level. This indicates the function level of the queue manager.
COMMANDQ	The name of the system-command input queue. Suitably authorized applications can put commands on this queue.
CPILEVEL	Reserved, this value has no significance.

DISPLAY QMGR

DEADQ The name of the queue to which messages are sent if they cannot be routed to their correct destination (the dead-letter queue or undelivered-message queue). The default is blanks.

For example, messages are put on this queue when:

- A message arrives at a queue manager, destined for a queue that is not yet defined on that queue manager
- A message arrives at a queue manager, but the queue for which it is destined cannot receive it because, possibly:
 - The queue is full
 - The queue is inhibited for puts
 - The sending node does not have authority to put the message on the queue
- An exception message 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.

DEFXMITQ 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.

DESCR Description.

DISTL 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.

INHIBTEV Whether inhibit events are generated.

LOCALEV Whether local error events are generated.

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

MAXMSGL 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 14 for more information.)

MAXPRTY The maximum priority. This is 9.

MAXUMSGS Maximum number of uncommitted messages within one syncpoint.
This keyword is not supported on OS/390; use DISPLAY MAXSMSGS instead.

PERFMEV Whether performance-related events are generated.

PLATFORM The architecture of the platform on which the queue manager is running. This is MVS, OPENVMS, NSK, OS2, OS400, UNIX, or WINDOWSNT.

QMID The internally generated unique name of the queue manager.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

QMNAME The name of the local queue manager. See “Rules for naming MQSeries objects” on page 5.

REMOTEEV	Whether remote error events are generated.
REPOS	The name of a cluster for which this queue manager is to provide a repository manager service.
	This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
REPOSNL	The name of a list of clusters for which this queue manager is to provide a repository manager service.
	This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.
STRSTPEV	Whether start and stop events are generated.
SYNCPT	Whether syncpoint support is available with the queue manager. On OS/2 Warp, OS/390, UNIX systems, and Windows NT it is always available.
TRIGINT	The trigger interval.

DISPLAY QUEUE

DISPLAY QUEUE

Digital OpenVMS	OS/390	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.

Notes:

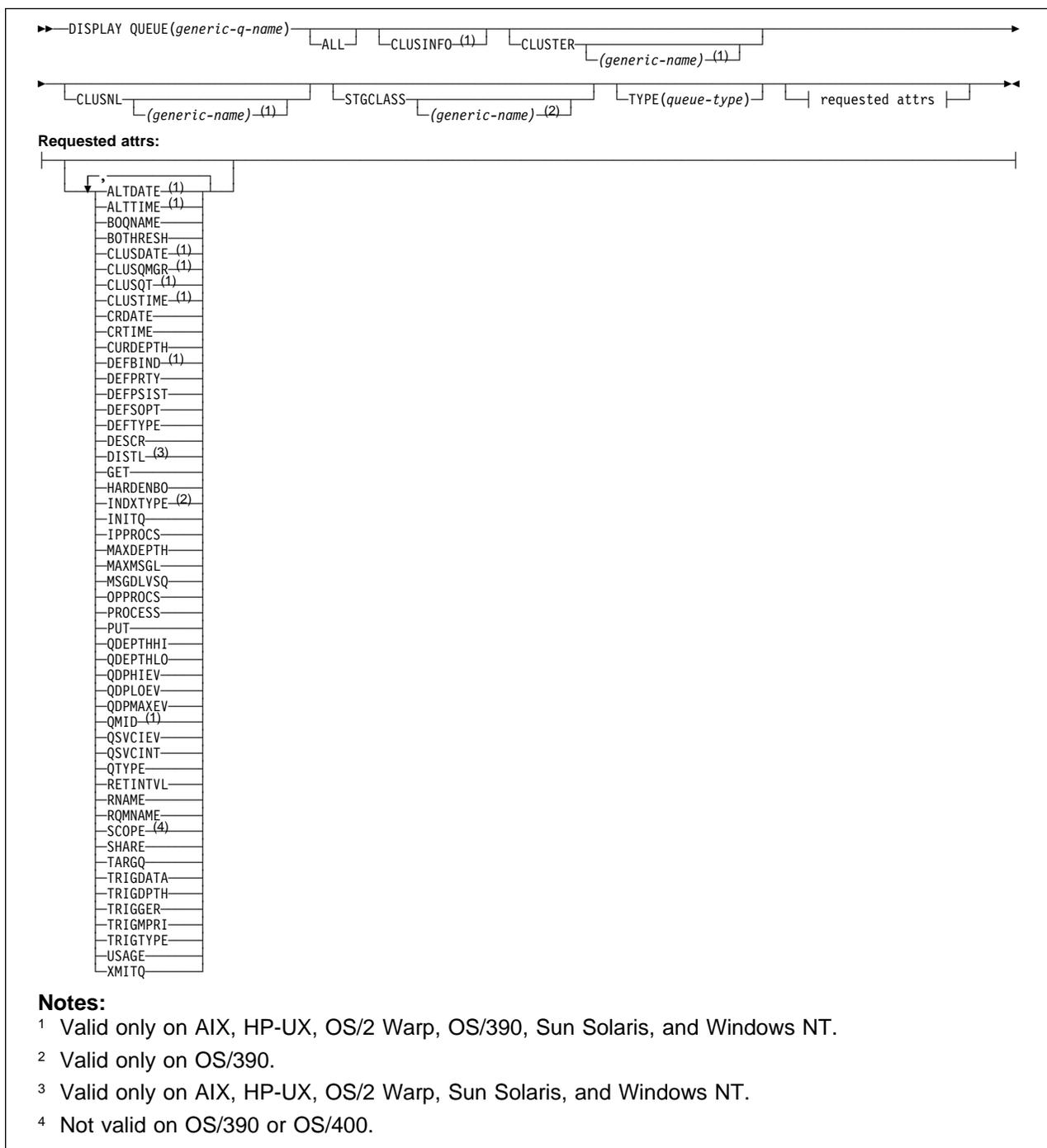
1. On AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT, you can use the following commands (or their synonyms) as an alternative way to display these attributes.

- DISPLAY QALIAS
- DISPLAY QCLUSTER
- DISPLAY QLOCAL
- DISPLAY QMODEL
- DISPLAY QREMOTE

These commands produce the same output as the DISPLAY QUEUE TYPE(*queue-type*) command. If you enter the commands this way, do not use the TYPE keyword because this causes an error.

2. On OS/390, the channel initiator must be running before you can display information about cluster queues (using TYPE(QCLUSTER) or the CLUSINFO keyword).

Synonym: DIS Q



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

DISPLAY QUEUE

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, OS/2 Warp, OS/390, Sun Solaris, and Windows NT, this is the default if you do not specify a generic name, and do not request any specific attributes.

CLUSINFO

This requests that, in addition to information about attributes of queues defined on this queue manager, information about these and other queues in the cluster that match the selection criteria is displayed. In this case, there might be multiple queues with the same name displayed. The cluster information is obtained from the repository on this queue manager.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSTER(*generic-name*)

This is optional, and limits the information displayed to queues with the specified cluster name if entered with a value in brackets. The value can be a generic name. Only queue types for which CLUSTER is a valid attribute are restricted in this way by this keyword; other queue types that meet the other selection criteria are displayed. If you do not enter a value to qualify this parameter, it is treated as a requested attribute, and cluster name information is returned about all the queues displayed.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

CLUSNL(*generic-name*)

This is optional, and limits the information displayed if entered with a value in brackets:

- For queues defined on the local queue manager, only those with the specified cluster list. The value can be a generic name. Only queue types for which CLUSNL is a valid attribute are restricted in this way; other queue types that meet the other selection criteria are displayed.
- For cluster queues, only those belonging to clusters in the specified cluster list if the value is not a generic name. If the value is a generic name, no restriction is applied to cluster queues.

If you do not enter a value to qualify this parameter, it is treated as a requested attribute, and cluster list information is returned about all the queues displayed.

This parameter is valid only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

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 OS/390.

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; this includes cluster queues if CLUSINFO is also specified. You can specify any of the queue types allowed for a DEFINE command (QLOCAL, QALIAS, QREMOTE, or their synonyms). A queue type of QCLUSTER can be specified to display only cluster queue information on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT.

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>					
	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
ALTDAT ¹	√	√	√	√	√
ALTTIME ¹	√	√	√	√	√
BOQNAME	√	√			
BOTHRESH	√	√			
CLUSDATE ¹					√
CLUSNL ¹	√		√	√	
CLUSQMGR ¹					√
CLUSQT ¹					√
CLUSTER ¹	√		√	√	√
CLUSTIME ¹					√
CRDATE	√	√			
CRTIME	√	√			
CURDEPTH	√				
DEFBIND ¹	√		√	√	
DEFPRTY	√	√	√	√	√
DEFPSIST	√	√	√	√	√
DEFSOPT	√	√			
DEFTYPE	√	√			
DESCR	√	√	√	√	√
DISTL ²	√	√			
GET	√	√	√		
HARDENBO	√	√			
INDXTYPE ³	√	√			
INITQ	√	√			
IPPROCS	√				
MAXDEPTH	√	√			
MAXMSGL	√	√			
MSGDLVSQ	√	√			
OPPROCS	√				

DISPLAY QUEUE

Table 3 (Page 2 of 2). Attributes that can be returned by the DISPLAY QUEUE command

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
PROCESS	√	√			
PUT	√	√	√	√	√
QDEPTHHI	√	√			
QDEPTHLO	√	√			
QDPHIEV	√	√			
QDPLOEV	√	√			
QDPMAXEV	√	√			
QMID ¹					√
QSVCI EV	√	√			
QSVCI NT	√	√			
QTYPE	√	√	√	√	√
RETINTVL	√	√			
RNAME				√	
RQMNAME				√	
SCOPE ⁴	√		√	√	
SHARE	√	√			
STGCLASS ³	√	√			
TARGQ			√		
TRIGDATA	√	√			
TRIGDPTH	√	√			
TRIGGER	√	√			
TRIGMPRI	√	√			
TRIGTYPE	√	√			
USAGE	√	√			
XMITQ				√	

Notes:

1. Supported only on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT
2. Supported only on AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT
3. Supported only on OS/390
4. Not supported on OS/390 or OS/400

ALTDATE The date on which the definition or information was last altered, in the form yyyy-mm-dd.

ALLTIME The time at which the definition or information was last altered, in the form hh.mm.ss.

BOQNAME Backout requeue name.

BOTHRESH Backout threshold.

CLUSDATE The date on which the definition became available to the local queue manager, in the form yyyy-mm-dd.

CLUSNL The namelist that defines the cluster that the queue is in.

CLUSQMGR The name of the queue manager that hosts the queue.

	CLUSQT	Cluster queue type. This can be:
	QALIAS	The cluster queue represents an alias queue.
	QLOCAL	The cluster queue represents a local queue.
	QMGR	The cluster queue represents a queue manager alias.
	QREMOTE	The cluster queue represents a remote queue.
	CLUSTER	The name of the cluster that the queue is in.
	CLUSTIME	The time at which the definition became available to the local queue manager, in the form hh.mm.ss.
	CRDATE	The date on which the queue was defined (in the form yyyy-mm-dd).
	CRTIME	The time at which the queue was defined (in the form hh.mm.ss).
	CURDEPTH	Current depth of queue.
	DEFBIND	Default message binding.
	DEFPRTY	Default priority of the messages put on the queue.
	DEFPSIST	Whether the default persistence of messages put on this queue is set to NO or YES. NO means that messages are lost across a restart of the queue manager.
	DEFSOPT	Default share option on a queue opened for input.
	DEFTYPE	Queue definition type. This can be: <ul style="list-style-type: none"> • PREDEFINED (Predefined) The queue was created with a DEFINE command, either by an operator or by a suitably authorized application sending a command message to the service queue. • PERMDYN (Permanent dynamic) Either the queue was created by an application issuing MQOPEN with the name of a model queue specified in the object descriptor (MQOD), or (if this is a model queue) this determines the type of dynamic queue that can be created from it. • TEMPDYN (Temporary dynamic) Either the queue was created by an application issuing MQOPEN with the name of a model queue specified in the object descriptor (MQOD), or (if this is a model queue) this determines the type of dynamic queue that can be created from it.
	DESCR	Descriptive comment.
	DISTL	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.)
	GET	Whether the queue is enabled for gets.
	HARDENBO	Whether to harden the get back out count.
	INDXTYPE	Index type (supported only on OS/390).
	INITQ	Initiation queue name.
	IPPROCS	Number of handles indicating that the queue is open for input.
	MAXDEPTH	Maximum depth of queue.
	MAXMSGL	Maximum message length.
	MSGDLVSQ	Message delivery sequence.
	OPPROCS	Number of handles indicating that the queue is open for output.

DISPLAY QUEUE

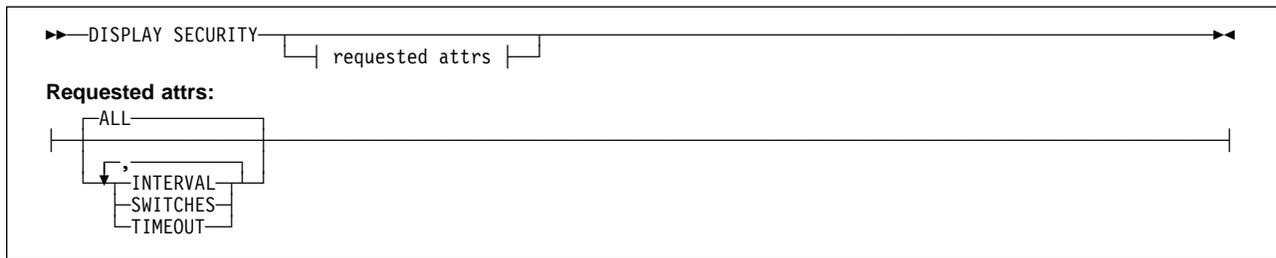
PROCESS	Process name.
PUT	Whether the queue is enabled for puts.
QDEPTHHI	Queue Depth High event generation threshold.
QDEPTHLO	Queue Depth Low event generation threshold.
QDPHIEV	Whether Queue Depth High events are generated.
QDPLOEV	Whether Queue Depth Low events are generated.
QDPMAXEV	Whether Queue Full events are generated.
QMID	The internally generated unique name of the queue manager that hosts the queue.
QSVCI EV	Whether service interval events are generated.
QSVCI NT	Service interval event generation threshold.
QTYPE	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 OS/390, the queue type is always displayed. On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, <code>TYPE(<i>type</i>)</code> can be used as a synonym for this parameter.
RETINTVL	Retention interval.
RNAME	Name of the local queue, as known by the remote queue manager.
RQMNAME	Remote queue manager name.
SCOPE	Scope of queue definition (not supported on OS/390 or OS/400).
SHARE	Whether the queue can be shared.
STGCLASS	Storage class.
TARGQ	Local name of aliased queue.
TRIGDATA	Trigger data.
TRIGDPTH	Trigger depth.
TRIGGER	Whether triggers are active.
TRIGMPRI	Threshold message priority for triggers.
TRIGTYPE	Trigger type.
USAGE	Whether or not the queue is a transmission queue.
XMITQ	Transmission queue name.

DISPLAY SECURITY

Digital OpenVMS	OS/390	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.

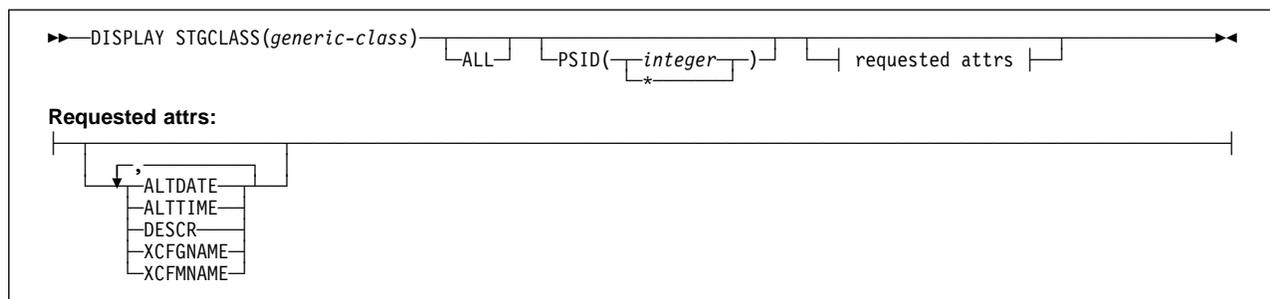
See “ALTER SECURITY” on page 67 for details of the TIMEOUT and INTERVAL attributes.

DISPLAY STGCLASS

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use DISPLAY STGCLASS to display information about storage classes.

Synonym: DIS STC



Keyword and parameter descriptions

You use DISPLAY STGCLASS to show the page set identifiers that are associated with each storage class.

(generic-class)

Name of the storage class. This is required.

This is 1 through 8 characters. The first character is in the range A through Z; subsequent characters are A through Z or 0 through 9.

A trailing asterisk (*) matches all storage classes with the specified stem followed by zero or more characters. An asterisk (*) on its own specifies all storage classes.

PSID(*integer*)

The page set identifier that a storage class maps to. This is optional.

The string consists of two numeric characters, in the range 00 through 99. An asterisk (*) on its own specifies all page set identifiers. See DEFINE PSID on page 103.

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.

This is the default if you do not specify a generic name, and 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.

The default, if no attributes are specified (and the ALL keyword is not specified) is the storage class names and their page set identifiers are displayed.

ALTDATA

The date on which the definition was last altered, in the form yyyy-mm-dd.

ALTTIME

The time at which the definition was last altered, in the form hh.mm.ss.

DESCR Descriptive comment.

XCFGNAME

The name of the XCF group that MQSeries is a member of.

XCFMNAME

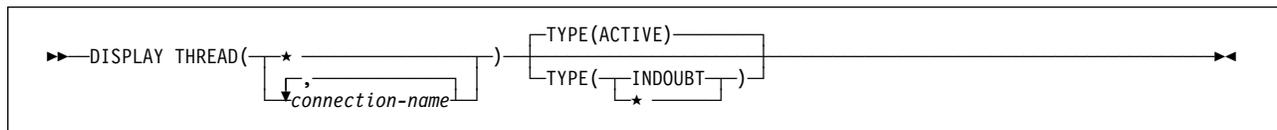
The XCF member name of the IMS system within the XCF group specified in XCFGNAME.

DISPLAY THREAD

Digital OpenVMS	OS/390	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
- For RRS connections, this is RRSBATCH

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. You might only have to start the recovery coordinator (CICS, IMS, or RRS), or you might need to do more. They might have been in doubt at the last restart, or they might have become in doubt since the last restart.

★ 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.

- | For more information about the DISPLAY THREAD command and in-doubt recovery, see the *MQSeries for OS/390 System Management Guide* and messages CSQV401I through CSQV406I in the *MQSeries for OS/390 Messages and Codes* manual.
- | **Note:** This command is issued internally by MQSeries when taking a checkpoint, and when the queue manager is starting and stopping, so that a list of threads that are in doubt at the time is written to the OS/390 console log.

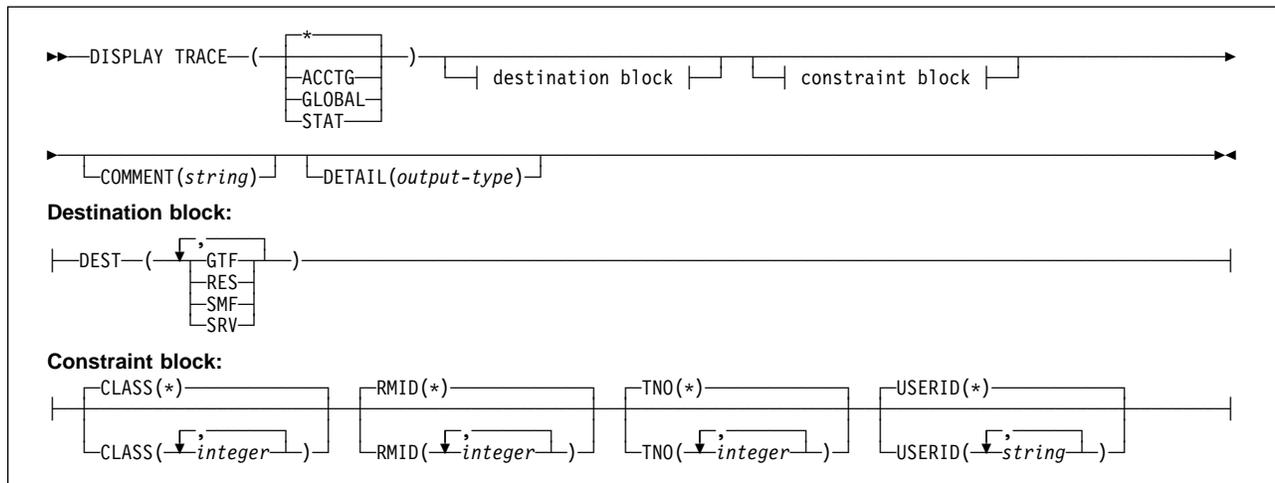
DISPLAY TRACE

DISPLAY TRACE

Digital OpenVMS	OS/390	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 CSQW127I (in the *MQSeries for OS/390 Messages and Codes* manual) for more information about trace qualification.
- 1,2 Display both summary and qualification information

* Display both summary and qualification information

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 201 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 201 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 201 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.

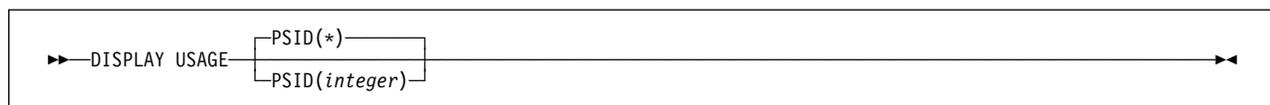
DISPLAY USAGE

DISPLAY USAGE

Digital OpenVMS	OS/390	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 103.

DISPLAY USAGE returns three sets of information in the following messages:

CSQI018I The number of pages currently being used on the page set specified.

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 OS/390 Messages and Codes* manual for more information about these messages.

| **Note:** This command is issued internally by MQSeries during shutdown so that the restart RBA is
| recorded on the OS/390 console log.

PING CHANNEL

Digital OpenVMS	OS/390	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:

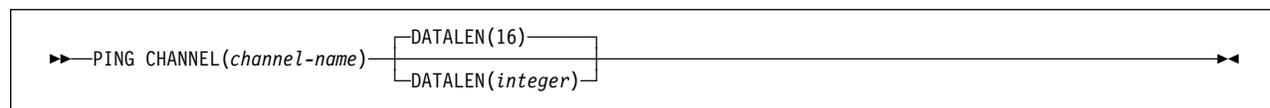
1. On OS/390, this is valid only for channels used for distributed queuing without CICS. If you are using CICS for distributed queuing, see Note 2 on page 13.

2. On OS/390, the command fails if the channel initiator has not been started.

3. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager's repository.

This command can be used only for sender (SDR), server (SVR), and cluster-sender (CLUSSDR) channels (including those that have been defined automatically). It is not valid if the channel is running; however, it is valid if the channel is stopped or in retry mode.

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

PING QMGR

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√		√	√	√	√	√

Use PING QMGR to test whether the queue manager is responsive to commands.

Note: 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

▶—PING QMGR—◀

RECOVER BSDS

Digital OpenVMS	OS/390	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.

Note: Command processing consists of allocating a data set with the same name as the one that encountered the error and copying onto the new data set the contents of the BSDS that does not have an error.

Synonym: REC BSDS

▶—RECOVER BSDS—▶

REFRESH CLUSTER

REFRESH CLUSTER

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use REFRESH CLUSTER to discard all locally held cluster information (including any autodefined channels that are in doubt), and force it to be rebuilt. This allows the cluster to be “cold-started.”

Notes:

1. On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.
2. On OS/390, the command fails if the channel initiator has not been started.

Synonym: REF CLUSTER

►—REFRESH CLUSTER(*clustername*)—◄

Keyword and parameter descriptions

(*clustername*)

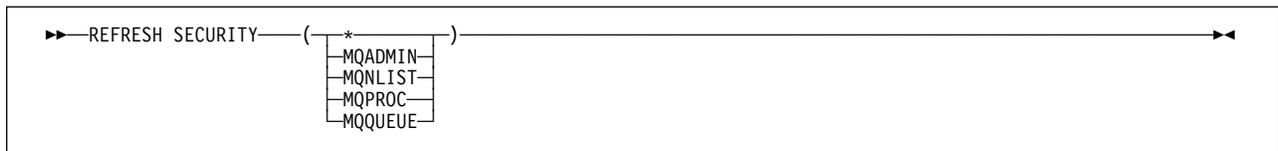
The name of the cluster to be refreshed. This is required.

REFRESH SECURITY

Digital OpenVMS	OS/390	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 class for which the security refresh is to be performed. The classes are:
- | * All resource classes
- | **MQADMIN** Administration type resources
- | **MQNLIST** Namelist resources
- | **MQPROC** Process resources
- | **MQQUEUE** Queue resources

Note: If, when refreshing this class, it is determined that a security switch relating to one of the other classes has been changed, a refresh for that class also takes place.

REBUILD SECURITY is another synonym for REFRESH SECURITY.

RESET CHANNEL

RESET CHANNEL

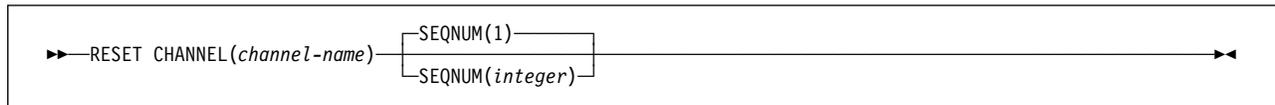
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

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.

Notes:

1. On OS/390, 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 13.
2. This command can be issued to a channel of any type except SVRCONN and CLNTCONN channels, (including those that have been defined automatically). However, if it is issued to a sender, server or cluster-sender channel, then in addition to resetting the value at the end at which the command is issued, the value at the other (receiver, requester, or cluster-receiver) end is also reset to the same value the next time this channel is initiated (and resynchronized if necessary).
3. If the command is issued to a receiver, requester, or cluster-requester channel, the value at the other end is *not* reset as well; this must be done separately if necessary.
4. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager's repository.

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 CLUSTER

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use RESET CLUSTER to perform special operations on clusters.

Notes:

1. On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.
2. On OS/390, the command fails if the channel initiator has not been started.

Synonym: None

▶—RESET CLUSTER(*clustname*)—ACTION(FORCEREMOVE)—QMNAME(*qmname*)—◀

Keyword and parameter descriptions

(clustname)

The name of the cluster to be reset. This is required.

ACTION(FORCEREMOVE)

Requests that the queue manager is forcibly removed from the cluster. This might be needed to ensure proper clean up after a queue manager has been deleted.

This action can be requested only by a repository queue manager.

QMNAME(*qmname*)

The name of the queue manager to be forcibly removed.

RESET TPIPE

RESET TPIPE

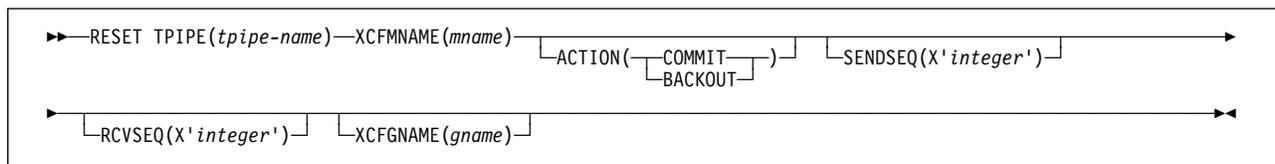
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use RESET TPIPE to reset the recoverable sequence numbers for an IMS Tpipe used by the MQSeries-IMS bridge.

Notes:

1. This command is used in response to the resynchronization error reported in message CSQ2020E, and initiates resynchronization of the Tpipe with IMS.
2. The command fails if the queue manager is not connected to the specified XCF member.
3. The command fails if the queue manager is connected to the specified XCF member, but the Tpipe is open.
4. 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(*X'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.

RCVSEQ(X'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

RESOLVE CHANNEL

Digital OpenVMS	OS/390	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 OS/390, 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 13.
2. On OS/390, the command fails if the channel initiator has not been started.
3. This command can be used only for sender (SDR), server (SVR), and cluster-sender (CLUSDR) channels (including those that have been defined automatically).
4. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager's repository.

Synonym: RESOLVE CHL (RES CHL on OS/390)

```
►—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

Usage notes

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.

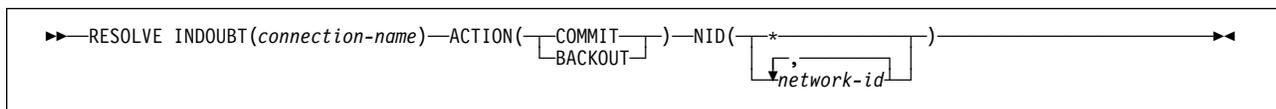
RESOLVE INDOUBT

Digital OpenVMS	OS/390	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.

Note: This command does not apply to units of recovery associated with batch or TSO applications, unless you are using the RRS adapter.

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.
- For an RRS connection, it is RRSBATCH.

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 as returned by the DISPLAY THREAD command, and is of the form *net-node.net-urid*, where:

- *net-node* identifies the originator of the thread, except for RRSBATCH where it is omitted.
- *net-urid* is the hexadecimal number assigned to the unit of recovery by the originating system for the specific thread to be resolved.

When *net-node* is present there must be a period (.) between it and *net-urid*.

(*) Resolves all threads associated with the connection.

Examples:

```
RESOLVE INDOUBT(CICSA) ACTION(COMMIT) NID(CICSA.ABCDEF0123456789)
```

```
RESOLVE INDOUBT(CICSA) ACTION(BACKOUT) NID(*)
```

RESUME QMGR

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use RESUME QMGR to inform other queue managers in a cluster that the local queue manager is available again for processing and can be sent messages. It reverses the action of the SUSPEND QMGR command.

Notes:

1. On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.
2. On OS/390, the command fails if the channel initiator has not been started.

Synonym: None

```

▶—RESUME QMGR—CLUSTER(clustname)
                |
                |—CLUSNL(nlname)
  
```

Keyword and parameter descriptions

CLUSTER(*clustname*)

The name of the cluster to resume availability for.

CLUSNL(*nlname*)

The name of the namelist specifying a list of clusters to resume availability for.

RVERIFY SECURITY

Digital OpenVMS	OS/390	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	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√	√	√	√

Use START CHANNEL to start a channel.

Notes:

1. On OS/390, 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 13.
2. On OS/390, the command fails if the channel initiator has not been started.
3. This command can be issued to a channel of any type except CLNTCONN channels (including those that have been defined automatically). If, however, it is issued to a receiver (RCVR), server-connection (SVRCONN) or cluster-receiver (CLUSRCVR) channel, the only action is to enable the channel, not to start it.
4. Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel. If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the channel that was last added to the local queue manager's repository.

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

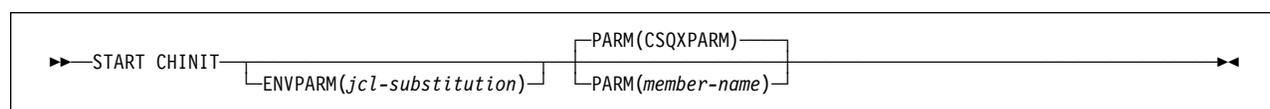
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
√	√	√	√		√	√

Use START CHINIT to start a channel initiator.

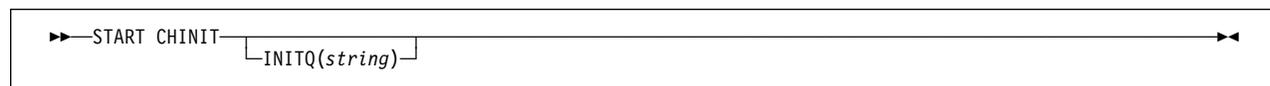
Note: On OS/390, this is valid only for channels used for distributed queuing without CICS.

Synonym: STA CHI

MQSeries for OS/390



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 OS/390.

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 OS/390 (the initiation queue on OS/390 is always SYSTEM.CHANNEL.INITQ). On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, you can specify which initiation queue to use; if you do not specify this, SYSTEM.CHANNEL.INITQ is used. On other platforms it must be specified.

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 OS/390.

START CMDSERV

Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

Use START CMDSERV to initialize the command server.

Synonym: STA CS



Usage notes

1. START CMDSERV starts the command server and allows it to process commands in the system-command input queue (SYSTEM.COMMAND.INPUT).
2. If this command is issued through the initialization files or through the operator console before work is released to the queue manager (that is, before the command server is started automatically), it overrides any earlier STOP CMDSERV command and allows the queue manager to start the command server automatically by putting it into an ENABLED state.
3. If this command is issued through the operator console while the command server is in a STOPPED or DISABLED state, it starts the command server and allows it to process commands on the system-command input queue immediately.
4. If the command server is in a RUNNING or WAITING state (including the case when the command is issued through the command server itself), or if the command server has been stopped automatically because the queue manager is closing down, no action is taken, the command server remains in its current state, and an error message is returned to the command originator.

START LISTENER

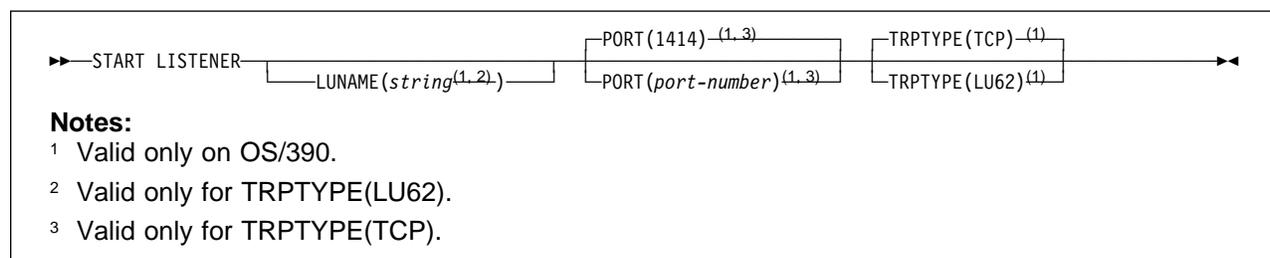
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√	√	√		√	√

Use START LISTENER to start a channel listener.

Notes:

1. On UNIX systems, the command is valid only for AIX, HP-UX, and Sun Solaris.
2. On OS/390, this is valid only for channels used for distributed queuing without CICS.
3. On OS/390, the command fails if the channel initiator has not been started.
4. On OS/400, OS/2 Warp, UNIX systems, and Windows NT this command is valid only for channels for which the transmission protocol (TRPTYPE) is TCP.

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 supported only on OS/390.

PORT(*port-number*)

Port number for TCP. This is valid only if the transmission protocol (TRPTYPE) is TCP.

This parameter is supported only on OS/390.

TRPTYPE

Transport type to be used. This is optional.

TCP TCP. This is the default if TRPTYPE is not specified.

LU62 SNA LU 6.2.

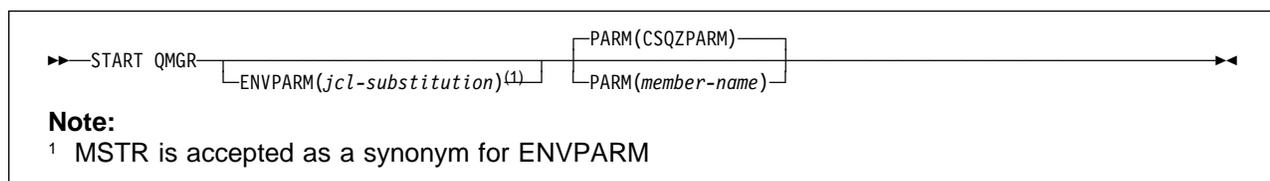
This parameter is supported only on OS/390.

START QMGR

Digital OpenVMS	OS/390	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.

ENVPARM(*jcl-substitution*)

The parameters and values to be substituted in the JCL procedure (xxxxMSTR, where xxxx is the queue manager name) that is used to start the queue manager address space.

jcl-substitution

One or more character strings of the form:

keyword=value

enclosed in single quotation marks. If you use more than one character string, separate the strings by commas and enclose the entire list in single quotation marks, for example ENVPARM('HLQ=CSQ,VER=120').

MSTR is accepted as a synonym for ENVPARM

PARM(*member-name*)

The load module that contains the queue manager initialization parameters. *member-name* is the name of a load module provided by the installation.

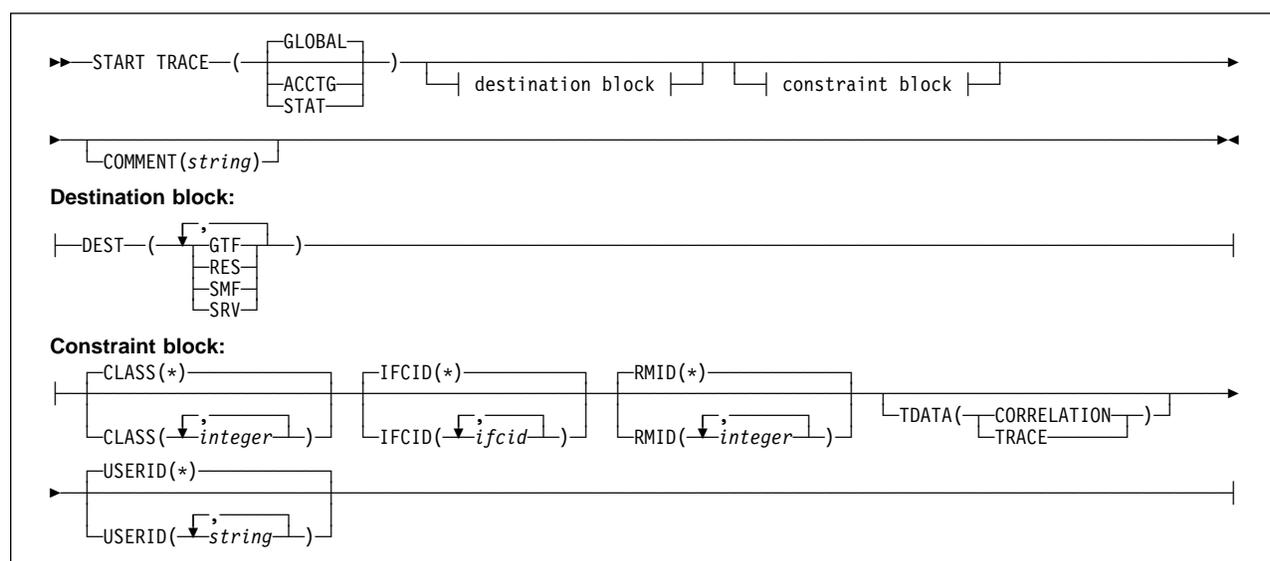
The default is CSQZPARM, which is provided by MQSeries.

START TRACE

Digital OpenVMS	OS/390	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 OS/390 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 OS/390 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 OS/390 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

START TRACE

<i>Table 7 (Page 2 of 2). Resource Manager identifiers that are allowed</i>	
RMID	Resource manager
23	General command processing
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

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

Digital OpenVMS	OS/390	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

1. When you issue the STOP CHINIT command, MQSeries stops any channels that are running in the following way:
 - Sender and server channels are stopped using STOP CHANNEL MODE(QUIESCE)
 - All other channels are stopped using STOP CHANNEL MODE(FORCE)

See “STOP CHANNEL” on page 205 for information about what this involves.
2. You might receive communications-error messages as a result of issuing the STOP CHINIT command.

STOP CMDSERV

STOP CMDSERV

Digital OpenVMS	OS/390	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

1. STOP CMDSERV stops the command server from processing commands in the system-command input queue (SYSTEM.COMMAND.INPUT).
2. If this command is issued through the initialization files or through the operator console before work is released to the queue manager (that is, before the command server is started automatically), it prevents the command server from starting automatically and puts it into a DISABLED state. It overrides an earlier START CMDSERV command.
3. If this command is issued through the operator console or the command server while the command server is in a RUNNING state, it stops the command server when it has finished processing its current command. When this happens, the command server enters the STOPPED state.
4. If this command is issued through the operator console while the command server is in a WAITING state, it stops the command server immediately. When this happens, the command server enters the STOPPED state.
5. If this command is issued while the command server is in a DISABLED or STOPPED state, no action is taken, the command server remains in its current state, and an error message is returned to the command originator.

STOP LISTENER

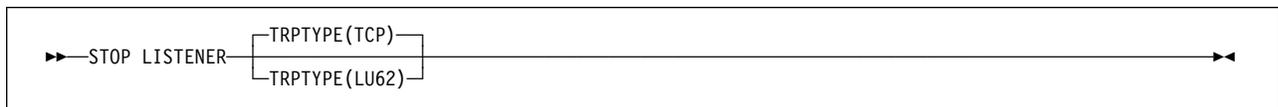
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√					

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. This is the default if TRPTYPE is not specified.

LU62 SNA LU 6.2.

On OS/390, 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

STOP QMGR

Digital OpenVMS	OS/390	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 OS/390 commands.

This option deregisters MQSeries from the MVS automatic restart manager (ARM).

FORCE Terminates programs currently being executed, including utilities. No new program is allowed to start. This option might cause in-doubt situations.

This option might not work if all the active logs are full, and log archiving has not occurred. In this situation it will be necessary to issue the OS/390 CANCEL command to terminate.

This option deregisters MQSeries from the MVS automatic restart manager (ARM).

RESTART Terminates programs currently being executed, including utilities. No new program is allowed to start. This option might cause in-doubt situations.

This option might not work if all the active logs are full, and log archiving has not occurred. In this situation it will be necessary to issue the MVS CANCEL command to terminate.

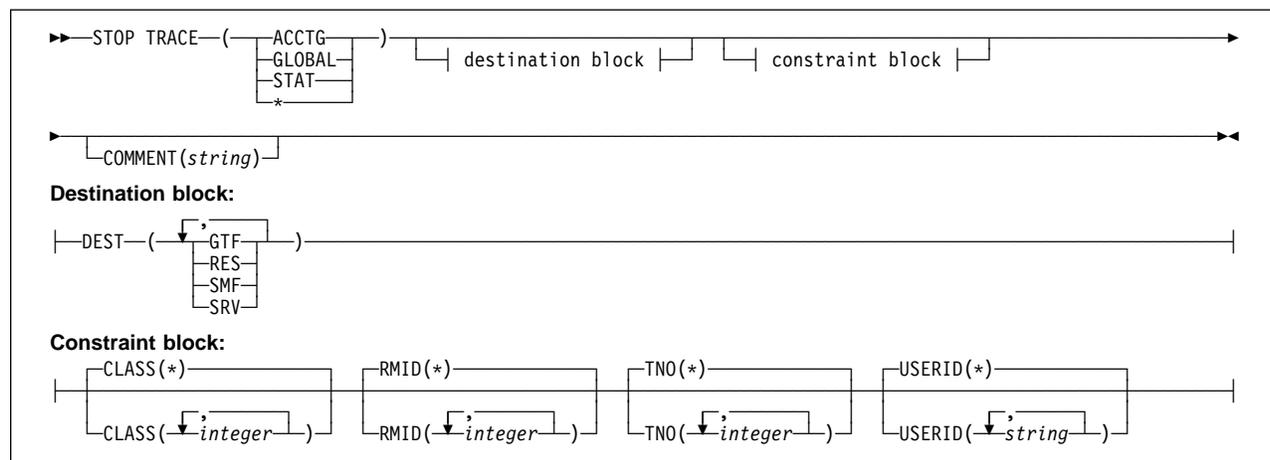
This option does not deregister MQSeries from ARM, so the queue manager is eligible for immediate automatic restart.

STOP TRACE

Digital OpenVMS	OS/390	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 OS/390 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 201.

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.

STOP TRACE

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 201 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.

SUSPEND QMGR

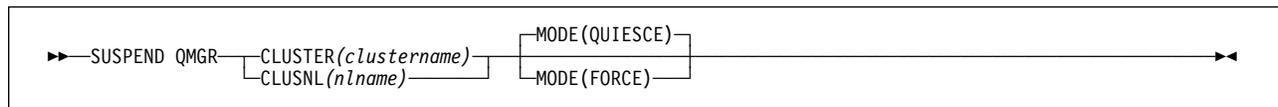
Digital OpenVMS	OS/390	OS/400	OS/2 Warp	Tandem NSK	UNIX systems	Windows NT
	√		√		√	√

Use SUSPEND QMGR to inform other queue managers in a cluster that the local queue manager is not available for processing and cannot be sent messages. Its action can be reversed by the RESUME QMGR command.

Notes:

1. On UNIX systems, the command is valid only on AIX, HP-UX, and Sun Solaris.
2. On OS/390, the command fails if the channel initiator has not been started.

Synonym: None



Keyword and parameter descriptions

CLUSTER(*clustername*)

The name of the cluster to suspend availability for.

CLUSNL(*nlname*)

The name of the namelist specifying a list of clusters to suspend availability for.

MODE Specifies how the suspension of availability is to take effect:

QUIESCE Other queue managers in the cluster are advised that the local queue manager should not be sent further messages.

FORCE All inbound and outbound channels to other queue managers in the cluster are stopped forcibly.

SUSPEND QMGR

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 OS/390.
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.

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
Notes:				
1. Applies on OS/390 only				
2. Does not apply on OS/390				

Command summary

<i>Table 9. 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 QCLUSTER (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			WRKMMSG	
Reset queue statistics	Reset Queue Statistics (See note 3)			
Notes:				
1. Does not apply on OS/390				
2. Applies on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only				
3. Does not apply on Tandem NSK				

<i>Table 10 (Page 1 of 2). Commands for process definition administration</i>				
Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Change process attributes	Change Process	ALTER PROCESS	CHGMQPRC	

Table 10 (Page 2 of 2). Commands for process definition administration

Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
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. (See note 1)

Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Change a namelist	Change Namelist	ALTER NAMELIST		
Copy a namelist	Copy namelist	DEFINE NAMELIST(x) LIKE(y)		
Define a namelist	Create Namelist	DEFINE NAMELIST		
Delete a namelist	Delete Namelist	DELETE NAMELIST		
Display a namelist	Inquire Namelist	DISPLAY NAMELIST		
Note:				
1. Applies on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only				

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	
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	runmqslr (See note 4)
Stop a channel	Stop Channel	STOP CHANNEL	ENDMQMCHL	

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
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	
Notes:				
<ol style="list-style-type: none"> Does not apply on OS/390 if you are using CICS for distributed queuing Applies on OS/390 only Applies on AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT only Applies on AIX, Digital OVMS, HP-UX, OS/2 Warp, Sun Solaris, Tandem NSK, and Windows NT only Applies on AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT only 				
In MQSeries for Tandem NonStop Kernel, use TS/MP or the control command runmqlsr to start TCP channel listeners.				

Table 13. Commands for cluster administration. (See Note 1)

Operation	PCF	MQSC	OS/400 CL	Commands for distributed platforms
Display cluster information	Inquire Cluster Queue Manager	DISPLAY CLUSQMGR		
Refresh cluster information	Refresh Cluster	REFRESH CLUSTER		
Reset cluster	Reset Cluster	RESET CLUSTER		
Resume cluster processing	Resume Queue Manager Cluster	RESUME QMGR		
Suspend cluster processing	Suspend Queue Manager Cluster	SUSPEND QMGR		
Note:				
1. Applies on AIX, HP-UX, OS/2 Warp, OS/390, Sun Solaris, and Windows NT only				

Table 14. 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		
Note:				
1. Applies on OS/390 only				

Table 15 (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		
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)

Command summary

Table 15 (Page 2 of 2). Commands for system-dependent function

Operation	PCF	MQSC (see note 1)	OS/400 CL	Commands for distributed platforms
Run dead-letter queue handler			STRMQMDLQ	runmqdlq (See note 6)
<p>Notes:</p> <ol style="list-style-type: none"> 1. Applies on OS/390 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, Sun Solaris, and Tandem NSK 5. Applies on AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT 6. Applies on AIX, HP-UX, OS/2 Warp, Sun Solaris, Tandem NSK, and Windows NT <p>In MQSeries for Tandem NonStop Kernel, as an alternative to the control commands dspmqcsv, strmqcsv, and endmqcsv, you can use PATHCOM commands.</p>				

Table 16. Other control commands in MQSeries for Tandem NonStop Kernel

Operation	Commands
Alter queue volume	altnmqfls
Perform housekeeping on a queue manager	cleanqmq
Convert V1.5.1 queues and channels to V2.2	cnv1520
Convert V1.5.1 messages to V2.2	cnvmsgs
Convert client channel definition table	cnvclchl
Install MQSeries for Tandem NonStop Kernel	instmqmq
<p>Note: As an alternative to the control command runmqtrm, you can use PATHCOM commands. There are no MQSC or PCF equivalents of commands in this group.</p>	

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*.

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*.

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 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 Chapter 9, "Administering remote MQSeries objects" 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 "Performing local administration tasks using MQSC commands" in 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, 1999. 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.

Issuing OS/2 commands

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 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.  
No commands have a syntax error.  
All valid MQSC commands were 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, 1999. 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.
No commands have a syntax error.
All valid MQSC commands were 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.

Appendix D. How to issue MQSC commands on OS/390

The MQSeries for OS/390 commands in this book can be issued from the following sources (except where specifically indicated otherwise):

- The OS/390 console (or equivalent, such as SDSF).

Note: Some commands, such as 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 OS/390 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 OS/390 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 OS/390 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 OS/390 System Management Guide* for more information about issuing commands on MQSeries for OS/390.

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 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.

Issuing Tandem NSK commands

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 Chapter 9, “Administering remote MQSeries objects” in the *MQSeries System Administration* manual for AIX, HP-UX, and Sun Solaris, or the *System Management Guide* for your platform.

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 “Performing local administration tasks using MQSC commands” in the *MQSeries System Administration* manual for AIX, HP-UX, and Sun Solaris, or the *System Management Guide* for your platform.

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. 1993,1999. 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.

Issuing UNIX system commands

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.

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.
No commands have a syntax error.
All valid MQSC commands were 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

Issuing UNIX system commands

The following report is sent to report.out, the output file:

```
5765-115 (C) Copyright IBM Corp. 1993,1999. 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.
No commands have a syntax error.
All valid MQSC commands were 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.

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 Chapter 9, "Administering remote MQSeries objects" in the *MQSeries System Administration* manual.

Many of these functions can also be invoked from the Microsoft management console (MMC) using the MQSeries snap-in applications, or from the Windows NT workstation using the Web administration application supplied with MQSeries. For more information, see "Managing objects with MQSeries" in the *MQSeries Planning Guide*.

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 "Performing local administration tasks using MQSC commands" in 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, 1999. 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.

Issuing Windows NT commands

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 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.
No commands have a syntax error.
All valid MQSC commands were 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

Issuing Windows NT commands

The following report is sent to report.out, the output file:

```
5697-177 (C) Copyright IBM Corp. 1996, 1999. 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.
  No commands have a syntax error.
  All valid MQSC commands were 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.

Appendix I. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

| IBM Director of Licensing
| IBM Corporation
| North Castle Drive
| Armonk, NY 10504-1785
| U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM documentation or non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those documents or Web sites. The materials for those documents or Web sites are not part of the materials for this IBM product and use of those documents or Web sites is at your own risk.

Notices

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
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.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX	CICS	DB2
AS/400	BookManager	IBM
IMS	MQSeries	MVS/ESA
OS/2	OS/400	OS/390
RACF	System/390	VSE/ESA

Lotus Notes is a trademark of Lotus Development Corporation in the United States, or other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

Other company, product, or service names, may be the trademarks or service marks of others.

Index

A

ACTION attribute
 RESET CLUSTER 189
 RESET TPIPE 190
 RESOLVE CHANNEL 192
 RESOLVE INDOUBT 193
 active thread, display 178
 active trace, display list of 180
 administrator commands 213
 alias queue
 alter attributes 38
 define 104
 delete definition 137
 display attributes 168
 ALL attribute
 DISPLAY CHANNEL 144
 DISPLAY CHSTATUS 150
 DISPLAY CLUSQMGR 155
 DISPLAY NAMELIST 161
 DISPLAY PROCESS 162
 DISPLAY QMGR 164
 DISPLAY QUEUE 170
 DISPLAY SECURITY 175
 DISPLAY STGCLASS 176
 ALTDATA attribute
 DISPLAY CHANNEL 144
 DISPLAY CLUSQMGR 156
 DISPLAY NAMELIST 161
 DISPLAY PROCESS 162
 DISPLAY QMGR 165
 DISPLAY QUEUE 172
 DISPLAY STGCLASS 176
 ALLTIME attribute
 DISPLAY CHANNEL 144
 DISPLAY CLUSQMGR 156
 DISPLAY NAMELIST 161
 DISPLAY PROCESS 162
 DISPLAY QMGR 165
 DISPLAY QUEUE 172
 DISPLAY STGCLASS 177
 application identifier
 See APPLICID attribute
 application type
 See APPLTYPE attribute
 APPLICID attribute
 ALTER PROCESS 36
 DEFINE PROCESS 101
 DISPLAY PROCESS 162
 APPLTYPE attribute
 ALTER PROCESS 36
 DEFINE PROCESS 101

APPLTYPE attribute (*continued*)
 DISPLAY PROCESS 162
 AUTHOREV attribute
 ALTER QMGR 51
 DISPLAY QMGR 165
 auto-definition exit program 51, 165
 auto-definition of channels 51, 165
 AUTOSTART attribute
 ALTER CHANNEL 23
 DEFINE CHANNEL 84
 DISPLAY CHANNEL 144

B

backing up the log 71
 backout count, harden
 See HARDENBO attribute
 backout requeue name
 See BOQNAME attribute
 backout threshold
 See BOTHRESH attribute
 BATCHES attribute
 DISPLAY CHSTATUS 152
 BATCHINT attribute
 ALTER CHANNEL 23
 DEFINE CHANNEL 84
 DISPLAY CHANNEL 144
 DISPLAY CLUSQMGR 156
 BATCHSZ attribute
 ALTER CHANNEL 23
 DEFINE CHANNEL 84
 DISPLAY CHANNEL 145
 DISPLAY CHSTATUS 152
 DISPLAY CLUSQMGR 156
 bibliography viii
 BookManager xiii
 bootstrap data set, recover 185
 BOQNAME attribute
 ALTER QLOCAL 43
 ALTER QMODEL 56
 DEFINE QLOCAL 111
 DEFINE QMODEL 120
 DISPLAY QUEUE 172
 BOTHRESH attribute
 ALTER QLOCAL 43
 ALTER QMODEL 56
 DEFINE QLOCAL 111
 DEFINE QMODEL 120
 DISPLAY QUEUE 172
 BSDS, recover 185
 buffer pool number
 See BUFFPOOL attribute, DEFINE PSID

index

- buffer pool, defining 74
 - BUFFERS attribute, DEFINE BUFFPOOL 74
 - BUFFPOOL attribute, DEFINE PSID 103
 - BUFSRCVD attribute, DISPLAY CHSTATUS 152
 - BUFSSSENT attribute, DISPLAY CHSTATUS 152
 - building command scripts 3
 - building commands
 - characters with special meanings 3
 - rules for 1
 - BYTSRCVD attribute, DISPLAY CHSTATUS 152
 - BYTSSSENT attribute, DISPLAY CHSTATUS 152
- ## C
- CCSID attribute
 - ALTER QMGR 51
 - DISPLAY QMGR 165
 - CHAD attribute
 - ALTER QMGR 51
 - DISPLAY QMGR 165
 - CHADEV attribute
 - ALTER QMGR 51
 - DISPLAY QMGR 165
 - CHADEXIT attribute
 - ALTER QMGR 51
 - DISPLAY QMGR 165
 - channel
 - alter attributes 14
 - auto-definition 51, 165
 - commands 217
 - define attributes 75
 - delete definition 134
 - display 142
 - ping 183
 - reset 188
 - resolve 192
 - start 196
 - start initiator 197
 - start listener 199
 - stop 205
 - CHANNEL attribute, DISPLAY CLUSQMGR 155
 - channel initiator
 - start 197
 - stop 207
 - channel status, displaying 147
 - channels, rules for names of 8
 - CHLTYPE attribute
 - ALTER CHANNEL 23
 - DEFINE CHANNEL 84
 - DISPLAY CHANNEL 145
 - CHSTADA attribute, DISPLAY CHSTATUS 153
 - CHSTATI attribute, DISPLAY CHSTATUS 153
 - CLASS attribute
 - ALTER TRACE 70
 - DISPLAY TRACE 181
 - START TRACE 202
 - CLASS attribute (*continued*)
 - STOP TRACE 212
 - CLUSDATE attribute
 - DISPLAY CLUSQMGR 155
 - DISPLAY QUEUE 172
 - CLUSINFO attribute, DISPLAY QUEUE 170
 - CLUSNL attribute
 - ALTER CHANNEL 24
 - ALTER QALIAS 39
 - ALTER QLOCAL 43
 - ALTER QREMOTE 64
 - DEFINE CHANNEL 85
 - DEFINE QALIAS 106
 - DEFINE QLOCAL 111
 - DEFINE QREMOTE 129
 - DISPLAY CHANNEL 145
 - DISPLAY QUEUE 170, 172
 - RESUME QMGR 194
 - SUSPEND QMGR 213
 - CLUSQMGR attribute, DISPLAY QUEUE 172
 - CLUSQT attribute, DISPLAY QUEUE 173
 - cluster
 - commands 218
 - refresh 186
 - reset 189
 - CLUSTER attribute
 - ALTER CHANNEL 24
 - ALTER QALIAS 39
 - ALTER QLOCAL 43
 - ALTER QREMOTE 64
 - DEFINE CHANNEL 85
 - DEFINE QALIAS 106
 - DEFINE QLOCAL 111
 - DEFINE QREMOTE 129
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 155
 - DISPLAY QUEUE 170, 173
 - RESUME QMGR 194
 - SUSPEND QMGR 213
 - cluster queue manager, display 154
 - clusters, rules for names of 8
 - CLUSTIME attribute
 - DISPLAY CLUSQMGR 155
 - DISPLAY QUEUE 173
 - CLWLDATA attribute
 - ALTER QMGR 52
 - DISPLAY QMGR 165
 - CLWLEXIT attribute
 - ALTER QMGR 52
 - DISPLAY QMGR 165
 - CLWLEN attribute
 - ALTER QMGR 52
 - DISPLAY QMGR 165
 - CMDLEVEL attribute, DISPLAY QMGR 165
 - coded character set identifier 51
 - See also* CCSID attribute

- command
 - summary 215
- command input queue name
 - See COMMANDQ attribute, DISPLAY QMGR
- command prefix string 229
- command scripts, building 3
- command server
 - display status 158
 - start 198
 - stop 208
- command string
 - entering quotes 2
 - preserving case 2
- COMMANDQ attribute, DISPLAY QMGR 165
- commands 213
 - rules for building 1
 - rules for naming objects in 5
 - rules for using 1
 - synonym 2
- COMMENT attribute
 - ALTER TRACE 70
 - DISPLAY TRACE 180
 - START TRACE 201
 - STOP TRACE 211
- CONNAME attribute
 - ALTER CHANNEL 24
 - DEFINE CHANNEL 85
 - DISPLAY CHANNEL 145
 - DISPLAY CHSTATUS 150
 - DISPLAY CLUSQMGR 156
- CONVERT attribute
 - ALTER CHANNEL 26
 - DEFINE CHANNEL 87
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 156
- CPF 229
- CPILEVEL attribute, DISPLAY QMGR 165
- CRDATE attribute, DISPLAY QUEUE
- creation date
 - See CRDATE attribute, DISPLAY QUEUE
- creation time
 - See CRTIME attribute, DISPLAY QUEUE
- CRTIME attribute, DISPLAY QUEUE 173
- CURDEPTH attribute, DISPLAY QUEUE 173
- CURLUWID attribute, DISPLAY CHSTATUS 151
- CURMSGs attribute, DISPLAY CHSTATUS 151
- CURRENT attribute, DISPLAY CHSTATUS 150
- current queue depth
 - See CURDEPTH attribute, DISPLAY QUEUE
- CURSEQNO attribute, DISPLAY CHSTATUS 151

- D**
- DATALEN attribute, PING CHANNEL 183
- dead-letter queue name
 - See DEADQ attribute
- DEADQ attribute
 - ALTER QMGR 52
 - DISPLAY QMGR 166
- 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
- DEFBIND attribute
 - ALTER QALIAS 39
 - ALTER QLOCAL 43
 - ALTER QREMOTE 64
 - DEFINE QALIAS 106
 - DEFINE QLOCAL 112
 - DEFINE QREMOTE 129
 - DISPLAY QUEUE 173
- DEFPRTY attribute
 - ALTER QALIAS 38
 - ALTER QLOCAL 42
 - ALTER QMODEL 56
 - ALTER QREMOTE 64
 - DEFINE QALIAS 105
 - DEFINE QLOCAL 110
 - DEFINE QMODEL 119
 - DEFINE QREMOTE 128
 - DISPLAY QUEUE 173
- DEFPSIST attribute
 - ALTER QALIAS 39
 - ALTER QLOCAL 42
 - ALTER QMODEL 56
 - ALTER QREMOTE 64
 - DEFINE QALIAS 105
 - DEFINE QLOCAL 110
 - DEFINE QMODEL 119
 - DEFINE QREMOTE 128
 - DISPLAY QUEUE 173
- DEFSOPT attribute
 - ALTER QLOCAL 43
 - ALTER QMODEL 57
 - DEFINE QLOCAL 112
 - DEFINE QMODEL 120
 - DISPLAY QUEUE 173
- DEFTYPE attribute
 - ALTER QMODEL 62
 - DEFINE QMODEL 126
 - DISPLAY CLUSQMGR 155
 - DISPLAY QUEUE 173
- DEFXMITQ attribute
 - ALTER QMGR 52
 - DISPLAY QMGR 166

index

DESCR attribute

- ALTER CHANNEL 26
- ALTER NAMELIST 35
- ALTER PROCESS 37
- ALTER QALIAS 39
- ALTER QLOCAL 42
- ALTER QMGR 52
- ALTER QMODEL 56
- ALTER QREMOTE 64
- ALTER STGCLASS 68
- DEFINE CHANNEL 87
- DEFINE NAMELIST 99
- DEFINE PROCESS 102
- DEFINE QALIAS 105
- DEFINE QLOCAL 111
- DEFINE QMODEL 120
- DEFINE QREMOTE 129
- DEFINE STGCLASS 132
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 156
- DISPLAY NAMELIST 161
- DISPLAY PROCESS 163
- DISPLAY QMGR 166
- DISPLAY QUEUE 173
- DISPLAY STGCLASS 177

description

See DESCR attribute

DEST attribute

- DISPLAY TRACE 181
- START TRACE 202
- STOP TRACE 212

DETAIL attribute, DISPLAY TRACE 180

Digital OpenVMS

- error messages 223
- example command input file 222
- example report file 223
- issuing commands from a command file 222
- issuing commands interactively 221
- runmqsc command 221
- verifying commands 224

directing OS/390 commands 230

DISCINT attribute

- ALTER CHANNEL 26
- DEFINE CHANNEL 87
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 156

DISTL attribute

- ALTER QLOCAL 44
- ALTER QMODEL 57
- DEFINE QLOCAL 112
- DEFINE QMODEL 120
- DISPLAY QMGR 166
- DISPLAY QUEUE 173

dual BSDS, reestablish 185

E

environment information

See ENVRDATA attribute

ENVPARM attribute

- START CHINIT 197
- START QMGR 200

ENVRDATA attribute

- ALTER PROCESS 37
- DEFINE PROCESS 102
- DISPLAY PROCESS 163

error messages

- Digital OpenVMS 223
- OS/2 Warp 227
- Tandem NSK 235
- UNIX systems 240
- Windows NT 244

example OS/400 MQSeries command 231

examples

- Digital OpenVMS command input file 222
- Digital OpenVMS report file 223
- OS/2 Warp command input file 226
- OS/2 Warp report file 227
- OS/400 command input file 231
- OS/400 report file 232
- Tandem NSK command input file 234
- Tandem NSK report file 235
- UNIX systems command input file 239
- UNIX systems report file 240
- Windows NT command input file 243
- Windows NT report file 244

excessive backout requeue name

See BOQNAME attribute

F

FIFO queue

- ALTER QLOCAL 45
- ALTER QMODEL 58
- DEFINE QLOCAL 114
- DEFINE QMODEL 122

FORCE option

- ALTER QALIAS 38
- ALTER QLOCAL 42
- ALTER QMGR 50
- ALTER QREMOTE 63

G

GET attribute

- ALTER QALIAS 40
- ALTER QLOCAL 44
- ALTER QMODEL 57
- DEFINE QALIAS 106
- DEFINE QLOCAL 112
- DEFINE QMODEL 120
- DISPLAY QUEUE 173

getting help when issuing commands

- OS/2 Warp 226
- UNIX systems 238
- Windows NT 242

H

handles indicating queue open for input

See IPPROCS attribute, DISPLAY QUEUE

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 45
- ALTER QMODEL 59
- DEFINE QLOCAL 114
- DEFINE QMODEL 122
- DISPLAY QUEUE 173

HBINT attribute

- ALTER CHANNEL 26, 87
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 156

how to issue commands

- Digital OpenVMS 221
- OS/2 Warp 225
- OS/390 229
- OS/400 231
- Tandem NSK 233
- UNIX systems 237
- Windows NT 241

HTML (Hypertext Markup Language) xiii

Hypertext Markup Language (HTML) xiii

I

IFCID attribute

- ALTER TRACE 70
- START TRACE 203

IMS Tpipe, reset sequence numbers manually 190

in-doubt thread

- display 178
- resolve manually 193

INDOUBT attribute, DISPLAY CHSTATUS 151

INDXTYPE attribute

- ALTER QLOCAL 44
- ALTER QMODEL 57
- DEFINE QLOCAL 112
- DEFINE QMODEL 121
- DISPLAY QUEUE 173

INHIBTEV attribute

- ALTER QMGR 53
- DISPLAY QMGR 166

initiation queue name

See INITQ attribute

INITQ attribute

- ALTER QLOCAL 45
- ALTER QMODEL 58
- DEFINE QLOCAL 113
- DEFINE QMODEL 121
- DISPLAY QUEUE 173
- START CHINIT 197

INTERVAL attribute

- ALTER SECURITY 67
- DISPLAY SECURITY 175

IPPROCS attribute, DISPLAY QUEUE 173

J

JOBNAME attribute, DISPLAY CHSTATUS 153

L

LIKE option

- DEFINE CHANNEL 87
- DEFINE NAMELIST 98
- DEFINE PROCESS 100
- DEFINE QALIAS 104
- DEFINE QLOCAL 110
- DEFINE QMODEL 119
- DEFINE QREMOTE 128
- DEFINE STGCLASS 132

list of queue names

- alter 35
- define 98
- delete 135
- display 161

listener

- start 199
- stop 209

local queue

- alter attributes 41
- clear 73
- define 108
- delete definition 138
- display attributes 168

LOCALEV attribute

- ALTER QMGR 53
- DISPLAY QMGR 166

log, archive 71

LONGRTS attribute, DISPLAY CHSTATUS 153

LONGRTY attribute

- ALTER CHANNEL 26
- DEFINE CHANNEL 88
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 156

LONGTMR attribute

- ALTER CHANNEL 27
- DEFINE CHANNEL 88

index

- LONGTMR attribute (*continued*)
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 156
 - LSTLUWID attribute, DISPLAY CHSTATUS 151
 - LSTMSGDA attribute, DISPLAY CHSTATUS 153
 - LSTMSGTI attribute, DISPLAY CHSTATUS 153
 - LSTSEQNO attribute, DISPLAY CHSTATUS 151
- ## M
- MAXDEPTH attribute
 - ALTER QLOCAL 45
 - ALTER QMODEL 58
 - DEFINE QLOCAL 113
 - DEFINE QMODEL 121
 - DISPLAY QUEUE 173
 - MAXHANDS attribute
 - ALTER QMGR 53
 - DISPLAY QMGR 166
 - maximum message depth
 - See MAXDEPTH attribute
 - maximum message length
 - See MAXMSGL attribute
 - maximum number of messages, define 97
 - maximum number of open handles
 - See MAXHANDS attribute
 - maximum priority
 - See MAXPRTY attribute, DISPLAY QMGR
 - MAXMSGL attribute
 - ALTER CHANNEL 27
 - ALTER QLOCAL 45
 - ALTER QMGR 53
 - ALTER QMODEL 58
 - DEFINE CHANNEL 88
 - DEFINE QLOCAL 113
 - DEFINE QMODEL 122
 - DISPLAY CHANNEL 145
 - DISPLAY CHSTATUS 153
 - DISPLAY CLUSQMGR 156
 - DISPLAY QMGR 166
 - DISPLAY QUEUE 173
 - MAXPRTY attribute, DISPLAY QMGR 166
 - maxsmsgs
 - define 97
 - display 160
 - MAXUMSGS attribute, ALTER QMGR 53
 - MCANAME attribute
 - ALTER CHANNEL 27
 - DEFINE CHANNEL 88
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 156
 - MCASTAT attribute, DISPLAY CHSTATUS 153
 - MCATYPE attribute
 - ALTER CHANNEL 27
 - DEFINE CHANNEL 89
 - DISPLAY CHANNEL 145
 - MCATYPE attribute (*continued*)
 - DISPLAY CLUSQMGR 156
 - MCAUSER attribute
 - ALTER CHANNEL 27
 - DEFINE CHANNEL 89
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 156
 - 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 71
 - STOP CHANNEL 205
 - STOP QMGR 210
 - SUSPEND QMGR 213
 - model queue
 - alter attributes 55
 - define 118
 - delete definition 139
 - display attributes 168
 - MODENAME attribute
 - ALTER CHANNEL 28
 - DEFINE CHANNEL 89
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 157
 - MQSC commands
 - how to issue on Digital OpenVMS 221
 - how to issue on OS/2 Warp 225
 - how to issue on OS/390 229
 - how to issue on OS/400 231
 - how to issue on Tandem NSK 233
 - how to issue on UNIX systems 237
 - how to issue on Windows NT 241
 - MQSeries commands 213
 - MQSeries for AS/400, how to issue commands 231
 - MQSeries for OS/2 Warp, how to issue commands 225
 - MQSeries for OS/390 commands
 - directing to the correct queue manager 230
 - how to issue 229
 - MQSeries for Windows NT, how to issue commands 241
 - MQSeries on Digital OpenVMS, how to issue commands 221
 - MQSeries on Tandem NSK, how to issue commands 233

MQSeries on UNIX systems, how to issue commands 237

MQSeries publications viii

MRDATA attribute

- ALTER CHANNEL 28
- DEFINE CHANNEL 89
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 157

MREXIT attribute

- ALTER CHANNEL 28
- DEFINE CHANNEL 89
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 157

MRRTY attribute

- ALTER CHANNEL 28
- DEFINE CHANNEL 90
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 157

MRTMR attribute

- ALTER CHANNEL 28
- DEFINE CHANNEL 90
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 157

MSGDATA attribute

- ALTER CHANNEL 29
- DEFINE CHANNEL 90
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 157

MSGDLVSQ attribute

- ALTER QLOCAL 45
- ALTER QMODEL 58
- DEFINE QLOCAL 114
- DEFINE QMODEL 122
- DISPLAY QUEUE 173

MSGEXIT attribute

- ALTER CHANNEL 29
- DEFINE CHANNEL 90
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 157

MSGS attribute, DISPLAY CHSTATUS 153

N

NAMCOUNT attribute, DISPLAY NAMELIST 161

name spaces 5

namelist

- alter 35
- commands 217
- define 98
- delete 135
- display contents 161
- rules for names of 8

NAMES attribute

- ALTER NAMELIST 35
- DEFINE NAMELIST 99
- DISPLAY NAMELIST 161

NETPRTY attribute

- ALTER CHANNEL 30
- DEFINE CHANNEL 91
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 157

network identifier

See NID attribute, RESOLVE INDOUBT

NID attribute, RESOLVE INDOUBT 193

NOHARDENBO attribute

- ALTER QLOCAL 45
- ALTER QMODEL 59
- DEFINE QLOCAL 114
- DEFINE QMODEL 122

NOPURGE attribute, DELETE QLOCAL 138

NOREPLACE option

- DEFINE CHANNEL 91
- DEFINE NAMELIST 98
- DEFINE PROCESS 101
- DEFINE QALIAS 105
- DEFINE QLOCAL 110
- DEFINE QMODEL 119
- DEFINE QREMOTE 128
- DEFINE STGCLASS 133

NOSHARE attribute

- ALTER QLOCAL 46
- ALTER QMODEL 59
- DEFINE QLOCAL 114
- DEFINE QMODEL 122

NOTRIGGER attribute

- ALTER QLOCAL 46
- ALTER QMODEL 59
- DEFINE QLOCAL 114
- DEFINE QMODEL 123

NPMSPEED attribute

- ALTER CHANNEL 30
- DEFINE CHANNEL 91
- DISPLAY CHANNEL 145
- DISPLAY CLUSQMGR 157

O

objects, reserved names 8

operator commands 213

OPPROCS attribute

- DISPLAY QUEUE 173

OS/2 Warp

- error messages 227
- example command input file 226
- example report file 227
- getting help when issuing commands 226
- issuing commands from a command file 226
- issuing commands interactively 225
- runmqsc command 225
- verifying commands 228

OS/390 commands

- directing to the correct queue manager 230

index

OS/390 commands (*continued*)
 how to issue 229
OS/390 trace
 alter events being traced 70
 display list of active traces 180
 start 201
 stop 211
OS/400
 example command input file 231
 example report file 232
 how to issue commands 231
 verifying commands 232
OS/400 MQSeries command, example 231

P

page set
 define 103, 132
 display usage 182
page set identifier
 See PSID attribute
PARM attribute
 START CHINIT 197
 START QMGR 200
PASSWORD attribute
 ALTER CHANNEL 30
 DEFINE CHANNEL 92
 DISPLAY CHANNEL 145
 DISPLAY CLUSQMGR 157
PDF (Portable Document Format) xiii
PERFMEV attribute
 ALTER QMGR 53
 DISPLAY QMGR 166
persistence on restart
 See DEFPSIST attribute
PLATFORM attribute, DISPLAY QMGR 166
Portable Document Format (PDF) xiii
PostScript format xiv
priority queue
 ALTER QLOCAL 45
 ALTER QMODEL 58
 DEFINE QLOCAL 114
 DEFINE QMODEL 122
priority, maximum
 See MAXPRTY attribute, DISPLAY QMGR
PROCESS attribute
 ALTER QLOCAL 46
 ALTER QMODEL 59
 DEFINE QLOCAL 114
 DEFINE QMODEL 123
 DISPLAY QUEUE 174
process definition
 alter 36
 commands 216
 define 100
 delete 136

process definition (*continued*)
 display 162
processes, rules for names of 8
PSID attribute
 ALTER STGCLASS 68
 DEFINE STGCLASS 133
 DISPLAY STGCLASS 176
 DISPLAY USAGE 182
publications, MQSeries viii
PURGE attribute, DELETE QLOCAL 138
PUT attribute
 ALTER QALIAS 39
 ALTER QLOCAL 42
 ALTER QMODEL 56
 ALTER QREMOTE 64
 DEFINE QALIAS 105
 DEFINE QLOCAL 111
 DEFINE QMODEL 120
 DEFINE QREMOTE 129
 DISPLAY QUEUE 174
PUTAUT attribute
 ALTER CHANNEL 30
 DEFINE CHANNEL 92
 DISPLAY CHANNEL 145
 DISPLAY CLUSQMGR 157

Q

QDEPTHHI attribute
 ALTER QLOCAL 46
 ALTER QMODEL 59
 DEFINE QLOCAL 115
 DEFINE QMODEL 123
 DISPLAY QUEUE 174
QDEPTHLO attribute
 ALTER QLOCAL 46
 ALTER QMODEL 59
 DEFINE QLOCAL 115
 DEFINE QMODEL 123
 DISPLAY QUEUE 174
QDPHIEV attribute
 ALTER QLOCAL 46
 ALTER QMODEL 60
 DEFINE QLOCAL 115
 DEFINE QMODEL 123
 DISPLAY QUEUE 174
QDPLOEV attribute
 ALTER QLOCAL 47
 ALTER QMODEL 60
 DEFINE QLOCAL 115
 DEFINE QMODEL 124
 DISPLAY QUEUE 174
QDPMAXEV attribute
 ALTER QLOCAL 47
 ALTER QMODEL 60
 DEFINE QLOCAL 115

QDPMAXEV attribute (*continued*)

DEFINE QMODEL 124
 DISPLAY QUEUE 174

QMID attribute

DISPLAY CLUSQMGR 155
 DISPLAY QMGR 166
 DISPLAY QUEUE 174

QMNAME attribute

ALTER CHANNEL 31
 DEFINE CHANNEL 92
 DISPLAY CHANNEL 145
 DISPLAY QMGR 166
 RESET CLUSTER 189

QMTYPE attribute, DISPLAY CLUSQMGR 155

QSVCIIEV attribute

ALTER QLOCAL 47
 ALTER QMODEL 60
 DEFINE QLOCAL 116
 DEFINE QMODEL 124
 DISPLAY QUEUE 174

QSVCIINT attribute

ALTER QLOCAL 47
 ALTER QMODEL 60
 DEFINE QLOCAL 116
 DEFINE QMODEL 124
 DISPLAY QUEUE 174

QTYPE attribute, DISPLAY QUEUE 174

queue

commands 216

queue attributes, display 168

queue creation date

See CRDATE attribute, DISPLAY QUEUE

queue creation time

See CRTIME attribute, DISPLAY QUEUE

queue depth

See CURDEPTH attribute, DISPLAY QUEUE

queue manager

alter attributes 50

commands 215

directing OS/390 commands to 230

display attributes 164

name

See QMNAME attribute

ping 184

resume 194

start 200

stop 210

suspend 213

queue names 6

queue type, default

See DEFTYPE attribute

queue-manager alias, defining 127

queues

reserved names 6

rules for names of 6

R

railroad diagrams, how to read 9

RCVDATA attribute

ALTER CHANNEL 31
 DEFINE CHANNEL 92
 DISPLAY CHANNEL 145
 DISPLAY CLUSQMGR 157

RCVEXIT attribute

ALTER CHANNEL 31
 DEFINE CHANNEL 93
 DISPLAY CHANNEL 145
 DISPLAY CLUSQMGR 157

RCVSEQ attribute, RESET TPIPE 191

remote queue

alter attributes 63

define 127

delete definition 140

display attributes 168

remote queue manager name

See RQMNAME attribute

remote queue name

See RNAME attribute

REMOTEEV attribute

ALTER QMGR 53
 DISPLAY QMGR 167

REPLACE option

DEFINE CHANNEL 91
 DEFINE NAMELIST 98
 DEFINE PROCESS 101
 DEFINE QALIAS 105
 DEFINE QLOCAL 110
 DEFINE QMODEL 119
 DEFINE QREMOTE 128
 DEFINE STGCLASS 133

reply-to queue alias, defining 127

REPOS attribute

ALTER QMGR 54
 DISPLAY QMGR 167

REPOSNL attribute

ALTER QMGR 54
 DISPLAY QMGR 167

reserved names

objects 8

queues 6

resource manager identifier

See RMID attribute

retention interval

See RETINTVL attribute

RETINTVL attribute

ALTER QLOCAL 47
 ALTER QMODEL 61
 DEFINE QLOCAL 116
 DEFINE QMODEL 124
 DISPLAY QUEUE 174

index

- RMID attribute
 - DISPLAY TRACE 181
 - START TRACE 203
 - STOP TRACE 212
- RNAME attribute
 - ALTER QREMOTE 65
 - DEFINE QREMOTE 130
 - DISPLAY QUEUE 174
- RQMNAME attribute
 - ALTER QREMOTE 65
 - DEFINE QREMOTE 130
 - DISPLAY QUEUE 174
- rules for using commands 1
- runmqsc command
 - using on Digital OpenVMS 221
 - using on OS/2 Warp 225
 - using on Tandem NSK 233
 - using on UNIX systems 237
 - using on Windows NT 241
- S**
- SAVED attribute, DISPLAY CHSTATUS 150
- SCOPE attribute
 - ALTER QALIAS 40
 - ALTER QLOCAL 48
 - ALTER QREMOTE 65
 - DEFINE QALIAS 106
 - DEFINE QLOCAL 116
 - DEFINE QREMOTE 130
 - DISPLAY QUEUE 174
- SCYDATA attribute
 - ALTER CHANNEL 31
 - DEFINE CHANNEL 93
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 157
- SCYEXIT attribute
 - ALTER CHANNEL 32
 - DEFINE CHANNEL 93
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 157
- security
 - alter attributes 67
 - commands 218, 219
 - display attributes 175
 - rebuild 187
 - refresh 187
 - reverify 195
- SENDDATA attribute
 - ALTER CHANNEL 32
 - DEFINE CHANNEL 93
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 157
- SENDEXIT attribute
 - ALTER CHANNEL 32
 - DEFINE CHANNEL 94
- SENDEXIT attribute (*continued*)
 - DISPLAY CHANNEL 145
 - DISPLAY CLUSQMGR 157
- SENDSEQ attribute, RESET TPIPE 190
- SEQNUM attribute, RESET CHANNEL 188
- sequence numbers
 - resetting on an IMS Tpipe 190
- SEQWRAP attribute
 - ALTER CHANNEL 32
 - DEFINE CHANNEL 94
 - DISPLAY CHANNEL 146
 - DISPLAY CLUSQMGR 157
- SHARE attribute
 - ALTER QLOCAL 46
 - ALTER QMODEL 59
 - DEFINE QLOCAL 114
 - DEFINE QMODEL 122
 - DISPLAY QUEUE 174
- share options, default
 - See* DEFSOPT attribute
- SHORTRTS attribute, DISPLAY CHSTATUS 153
- SHORTRTY attribute
 - ALTER CHANNEL 33
 - DEFINE CHANNEL 94
 - DISPLAY CHANNEL 146
 - DISPLAY CLUSQMGR 157
- SHORTTMR attribute
 - ALTER CHANNEL 33
 - DEFINE CHANNEL 94
 - DISPLAY CHANNEL 146
 - DISPLAY CLUSQMGR 157
- softcopy books xiii
- STATUS attribute
 - DISPLAY CHSTATUS 151
 - DISPLAY CLUSQMGR 155
- STGCLASS attribute
 - ALTER QLOCAL 48
 - ALTER QMODEL 61
 - DEFINE QLOCAL 117
 - DEFINE QMODEL 125
 - DISPLAY QUEUE 170, 174
- STOPREQ attribute, DISPLAY CHSTATUS 153
- storage class
 - alter 68
 - define 132
 - delete 141
 - display 176
- storage class name
 - See* STGCLASS attribute
- storage classes, rules for names of 8
- STRSTPEV attribute
 - ALTER QMGR 54
 - DISPLAY QMGR 167
- summary of commands 215
- SUSPEND attribute, DISPLAY CLUSQMGR 156

SWITCHES attribute, DISPLAY SECURITY 175
 syncpoint support
 See SYNCPT attribute, DISPLAY QMGR
 SYNCPT attribute, DISPLAY QMGR 167
 syntax diagrams, how to read 9
 system-command input queue name
 See COMMANDQ attribute, DISPLAY QMGR

T

Tandem NSK

error messages 235
 example command input file 234
 example report file 235
 issuing commands from a command file 234
 issuing commands interactively 233
 runmqsc command 233
 verifying commands 236

target queue name

 See TARGQ attribute

TARGQ attribute

ALTER QALIAS 40
 DEFINE QALIAS 107
 DISPLAY QUEUE 174

TDATA attribute, START TRACE 204

thread

display information about 178
 resolving in-doubt manually 193

TIME attribute, ARCHIVE LOG 71

TIMEOUT attribute

ALTER SECURITY 67
 DISPLAY SECURITY 175

TNO attribute

ALTER TRACE 70
 DISPLAY TRACE 181
 STOP TRACE 212

TPNAME attribute

ALTER CHANNEL 33
 DEFINE CHANNEL 95
 DISPLAY CHANNEL 146
 DISPLAY CLUSQMGR 157

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 48
 ALTER QMODEL 61
 DEFINE QLOCAL 117
 DEFINE QMODEL 125
 DISPLAY QUEUE 174

TRIGDPTH attribute

ALTER QLOCAL 48
 ALTER QMODEL 61
 DEFINE QLOCAL 117
 DEFINE QMODEL 125
 DISPLAY QUEUE 174

TRIGGER attribute

ALTER QLOCAL 46
 ALTER QMODEL 59
 DEFINE QLOCAL 114
 DEFINE QMODEL 123
 DISPLAY QUEUE 174

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 54
 DISPLAY QMGR 167

TRIGMPRI attribute

ALTER QLOCAL 49
 ALTER QMODEL 61
 DEFINE QLOCAL 117
 DEFINE QMODEL 125
 DISPLAY QUEUE 174

TRIGTYPE attribute

ALTER QLOCAL 49
 ALTER QMODEL 61
 DEFINE QLOCAL 117
 DEFINE QMODEL 125
 DISPLAY QUEUE 174

TRPTYPE attribute

ALTER CHANNEL 34
 DEFINE CHANNEL 95
 DISPLAY CHANNEL 146
 DISPLAY CLUSQMGR 157

TYPE attribute

DISPLAY CHANNEL 144
 DISPLAY QUEUE 170
 DISPLAY THREAD 178

U

undelivered-message queue name

 See DEADQ attribute

unit-of-work ID, display 178

UNIX systems

error messages 240

index

UNIX systems (*continued*)

- example command input file 239
 - example report file 240
 - getting help when issuing commands 238
 - issuing commands from a command file 238
 - issuing commands interactively 237
 - runmqsc command 237
 - verifying commands 240
- USAGE attribute
- ALTER QLOCAL 49
 - ALTER QMODEL 62
 - DEFINE QLOCAL 117
 - DEFINE QMODEL 125
 - DISPLAY QUEUE 174
- usage, page set
- display 182
- USERDATA attribute
- ALTER PROCESS 37
 - DEFINE PROCESS 102
 - DISPLAY PROCESS 163
- USERID attribute
- ALTER CHANNEL 34
 - DEFINE CHANNEL 95
 - DISPLAY CHANNEL 146
 - DISPLAY CLUSQMGR 157
 - DISPLAY TRACE 181
 - START TRACE 204
 - STOP TRACE 212
- using commands
- rules for 1

V

- verifying commands
- Digital OpenVMS 224
 - OS/2 Warp 228
 - OS/400 232
 - Tandem NSK 236
 - UNIX systems 240
 - Windows NT 244

W

- WAIT attribute, ARCHIVE LOG 71
- Windows Help xiv
- Windows NT
- error messages 244
 - example command input file 243
 - example report file 244
 - getting help when issuing commands 242
 - issuing commands from a command file 242
 - issuing commands interactively 241
 - runmqsc command 241
 - verifying commands 244

X

- XCFGNAME attribute
- ALTER STGCLASS 69
 - DEFINE STGCLASS 133
 - DISPLAY STGCLASS 177
 - RESET TPIPE 191
- XCFMNAME attribute
- ALTER STGCLASS 69
 - DEFINE STGCLASS 133
 - DISPLAY STGCLASS 177
 - RESET TPIPE 190
- XMITQ attribute
- ALTER CHANNEL 34
 - ALTER QREMOTE 66
 - DEFINE CHANNEL 96
 - DEFINE QREMOTE 131
 - DISPLAY CHANNEL 146
 - DISPLAY CHSTATUS 149
 - DISPLAY QUEUE 174

Sending your comments to IBM

MQSeries®

Command Reference

SC33-1369-10

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: HURSLEY(IDRCF)
 - Internet: idrcf@hursley.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

SC33-1369-10

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	

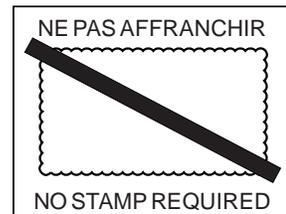
If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

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.

SC33-1369-10

